NDNFinfo

Schafer's Wafers

2010-2011
EE Senior Design

Rob Maurer
John Plunkett
Barbara Raynal
Matt Schueler

2. Table of Contents

3. Introduction

3.1 Problem Statement:

The addition of Stinson-Remick Hall and the new Notre Dame Nanofabrication Facility has created a lot of excitement around the electrical engineering department. The new cleanroom has been in use for 8 months, and the transition towards becoming a functional, full-time research facility is nearly complete. The new facility boasts ultra-clean class 1000 and class 100 areas (100 particles in the air per cubic meter). Inside these areas, users and technicians are required to wear coveralls, gloves, masks, and booties to contain unwanted particles and ions, which could damage devices. Since every user in the cleanroom is wearing containment equipment, it is extremely difficult to quickly identify who is working using the lab at any one time. There is also the additional safety hazard of not knowing which users are in the lab at a given time. In case there is a fire or chemical spill, it would be valuable for rescue teams to know who was in the lab at the time of the accident.

Within the cleanroom, there are dozens of machines used for fabrication. In order to facilitate the use of relatively few machines by hundreds of users, an online database system called Coral was set up. This system allows users to make machine reservations and see which machines are malfunctioning from outside the lab. As it stands, users must download the remote Coral access Java applet in order to see the availability of machines. This system has slowly been integrated into use this year, but currently is not as user-friendly as it could ideally be.

The lab is now used daily by many different research groups and advisors from multiple departments. The Lenel server at the lock shop on campus keeps a running tab of when users enter or exit the cleanroom. This is important because advisors are billed for each hour their student spends in the cleanroom, but there is not much freedom for advisors to conveniently see how much time they will be billed for. This prevents advisors from getting the in-depth view of their group's overall and individual user lab activities. It would also be convenient for advisors to view how their students are spending their time in the lab to make sure that they are working on the machines necessary for their project.

Additionally, when a machine breaks in the lab, sometimes it is difficult to know who it was that caused the problem. Records of machine use are kept in the Coral PSQL database, but there is currently no user-friendly method available of accessing this information. Finally, in order to properly bill research groups, it is necessary for lab users to swipe out with their RFID card when they are finished using the facility. However, users often do not remember this extra step, and at times they do not swipe out. It would be convenient for users to have a quick method of checking whether or not they have properly exited from the lab, as this would save them the time of walking back to the lab. It could also save their research group money by letting them know that they are being billed even though they are out of the lab.

To address these problems, we proposed the idea of a main display located outside the cleanroom in the Stinson-Remick lobby. This would display a list of the users currently working in the lab, a list of the “Super Users” for the week that have spent the most time for the week using the cleanroom, and a plot of usage activity over the past several hours to monitor activity. In addition to this main requirement, we decided to implement NDNFinfo, a convenient virtual portal to the Notre Dame Nanofabrication Facility. The main page of the NDNfinfo website shows a list of users who are currently working in the cleanroom. To the side, a list of the ten users who have logged the most cleanroom time for the day and for all time is displayed. The web site also has a “graphing central” area, which allows a cleanroom-registered user to create an ndnfinfo account, and to create custom groups of users from a list. Creating groups gives the user the option of easily making customized graphs and reports. Cleanroom usage can be plotted for each user of the group for each day of a selected time period. In addition, one can plot machine usage over a certain time period, organized by user or machine. We supplemented the network with an Android app. The Android and iPhone apps allow the user to check current lab users, top users, machine statuses, reservations, session histories, and plots of machine activities. We believe that when implemented by the department, NDNFinfo can greatly increase the efficiency of the Nanofabrication facility.

3.2 System Requirements:

Originally, our primary goal was simply to be able to display the names of current users on a monitor outside of the cleanroom. The secondary goal was to create extra functions like listing top users and showing machine statuses and to have a web site that could interact with our display system. As we worked through the design process this semester, we realized that we needed to maintain an organized database for record storage if we were going to be able to list top users or create any sort of useful reports. As we created the database, added to it, and improved upon the original design, we realized that it had much more potential to be useful than we had originally planned.

Ultimately, we decided that our system should be able to accurately keep track of cleanroom usage logs and actively track current users in the lab. If it has been shut off for a period of time, it should be able to read in back-logged emails, and correctly modify the database to accommodate for missed log entries so that it is easy to turn on or off without losing important information. The web site should be user-friendly, and should allow for customizable graphs of usage for machines and for log time on demand. If used correctly, it should be 100% accurate and correct. We must also be careful not to violate privacy laws or compromise the security of official records in systems that we communicate with.

Originally, we intended to have an Android app with limited functionality (in other words, a login screen and an area to display current users). However, with the addition of the MySQL database and the Coral account created, we realized that we could extend the functionality to include machine reservations, login histories, and graphing. We also realized that it would be helpful to include an iPhone app to extend these abilities to users without Android phones. Therefore, we created this app later after finishing the Android app.

3.3 High Level Description:

As a user scans in or out of the cleanroom, the Lenel server in the lock shop receives the data of who entered or exited and what time the swipe occurred. This is saved for official records and billing. With the help of Andy Tripp, manager of lock shop services, there is a program implemented that automatically forwards these records to an email address on the Notre Dame server (ndnfinfo.nd.edu). Every time someone scans into or out of the cleanroom, ndnfinfo receives an email outlining the user, the date, the time of the swipe, and whether they are entering or exiting the cleanroom.

The bulk of our design process was software-based, and all of the software for tracking records and hosting the website exists on the Zotac MAG single board computer. The main program for tracking records is called “main_program.cpp”. This program calls the mail client, fetchmail, to pipe the current emails into base-64 encoded .txt files. This is run through a decoder, and the important values are extracted. If an email is processed, the important data is sent to a PHP script called database_code, which automatically organizes and updates the database. Once an email has been processed, it is removed from the mailbox and the .txt file is deleted so that log entries are not recorded multiple times.

The records are stored in a MySQL database called ndnfinfo. The database has two global tables called “users” and “users2”. “users” has 2 columns, “user” and “tin”, which show the names of all users presently in the cleanroom and when they logged in respectively. “users2” holds more permanent records. All users that have logged in or out since our system began recording are in this table. This table stores their most recent scan date, their total time logged for that day, and their all-time total log time.

In addition to the two global tables, each user has 2 user-independent tables. For a user with first name, “User”, and last name, “Name”, one table is called

"UserName" and the other is "UserName_By_Day" . "UserName" is a table that records the entry and exit time of each individual session. "UserName_By_Day" records the total time that the user logged per day, listed by day.

The database also holds a table called "logins" and a table called "Groups", which are both important to the functionality of the website. "logins" is a record of registered website users and passwords. "Groups" keeps track of which groups have been created, and which user created them. Each group then has its own table that lists the users that are a part of that group.

The website is an extra resource that cleanroom users can visit to obtain more information. In order to use the website, an account must be created. When the button "Create Account" is clicked, a pop up window that asks for the user's NetID and password appears. If the NetID and password entered match with the list of allowed users then the site asks them for the username and password they wish to use to create their account. If it does not match then the account cannot be created. Once the account exists, users do not need to enter their NetID and passwords again. They can directly click on the "Login" tab and enter the username and password of the account they created.

Once the website has been entered there are seven dropdown menus to choose from. The first step is to create a group by selecting "Edit/Create" from the dropdown menu under "Edit Group". A list of all cleanroom users appears with check boxes next to every name. Once the group members have been created a name for the group must be entered. All groups created by the same user appear under the "Groups" dropdown menu. This tab is also used to edit an existing group. If a group needs to be deleted, there is another option under "Edit Group". When "Delete Group" is selected, only the groups that belong to the user appear. All groups can only be edited or deleted by the creator.

When a group is selected there are two options. The user must choose between a pie chart and a scatter plot. If the pie chart is chosen no dates need to be entered because it only displays data for the past seven days. The default time period for the scatter plot is also for the past seven days but that can be edited by the user. Other graphs are available under the "Graphs" drop-down menu. The two options are "Machines" and "By User" which plot all the users of a specified machine over a specified time period or all the machines used by a labmember over a specified period of time. The default time period is over the past seven days.

If the user wants a list of the lab users that have spent the most time in the cleanroom then he or she should click on "Top Users". This option shows a table with the ten all time top users, top users for the past week, and top users of the day.

The “Machines” drop-down section has two options. A list of machines that are not currently operational can be displayed by clicking on “Machines Down”, and a machine can be chosen its reservations over any period of time.

The “Current Users” tab goes back to the default display which is a list of users currently inside the cleanroom. The “Help” tab displays a short guide with tips on how to use the website.

The Android and iPhone apps provide quick and easy connections to viewing information stored in the NDNFinfo and Coral databases. Each app has a login screen (where the username and password created on the site is entered) and a homepage with several buttons that lead to different activities.

3.4 Expectations/Results:

The system met all of our expectations. If the main program loop is running, the system actively and accurately shows who is in the cleanroom at any one time. The system can successfully read back-logged card swipe emails, and accurately update the database accordingly. If the program is turned off, then the loop is started back up again, and the database will be accurate and up-to-date.

The web site/graphing interface exceeded our expectations. A user can log in to the site, and create a group of cleanroom users to keep track of. This group can only be seen or edited by the person who created it. The plots are nice and colorful, and easy to read. If you want to single out certain members of the group without creating new groups, you simply click on the user’s name to toggle them on and off of the graph.

The Android app allows users to see the current users, top users, and machine statuses from their phone. It also provides information about machine reservations and personal session histories, as well as graphing capabilities for phones with extended JavaScript capabilities. An iPhone app provides similar capabilities, although it is not as polished as the Android app due to time limitations.

4. Detailed Project Description

4.1 System Theory of Operation:

The processor runs the Ubuntu V.10.10 ("Maverick Meerkat") operating system that was installed from a flash drive. The main program (/home/ndnfinfo/mail/Mail/new/main_program.cpp) collects emails as base64 encoded text files with fetchmail. These files are decoded with a base64 decoder. From each email, the important information is extracted and fed directly into the database code (/home/ndnfinfo/database_code.php). The database code organizes the information in the MySQL database. The website consists of separate PHP scripts on an Apache web server that queries the database for information when requested by the user, as well as JavaScript used to obtain the plots. Machine information is stored in a PSQL database, which is accessed via a read-only account. Similarly to the MySQL database, the machine data is pulled from the PSQL database as needed by the website, the main display, and the Android and iPhone applications. The Android application contains instantiation of HTTP objects that allow the program to connect to the database for any information necessary. Since the iPhone app is largely web-based, it is possible to embed PHP queries directly into the HTML for the display to obtain the information to show to the user.

4.2 System Block Diagram:

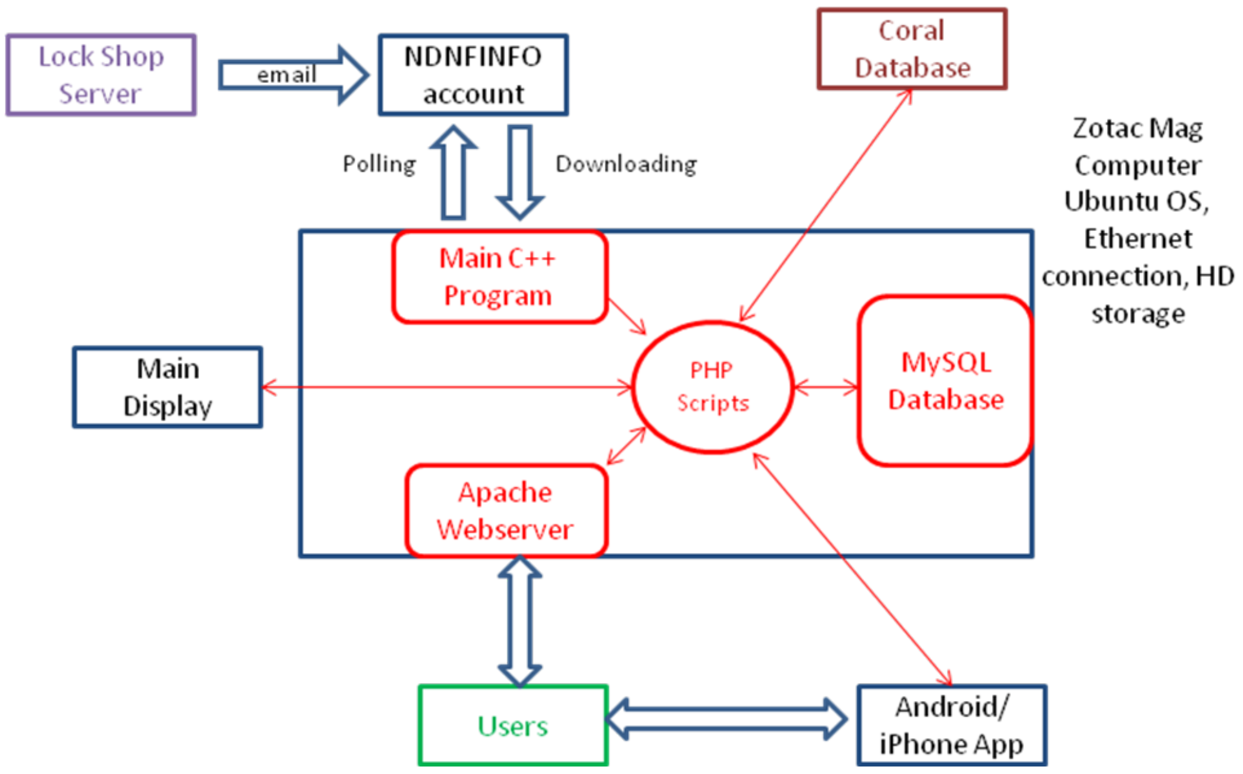


Figure 1. Block diagram of the system that shows which elements are inside the zotac, which are located externally and how they interact with each other.

4.3 Detailed Operation of Main Program:

The purpose of the main program is to continually check for new emails to ensure that the display and the website are up to date. It does this by periodically calling the email client fetchmail, which is its interface to the the ndnfinfo gmail account. Its interfaces to the MySQL database are the PHP scripts database_code and daychange, which is how the main program alters

The main program was written in C++. This language was selected due to its overall flexibility and the fact that it is easy to interface with the kernel via system commands in C++. This is useful, as several parts of the main program require using the system command function (executing fetchmail, ls, rm, opening and reading files, pausing the program operation, executing PHP scripts on the command line, etc.). The main tasks take place in the file main_program. Secondary functions necessary for functionality are included in the header file functions.h, and the functions for decoding base64 are included via base64decoder.h. The implementation of these header files is contained in their corresponding files functions.cpp and base64decoder.cpp. The program executes one large "while" loop to ensure that it continually checks for updates. First, the system command fetchmail runs the email client to download all new emails to the folder /home/ndnfinfo/mail/Mail/new. Then, the "ls" system command is piped into a file so that the main program has access to a list of these new emails. Each name is systemically read into the main program, and the file is opened to read its contents. The main program skips to the pause functions when there are no new emails to be read.

First, the date from the email is saved into the variables weekday, day, month, year, and time. Next, either "entering" or "leaving" is read into the string variable status. The rest of the email is encoded in base64, so it is then filtered through the function base64_decode from the header file before further filtering. The date is converted into the MySQL datetime format by converting the month to a number and adding dashes between the numbers (format: YYYY-MM-DD HH:MM:SS), and the name is filtered to remove newlines characters present from the email. The PHP script daychange is called to check to see if any users have been in the cleanroom for more than 24 hours. If they have, they are removed from the list of current users. This step ensures that even if someone swipes out, their name won't remain on the list for an extended period of time. The PHP script database_code updates the information in the database, and it is passed the arguments of the user's name, status, and the date and time of the swipe. Finally, the process pauses for five seconds, then repeats the cycle again.

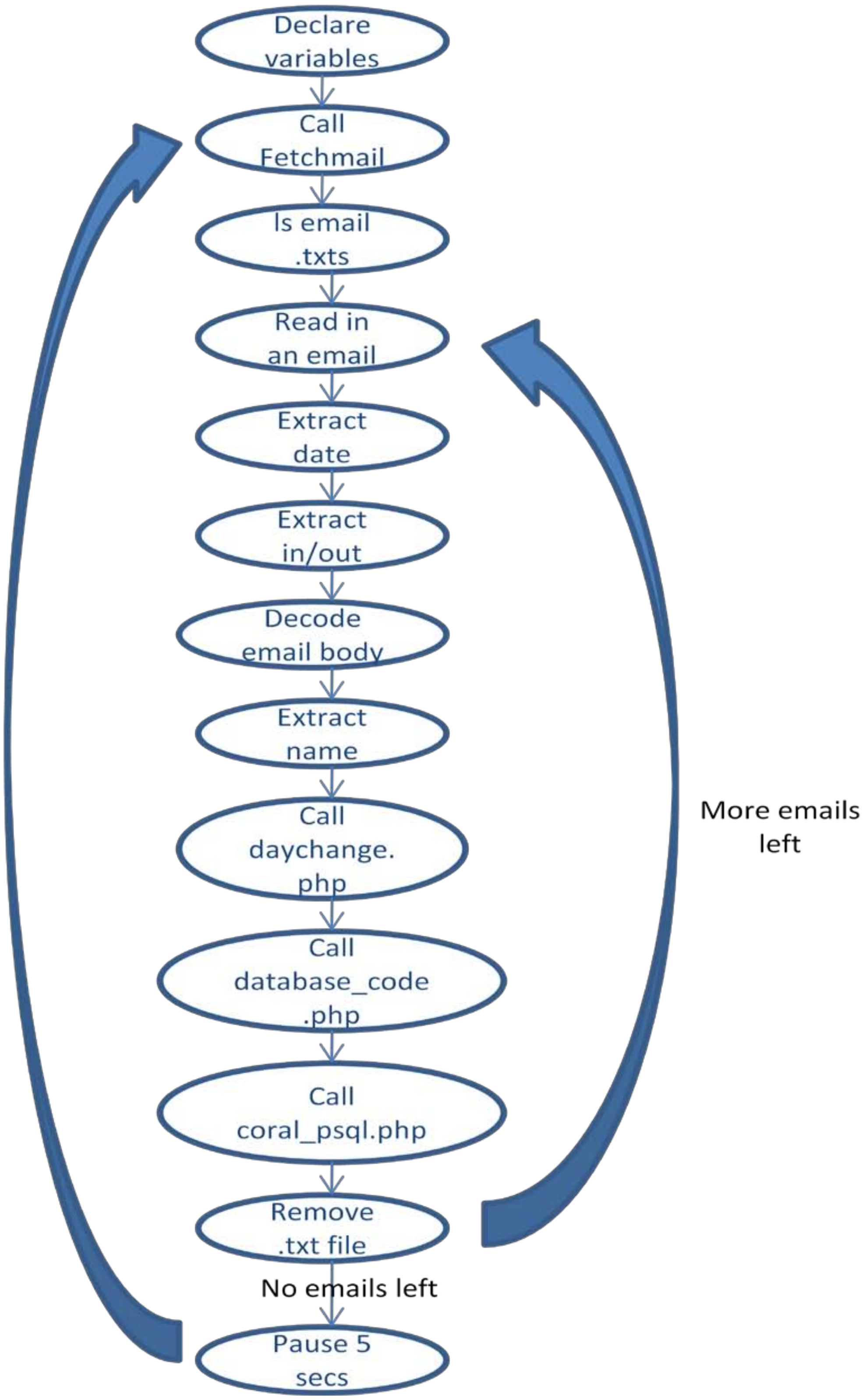


Figure 2. Flow chart that shows the components of the main program

4.4 Detailed Operation of Database:

The database serves as the record book for log information. All of the information must be stored in an easily accessible way so that it can be retrieved on request from the web server, Android/iPhone app, and display. Any time a log email is read, it is entered into the database. It also holds account information for the web site. Here is a breakdown of the tables in the database:

users – the table users has 2 columns (user and tin). The user column has the name of all users currently logged in to the cleanroom, and tin is a timestamp with the login time.

users2 – the table users2 has a row for every user who has swiped in or out since we started taking records. It has columns (user, time, day, days, total, super, and lastdate). The column “user” has the user’s name. “Time” has the user’s total time in the cleanroom (minus days*24 hours). “Day” holds the time spent in the cleanroom on the day of lastdate. “Days” is the number of complete 24 hour periods the user has spent in the cleanroom. “Total” is the combination of day and time which is displayed as total time on the website. “Super” is labeled “YES” if the user is allowed to be a super user, and “NO” if they can’t. “Lastdate” is the date of the last swipe.

UserName – There is a table like this for every individual user. This has a column for time in and a column for time out. This is a log of all sessions this user has had.

UserName_By_Day – There is a table like this for every individual user. This shows the sum total of time spent in the cleanroom each day organized by day. There is one column for the day and one column for the time.

Logins – This has the username and password of all accounts for the website.

Groups – This has the name of all groups and the owner of each group recorded.

Groupname – Each group has a table that has all members.

The Daychange.php program runs every time the date changes. It checks to make sure that there are no sessions that have been going for more than 24 hours. If someone started a session more than 24 hours ago at midnight, their session will expire, and will be removed from the records. In this case, total time in the cleanroom will not be correct, and the user will still be billed for the session. We remove it because the user is not actually in the cleanroom, and it might throw off rescuers in an emergency situation. If you would like to remove this fix, you can simply comment it out of the main loop where it is called.

The bulk of the database code is in a file called “database_code.php” in the /home/ndnfinfo directory. The code is called from the main program every time an email is processed. It is called with three additional command line arguments: The user’s name followed by a character (‘y’ means entering the cleanroom and ‘n’ means exiting) followed by the timestamp of the entry. The timestamp is in the form, ‘yyyy-mm-dd hh:mm:ss'.

The PHP script does basic handling of data variables, and automatically creates and organizes the database with MySQL queries. First, crosshatches and quotes are removed from the name variables to prevent unintentional code-commenting and termination. Since table names in MySQL cannot have spaces, a space-free version of the name is stored.

Since it is possible for a user to scan in without actually entering and scan out without actually exiting, we needed to be prepared for both of these conditions. We decided that when a person scans in twice in a row, they will be billed for the extra time, so the time will be added to their total time counter. Since they were not actually inside of the cleanroom, the session is not saved in the session log. When someone logs out twice in a row, no action is taken.

The PHP code compares the name with the users2 table to see if they are in the system yet. If they are not in the system yet, a row in the users2 table and the user's two personal record tables are created. The name is then compared to the users table which only has people currently in the cleanroom. If they are on the list, their name is removed from the table, and is replaced by the name with the updated log time.

The user's log table is queried to see if there is an unfinished log session. If there is one, the timestamp from the beginning of the session is recorded, and the session time is processed. When the session log is processed, a MySQL query returns the difference between the current swipe time and the beginning of the unfinished session. This difference is added to the total time counter. The date is extracted from both the current swipe time and the beginning of the session to see if they took place on the same day. If they took place on the same day, the time added to the total time counter is added to the current day counter. If they took place on different days, then the session is divided at midnight, and a variable is stored for the session time that took place on each day. The variable from the first day is added to that day's day log time, and sent to the "UserName_By_Day" table. The current day log counter is reset, and the variable from the second day of the session is inserted. Sessions that last longer than 2 days are removed using a different script from the main program before they make it to the main PHP script.

If there is no unfinished session in progress, then a session is created using the swipe timestamp as a begin time. If this date does not match up with the date of this user's last swipe, then the day column is sent to the "_By_Day" column, and the day column is reset. If the user is entering the cleanroom, then the session start time is recorded and they are left on the users table. If they are leaving the cleanroom, the unfinished session is terminated in the UserName table and the user is removed from the users table.

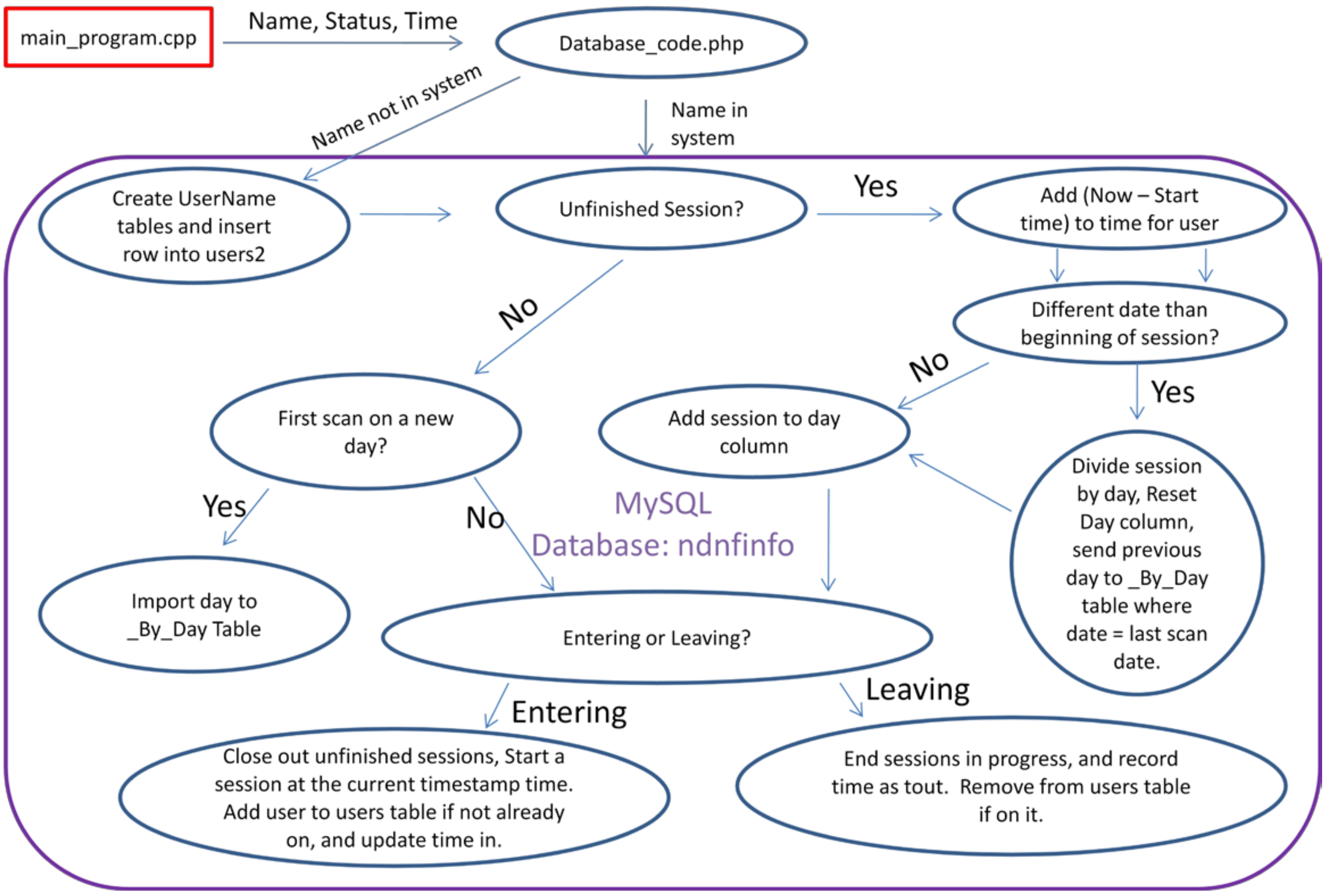

main_program.cpp
Name, Status, Time
Database_code.php
Name not in system
Name in system
Create UserName tables and insert row into users2
Unfinished Session?
Yes
Add (Now – Start time) to time for user
No
Different date than beginning of session?
No
Yes
First scan on a new day?
Add session to day column
Divide session by day, Reset Day column, send previous day to _By_Day table where date = last scan date.
MySQL Database: ndnfinfo
Yes
No
Import day to _By_Day Table
Entering or Leaving?
Entering
Leaving
Close out unfinished sessions, Start a session at the current timestamp time. Add user to users table if not already on, and update time in.
End sessions in progress, and record time as tout. Remove from users table if on it.

Figure 3. Block diagram of database.

4.5 Detailed Operation of Webserver:

The website was created with the purpose of displaying the information available on the database all in one place and organizing it in such a matter that it is easy to access. The database information is displayed mainly in two ways: by using graphs and tables. Tables are used to display information such as top users, reservations, and users currently inside the cleanroom. The graphs display information of a specific user, machine, or group members over a period of time specified by the user. The graphs are interactive making it easy for the user to customize them. The website is meant to be a user friendly interface to the database.

The webserver main directory (/var/www/) contains information needed for the operation of the main display, including PHP scripts for accessing MySQL database information, HTML and JavaScript files for the display layout and refreshing, and JPG images of sponsor information. PHP was selected as the language for many of the scripts due to the ease of querying MySQL and PSQL databases in PHP, as well as the fact that they are lightweight and can be executed “on the fly”. This provides our system with the flexibility to make updates only when necessary. For example, the server can respond to a user’s request to view a certain graph by executing a PHP script to query the database and making the information available. It is not necessary to have programs that are constantly running updating the graph and chart files for the website since the PHP scripts can do this on demand.

JavaScript was selected as the language for the graphs due to interactivity and its ability to make charts that are more aesthetically appealing. For example, it was easy to make graphs that only display certain lines that are selected using the JavaScript Highcharts API, but this would have likely been more difficult in PHP or a different language. The plots that were generated using PHP were not nearly as visually appealing as the ones found using the JavaScript APIs. As an added bonus, the JavaScript runs in the client’s browser (unlike PHP), which removes some of the burden from our server.

There are several subdirectories which contain the files for other parts of the system: android, iPhone, graphing, passwords, and site. The Android directory contains PHP scripts which access the MySQL and PSQL databases and JSON encode the output so that the app can interpret and parse the information. The iPhone directory also contains PHP scripts for similar purposes. However, it also contains several HTML files, as some of the iPhone app is designed as a web application due to time limitations. The Graphing directory contains files necessary for implementing the graphs on the website, the main display, and the apps. This includes PHP scripts for database access and HTML/JavaScript files for the display. Within the Graphs directory, the subdirectories “examples”, “exporting server”, “graphics” and “js” contain files JavaScript files used by the Highcharts utility for producing the graphs and plots. The Passwords directory contains PHP files used to verify users identity by checking the database for matches. Finally, the Site directory contains many PHP, HTML, and JavaScript files necessary for the functionality of our website.

Sometimes it is necessary to pass information between files. For example, if the user is asked to enter information then that information needs to be passed to the correct file that uses that information to send queries to the database. This is done by creating a form in html. By using a form variables can be passed between scripts through the GET method or the POST method. If GET is used then the variable is passed through the URL and the information is available to everyone. The POST method only works when the value being passed comes from an input by the user, so POST was used to pass all inputs like dates and names of machines. Sometimes however, a variable already existing in the script needed to be passed; this was done through the GET method.

The website provides extra information such as machine reservations and customizable graphs. The enterSite.php file creates the part of the website that allows users to create new accounts and to log in. The default information displayed on the iframe is the content of Users1.php which lists all users currently inside the cleanroom.

In order to create a new account, the users NetID and password are needed to make sure that only cleanroom user can create new accounts. A .htaccess file is in NDNFinfo's webfile space. This file maks a pop-up windo automatically ask the users for their username and passwords. If they are correct and on the list of allowed lab users then the user is directed to a page where he or she is asked to enter a username and password for their new account for this website. This is done on the saveAccount.php file. It checks if the username that was entered has already been chosen by somebody else, if it has then it asks the user to choose a different username, otherwise it creates the account and ads the username and password to the Logins table and creates a table for the user.

Once an account has been created, the user can access the site by clicking on "Login". The main.php file asks the user for their username and password and checks it against the usernames and passwords on the database. If they do not match then they are asked to try again, if they do then the contents of the site are displayed.

Once inside the site there are seven drop-down menus to choose from. The first tab, "Groups" shows all the groups that belong to the user that is logged in. If no groups have been created, nothing will happen. The groups.php file takes care of accessing the database and selecting the table of the user in order to get the names of all the groups and display them on the drop-down menu. The user is asked to input the dates for which to plot and to choose what kind of plot to display. The same file groups.php prepares the groupPieChar.csv file that contains the information to create the pie chart and redirects the user to groupPie.htm if the pie chart was selected. If a scatter plot was selected then the inputs are forwarded to Custom_Plots.php which creates the Custom_plot.csv file for the Custom_plot.htm. The csv file contains all the information and the htm file creates the plot.

The "Edit Group" drop-down menu is to add, edit, or delete groups. "Edit/Create" uses the createNew.php file to create or edit a new group. This file access the database and displays all users with checkboxes next to their names. All names selected as well as the name of the group are sent to createGroup.php which adds the name of the group to the Logged in user's table and also creates a new table for the group. When checking the database, if a group with the chosen name already exists and it belongs to somebody else, then the user is asked to choose a different name. If it belongs to the user, they are asked if they would like to overwrite the existing group. The answer is sent to edit.php, if they enter 'y' the new group is saved, if they enter 'n' then nothing is done and the old group remains intact. "Delete" uses the delete.php file to pull the names of groups that belong to the user from the database. The names of the groups are displayed so that the user can select which groups to delete. The selection is sent to the del.php file which accesses the database and removes the right tables.

To create graphs, either “Machines” or “By User” must be chosen from the “Graphs” drop-down menu. “Machines” uses the machine.php file which asks the user to select a machine and a begin date and end date. The inputs are sent to machines3.php which access the database to obtain the needed information based on the inputs from the user and creates the mach_data_by_user.csv file that the readcsv3.htm file uses to create the graph. “By User” uses the byUser.php file to ask the user to select the NetID of a lab member and dates for the plot. The information is sent to machies2.php which pulls the information from the database and writes it to user_data_by_mach.csv, readcsv2.htm access the csv file and crates the graph specified by the user.

The allTime.php file creates a list of the all time, daily and top users for the past week. This information is displayed in the form of a table. Under the “Machines” drop-down menu, if “Reservations” is chosen then a lit of all machines appears. Once a machine is selected, showRes.php asks the user for begin date and end date. The name of the machine and the selectd dates are sent to reservations.php, this file acces the PSQL database and obtains the reservations over the date specified for the machine selected. If “Machines Down” is selected, then machine_statuses.php will access the PSQL database and show a list of all the machines that are currently not operational.

The “Current Users” tab uses the Users1.php file to show a list of people that are currently inside the cleanoom. The last option is the “Help” button which uses help.php to display a short guide with tips and suggestions on how to use the website. The figure can be helpful in understanding the organization of the files that compose the website.

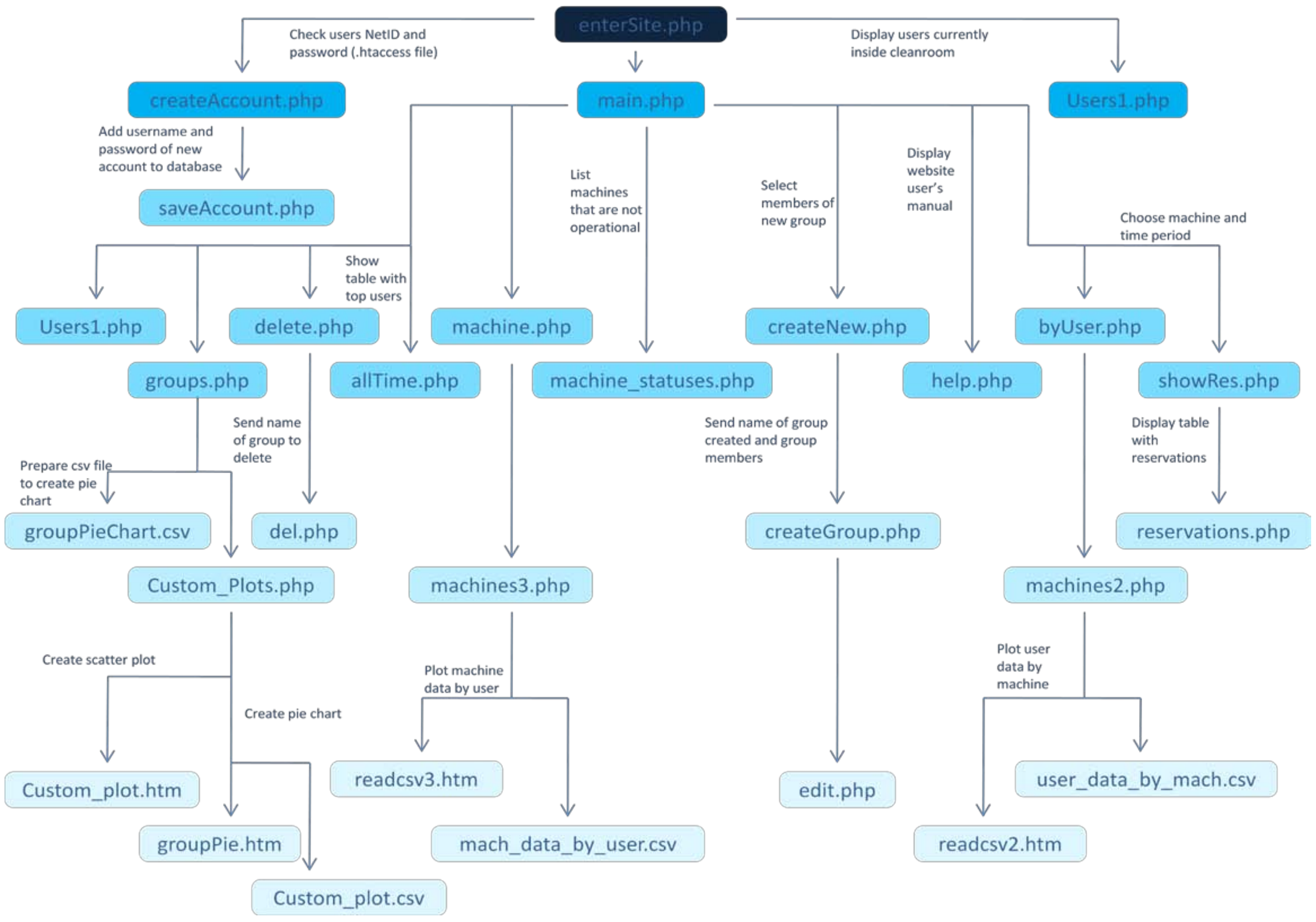

enterSite.php
Check users NetID and password (.htaccess file)
Display users currently inside cleanroom
createAccount.php
main.php
Users1.php
Add username and password of new account to database
saveAccount.php
List machines that are not operational
Select members of new group
Display website user's manual
Choose machine and time period
Show table with top users
Users1.php
delete.php
machine.php
createNew.php
byUser.php
groups.php
allTime.php
machine_statuses.php
help.php
showRes.php
Send name of group to delete
Send name of group created and group members
Display table with reservations
Prepare csv file to create pie chart
groupPieChart.csv
del.php
createGroup.php
reservations.php
Custom_Plots.php
machines3.php
machines2.php
Create scatter plot
Plot machine data by user
Plot user data by machine
Create pie chart
Custom_plot.htm
readcsv3.htm
edit.php
user_data_by_mach.csv
groupPie.htm
mach_data_by_user.csv
readcsv2.htm
Custom_plot.csv

Figure 4. Block diagram of the website. It shows all the components of the website. There are five levels. Each level down represents a file calling another file to carry out a part of the code

4.6 Detailed Operation of Graphing:

The purpose of the graphing is to provide the user with a way to visualize the data available pertaining to users' time spent in the lab and time spent working on machines. This gives advisors and graduate students an easy way to compare hours logged in the lab and to see who is to top user for a given machines. The interactivity of the graphs allows users to select and deselect individual members of a plot, and the graph is automatically resized. The main interface between the Graphs and the other subsystems are the PHP scripts that run queries on the databases and return the relevant information for the plot in the form of a csv file. The JavaScript then reads in the CSV file and creates the plot.

The graphing was done using an open source program from Highcharts.com. A combination of custom JavaScript code and functions from the Highcharts API were used. Several types of charts and graphs are present in the final version of the project. On the website, after creating a group, the user can select to display a scatter plot, which shows how many hours each member of the group was in the cleanroom on the days selected. Likewise, a pie chart on the website shows what percentage of the total group hours in the cleanroom each user has spent over the past seven days. Finally, the website has the capability to plot bar graphs for machines in the lab. The user can select to show how many hours a given user has spent on each machine, or how many hours several users has spent working on a given machine. All the files that create the grpahs are in the var/www/graphs/ directory. When either of the submenus "Machines" or "By User" are selected, the user is asked to choose a NetID/machine and specify a period of time. All these specifications made by the user are passed to the corresponding PHP files (groups.php/Custom_Plots.php) through a form using the post method so that the database can be queried for the right information.

The main display features a line graph with data points that show how many users there are in the cleanroom at the time of sampling. The script currentUsers.php runs a query on the MySQL database to return a list of users that are currently in the cleanroom. The number of elements in this list is summed, and it is passed with the current UNIX timestamp as a a data point. The file currentUsers.htm is the webpage on which the graph of the the number of users in the lab versus time is plotted. First, a request data function is run which requests a data point from the PHP script. This point is added to the current series, which is plotted on the chart. The JavaScript function setTimeout() determines how often this requestData() function is called, so it sets the time interval between points on the graph. Also, the value of the shift object sets how many points are to be plotted before cycling points off the screen. Finally, the end of the file adds the title and x-axis and y-axis lables, as well as creating a container for the plot.

The pie chart on the website shows the total hours spent in the lab by group broken down by each user's percentage. The file groups.php receives as input the name of the group to be plotted after the user selects it from the main menu. Then it queries the database to get a list of daily hours for each lab user over the past 7 days. These values are summed, and it is determined what percentage of the total time each user's time comprises. Then it generates a CSV file named groupPieChart.csv which contains each group member's name as the first entry on the line, and their percentage of total time as the second number. The other file, groupPie.htm, is the webpage on which the graph is displayed. It creates a chart object with the desired characteristics set. It opens the csv file and reads it in line by line, knowing that each line represents a chart slice. It pushes each slice onto the chart object, and finally, the chart is added to the container to be displayed.

The scatter plot shows how much time each member of the group spent in the cleanroom every day of the period specified. The user chooses a begin date and an end date. These two dates are used to send queries to the psql database requesting the amount of time worked during that period of time, Custom_Plots.php is the file that contains all the queries. This is done for every member of the group. Once all the times for each day have been requested, they are written into Custom_plot.csv. If the user chose the scatter plot option then Custom_Plots.php redirects the user to Custom_plot.htm. The htm file requests the data from the csv file and uses it to create the scatter plot.

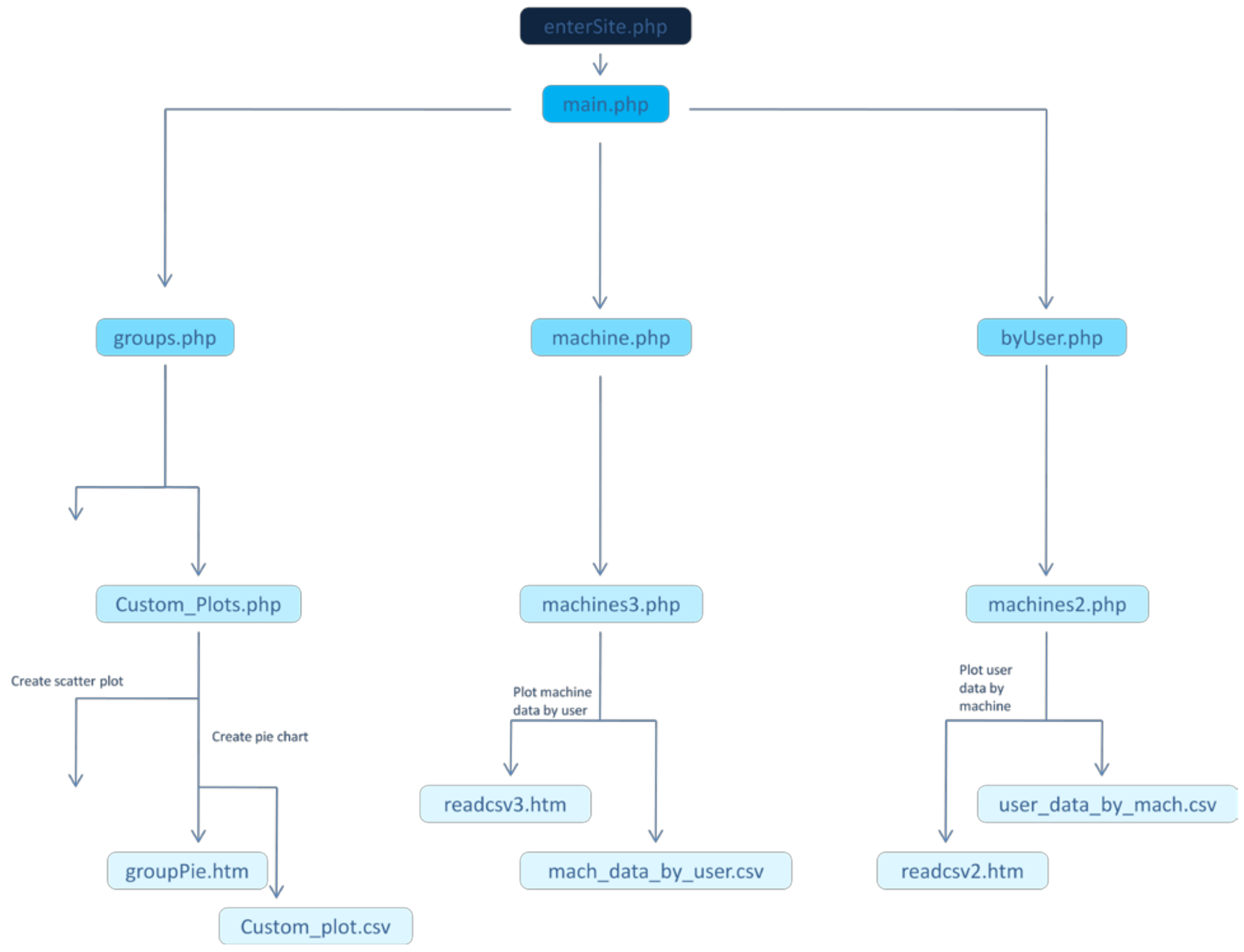


Figure 5. Plotting flow chart (subsection of website flow chart).

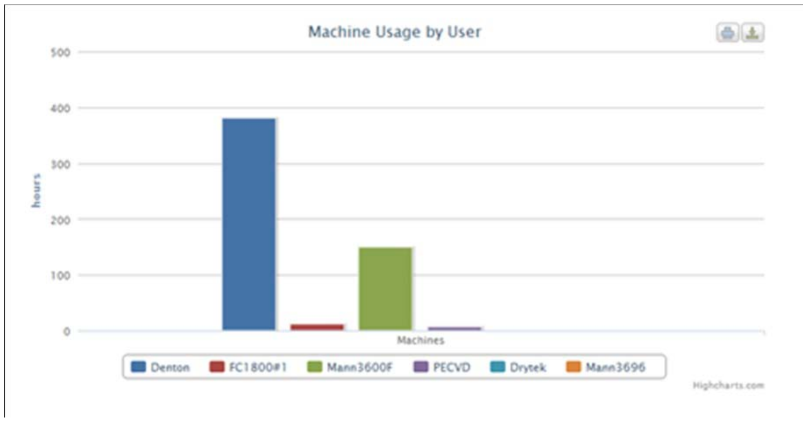


Figure 6a. Bar Graph of Machines used by a User

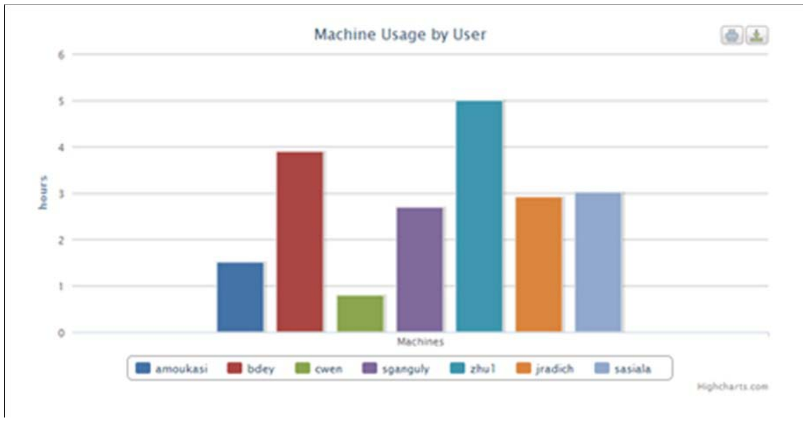


Figure 6b. Bar Graph of Users for a Machine

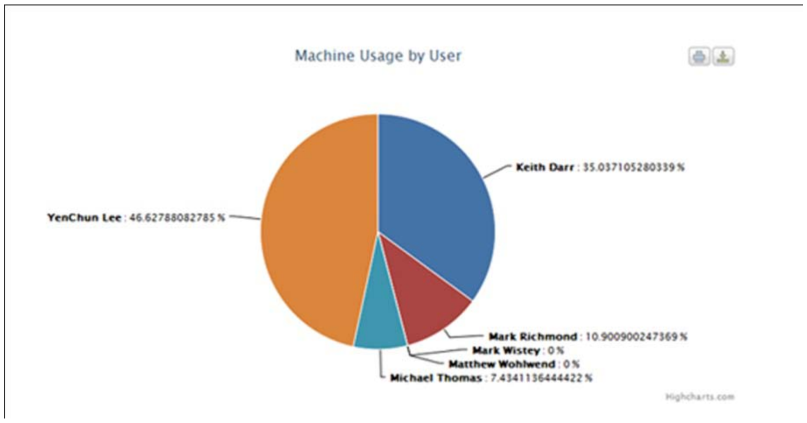


Figure 6c. Pie Chart for Groups

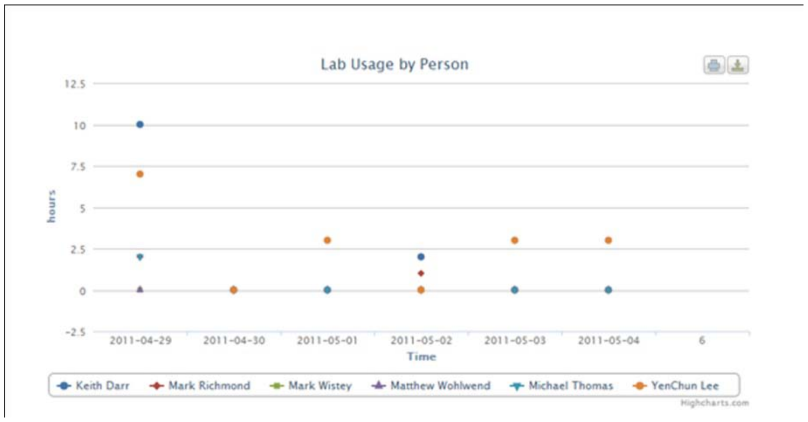


Figure 6d. Scatter Chart for Groups

4.7 Detailed Operation of Android App:

The purpose of the Android app is to provide users on the go with an easy way to check relevant information to cleanroom activity. This can help save time and possibly money, in the case where the user has forgotten to swipe out. The main interface the app has to other subsystems is the JSON encoded information provided from the database by PHP scripts.

The Android project main folder contains several subfolders that contain the files necessary for operation. The src folder contains the Java source files, the gen folder contains files generated by the Eclipse SDK, the res/drawable folders contain images used in the app, the res/layout folder contains xml files used for the various display layouts, the res/values contains String values used in the app, and the AndroidManifest file is an xml file that contains version numbers and permissions.

In the source folder, Main.java provides the startup functionality for the app (login screen information). This contains text input lines for the username and password instantiated from the file startup.xml. When the user clicks “submit”, these strings are taken in from the input lines and saved as variables. The program queries the database to obtain a list of registered users, and each username-password combination is tested to see if it matches the one supplied. If there is a match, Mainpage.java launches to display the homepage. This page contains buttons that list various functions that can be implemented, which are instantiated from the corresponding xml file. Each button contains an on-click listener that fires when a button is pressed. Depending on which button this is, an intent is created that can hold the new activity that is going to be launched. The first option is for rankings, current users, and machines down. Connect.java hosts each of these three tabs. A database query is executed by building an HTTP object in code with the proper URL and then calling the connect methods associated with this object. A JSON object is instantiated to hold the information from the query. This object can be parsed to remove the items of interest. A sample of JSON encoded output is:

```
[{“user”,”mschuele”},{“user”,”braynal”}]
```

In this case, the user is someone currently working in the lab, as the scipt was designed to return a list of all the current users in the lab. Curly brackets surround each data point, and the quotation marks separate each category or value. Commas separate cateogry-value pairs, as well as data points. Regular brackets enclose the entire data section. Users.java displays the current users in the lab, machines.java shows the current machines that are down, and rankings.java shows the top users for the day. Each of these files contains code that accesses the JSON encoded information provided by the PHP scripts on the Apache webserver on the Zotac, decodes it, and displays it on the phone screen.

The second option is to display the machine reservations. The activity reservations.java first intercepts the request to determine more information that is needed (i.e. the name of the machine, the start date, and the end date). The name of the machine is entered into a line that has autocomplete capabilities, to ensure that the machine name is entered exactly the same way that it appears in the database. A query to obtain the names of all the machines listed in the database is run to populate this field. The start date and end date are entered via calendar pop-ups that appear when the corresponding buttons are pressed. This information is passed between activities using the Intent.putExtra() function when calling the new activity. Finally, after the submit button is pressed, the activity showres.java accesses the server, decodes the information, and displays it on the screen.

The third option is to display the history of user login and logouts of the cleanroom. This can help a user to see how often they have been in the cleanroom between two dates, or quickly check what time they left the lab earlier in the day. The activity history.java is similar to reservations.java in that the start date and end date are entered using the calendar objects. The user name is entered into the line, which also contains autocomplete functionality. This is populated via a MySQL query to the database. Showhis.java is used to access the database and display information as before.

Finally, clicking the graphs button from the main page launches graphs.java, which displays a plot showing machine usage. This is accomplished via a simple WebView object. However, this part of the application might not be available on the user’s phone; it depends on the level that JavaScript is enabled on the user’s phone.

The images included that are used in the app are appbackground.jpg, apploginbackground.jpg, group.jpg, machine.jpg, and ranking.jpg. The latter three jpgs are used as thumbnail headers for the tabs in the tabview described earlier. The xml files included in the layout folder correspond to each major activity, using the setContentView function. Startup.xml is linked to Main.java , homepage.xml is used by mainpage.java, main.xml is linked to connect.java, date_request.xml is linked to reservations.java, history.xml corresponds to history.java, and webview.xml is linked to graphs.java.

A link for downloading the .apk for the Android app is available on the homepage for the NDNFinfo website. In order to download it, the option to allow non-market apps must first be set on the phone. The iPhone app is currently unavailable, since it is necessary to publish the app on the Apple appstore before users can download it. However, we currently do not have an account to allow us to publish our apps, so the iPhone app may be unavailable in the final release.

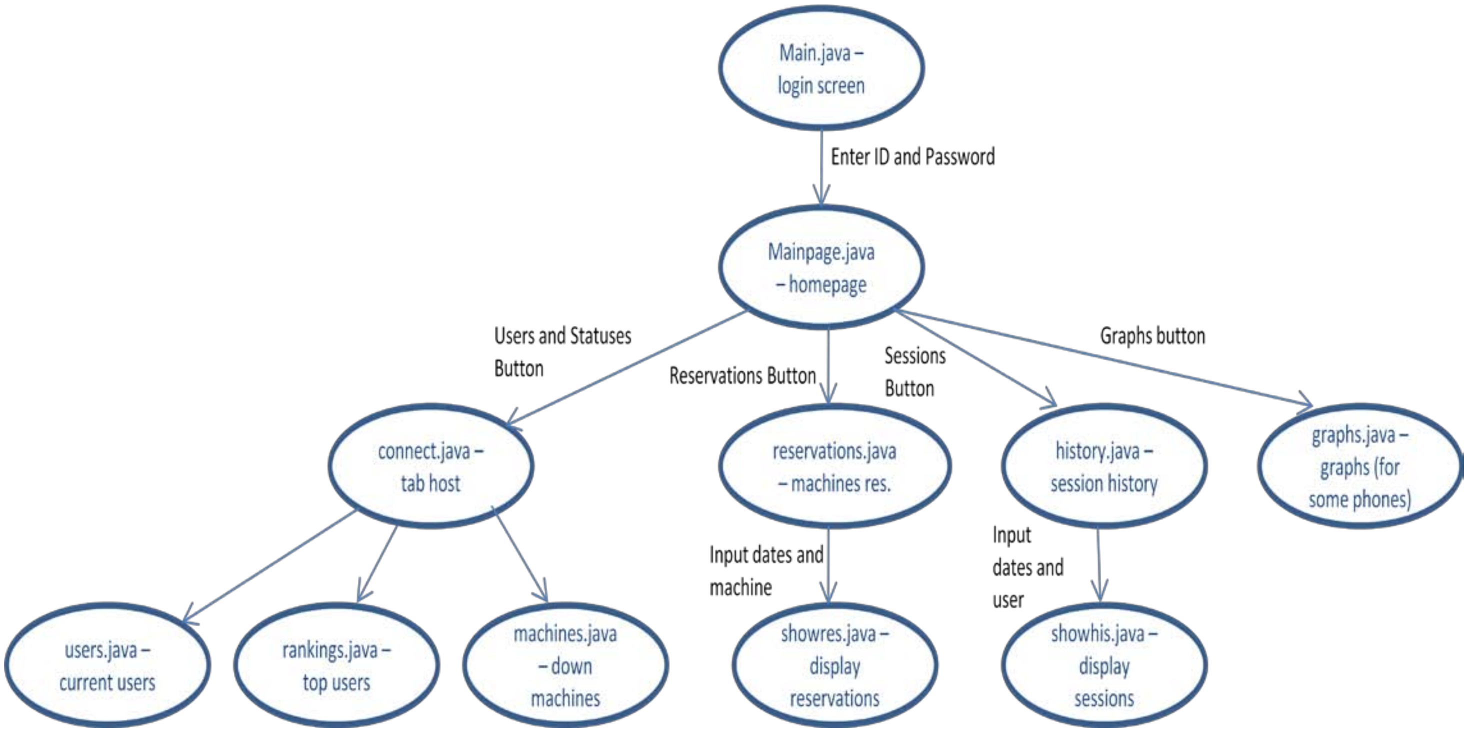
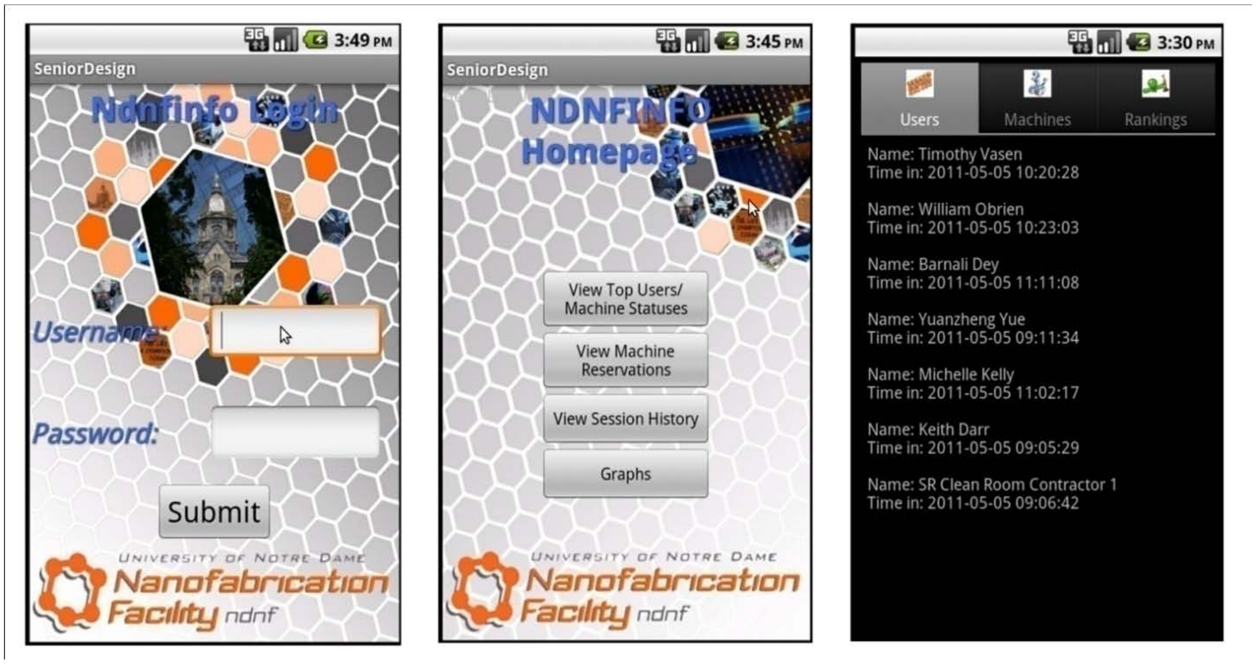

Main.java – login screen
Enter ID and Password
Mainpage.java – homepage
Users and Statuses Button
Reservations Button
Sessions Button
Graphs button
connect.java – tab host
reservations.java – machines res.
history.java – session history
graphs.java – graphs (for some phones)
Input dates and machine
Input dates and user
users.java – current users
rankings.java – top users
machines.java – down machines
showres.java – display reservations
showhis.java – display sessions

Figure 7. Block diagram of the Android App.



Name: Timothy Vasen
Time in: 2011-05-05 10:20:28

Name: William Obrien
Time in: 2011-05-05 10:23:03

Name: Barnali Dey
Time in: 2011-05-05 11:11:08

Name: Yuanzheng Yue
Time in: 2011-05-05 09:11:34

Name: Michelle Kelly
Time in: 2011-05-05 11:02:17

Name: Keith Darr
Time in: 2011-05-05 09:05:29

Name: SR Clean Room Contractor 1
Time in: 2011-05-05 09:06:42

Figure 8. Screen Shot of the Android App.

4.8 Detailed Operation of iPhone App:

The purpose of the iPhone app is to provide a suplement to the Android app for users that do not have an Android phone. The iPhone app provides much of the same functionality as the Android app for users that are not near a computer and want to quickly obtain cleanroom information. This app interfaces to other systems through PHP scripts, which query the MySQL and PSQL databases.

The initialization that runs on startup of the iPhone app is written in Objective C. This includes the Navigation Controller and main program with its header. However, since it is quicker and easier to write HTML, JavaScript, and PHP than Objective C due to our limited experience with this language and the lack of time available to learn it, we decided to implement most of the iOS application as a web app that is customized to fit on an iPhone screen. Therefore, it is preferable to use this app rather than just using the Safari browser because the login screen, mainpage, and various graph and chart views are designed so that they fit well on the iPhone screen, unlike for our webpage. Given more time, it would have been cleaner and more polished to write the entire app in the native language for the iPhone (as we did with the Android application), but given time and resource limitations, this was the best option available. In any case, the core functionality for the iPhone app is the same as for the Android app.

These HTML and PHP files are contained in the iPhone folder within the /var/www folder on the Zotac Mag. The index.html file contains the homepage for the app, which requires the user to enter a login id and password. These are read in using the getPost functions, and they are checked against the database information with the login.php script. The user is allowed to continue if they match up with an account found. The homepage is displayed with homepage.php. Several buttons are included on the homepage, created in the HTML portion of the file. The buttons are formatted to all be the same size using CSS formatting. The AppBackground.jpg provides the background image for the app. The first button that can be pressed is for current users, which launches users.php to display a table of the current lab users.

Another option lets the user see the top users for the day, which is implemented via topusers.php. Reservations.php and machres.php are used to let the users view the machine reservations over a two week span centered about the current day. Reservations.css includes HTML formatting that is used by this and several other files. Finally, several files are used to provide a plot of machine data that is sized to fit the iPhone screen. Machines.php first intercepts the users request and gets the relevant information from the user. Machines_by_user.php generates a csv file named mach_data_by_user.csv, which contains the information used in the plot. ReadCSV3.htm reads in this csv file, and generates the desired plot.

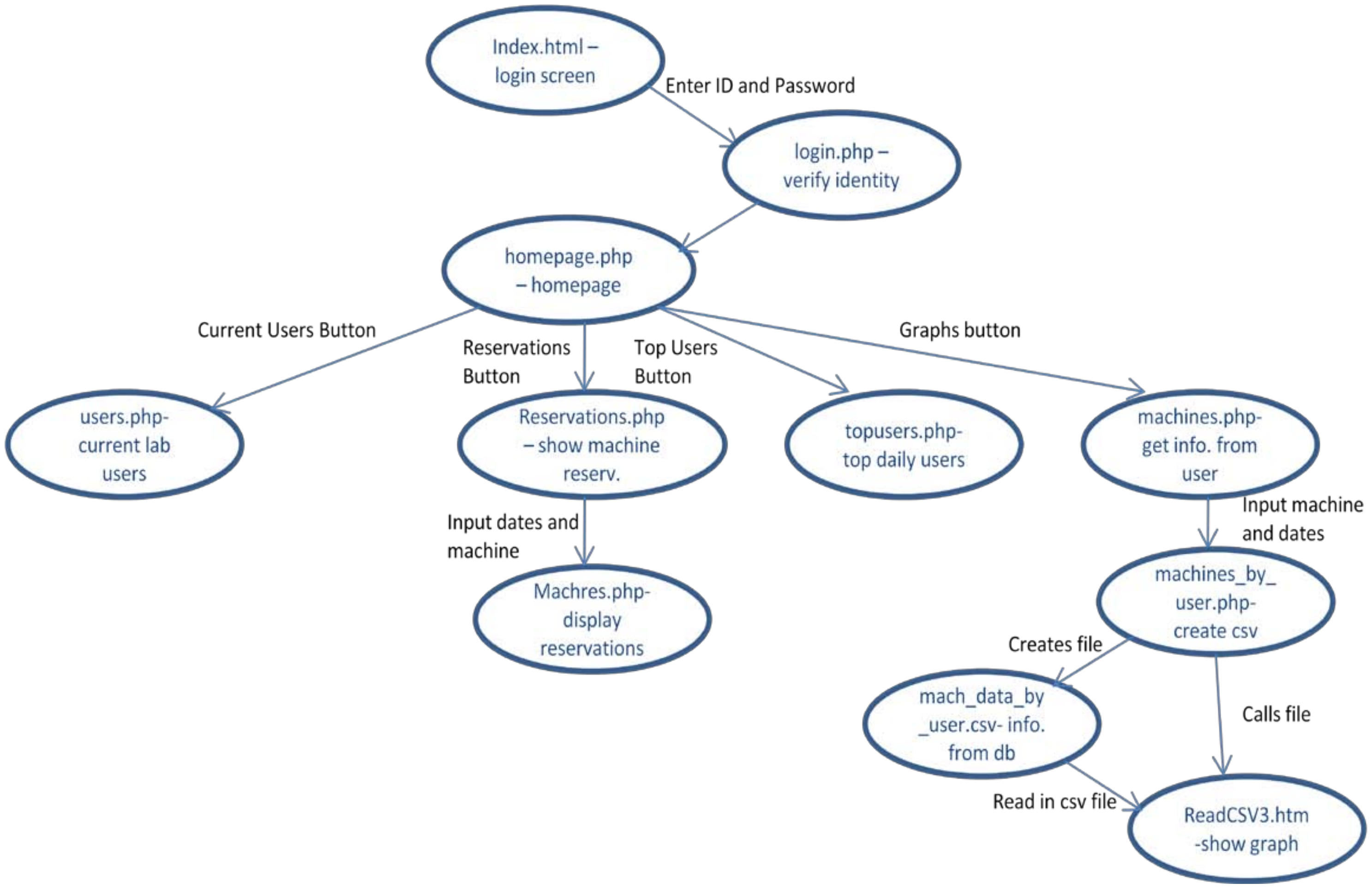

Index.html – login screen
Enter ID and Password
login.php – verify identity
homepage.php – homepage
Current Users Button
Reservations Button
Top Users Button
Graphs button
users.php- current lab users
Reservations.php – show machine reserv.
topusers.php- top daily users
machines.php- get info. from user
Input dates and machine
Input machine and dates
Machres.php- display reservations
machines_by_ user.php- create csv
Creates file
mach_data_by _user.csv- info. from db
Calls file
Read in csv file
ReadCSV3.htm -show graph

Figure 9. Block diagram of the iPhone App.

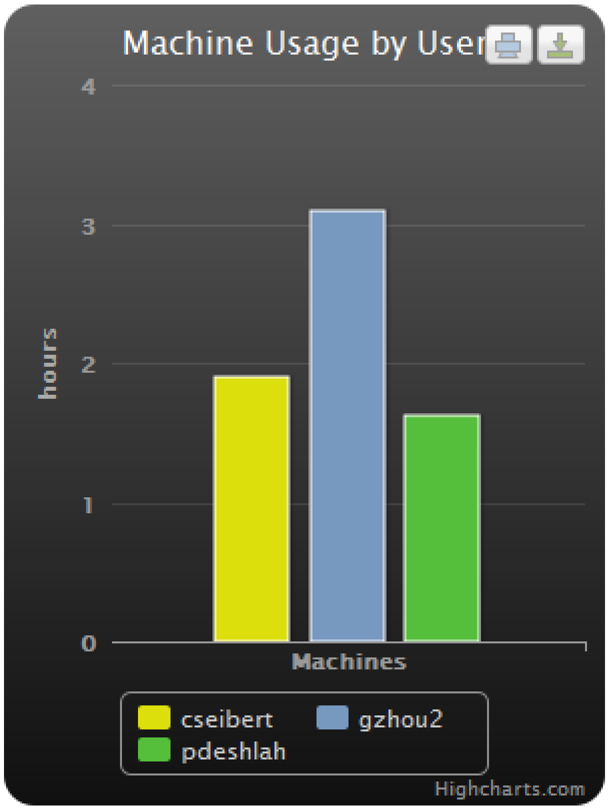
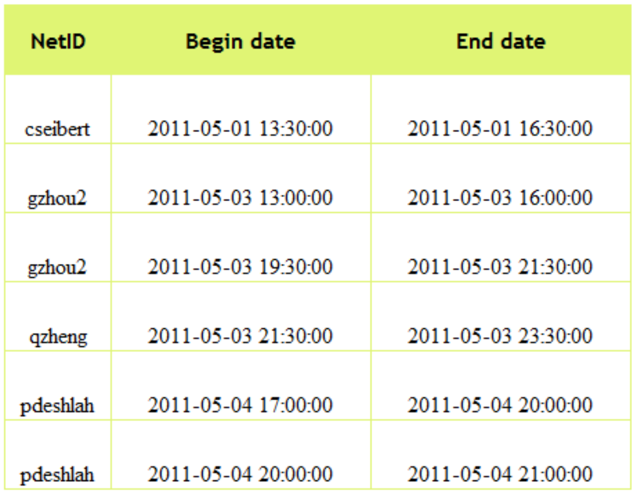


Figure 10a. Screen Shot of the iPhone App.



NetID	Begin date	End date
cseibert	2011-05-01 13:30:00	2011-05-01 16:30:00
gzhou2	2011-05-03 13:00:00	2011-05-03 16:00:00
gzhou2	2011-05-03 19:30:00	2011-05-03 21:30:00
qzheng	2011-05-03 21:30:00	2011-05-03 23:30:00
pdeshlah	2011-05-04 17:00:00	2011-05-04 20:00:00
pdeshlah	2011-05-04 20:00:00	2011-05-04 21:00:00

Figure 10b. Screen Shot of the iPhone App.

4.9 Detailed Operation of Display:

The display was the original requirement of the project. The display was to provide a convenient physical contact point so that others can quickly determine who else is using the facilities. There were other features originally on the display besides current users, super users, and usage hours- for example, machine statuses and who was using which machines. However, the client determined that a cleaner look would be easier to read and more aligned with the desired aesthetic. The display takes in data from the server and from the MySQL database.

The display code is in the index.php file, which is located on the webserver, and it calls UsersDisplay.php and Superusers.php. UsersDisplay.php queries the MySQL database to check who is currently in the clean room. Originally the display would scroll, using the marquee tag if there were too many users, but the client stated that he did not want the display to scroll, as it made the display too busy. Instead, depending on the number of users in the clean room, UsersDisplay.php will adjust the font size. This process is necessary; otherwise, there will be data overflow and the extra users will not be visible. Superusers.php queries the MySQL database and takes the non-faculty and non-contractor users who have been in the cleanroom the most in the last 7 days.

To display the graph that shows the usage hours, an Iframe was created that displays the HTML file: http://ndnfinfo.ee.nd.edu/graphs/currentUsers.htm. Index.php uses the JavaScript function reloadIframes to refresh the Iframes so that the data will be updated. The frequency that the Iframes refresh can be changed by altering the line starting with setTimeout; the number should be changed, noting that the unit is milliseconds. Sizing the display horizontally is not an issue, but the vertical spacing may need tweaking for different screen resolutions. The Display is implemented by using the Firefox add-on Fullscreen, available at https://addons.mozilla.org/en-US/firefox/addon/full-fullscreen/. This seems to be the best add-on because to enter and exit Kiosk mode, the user just has to press “F11.” The user needs to make sure to also go to the add-on preferences and disable the tab bar, so that it will not appear.

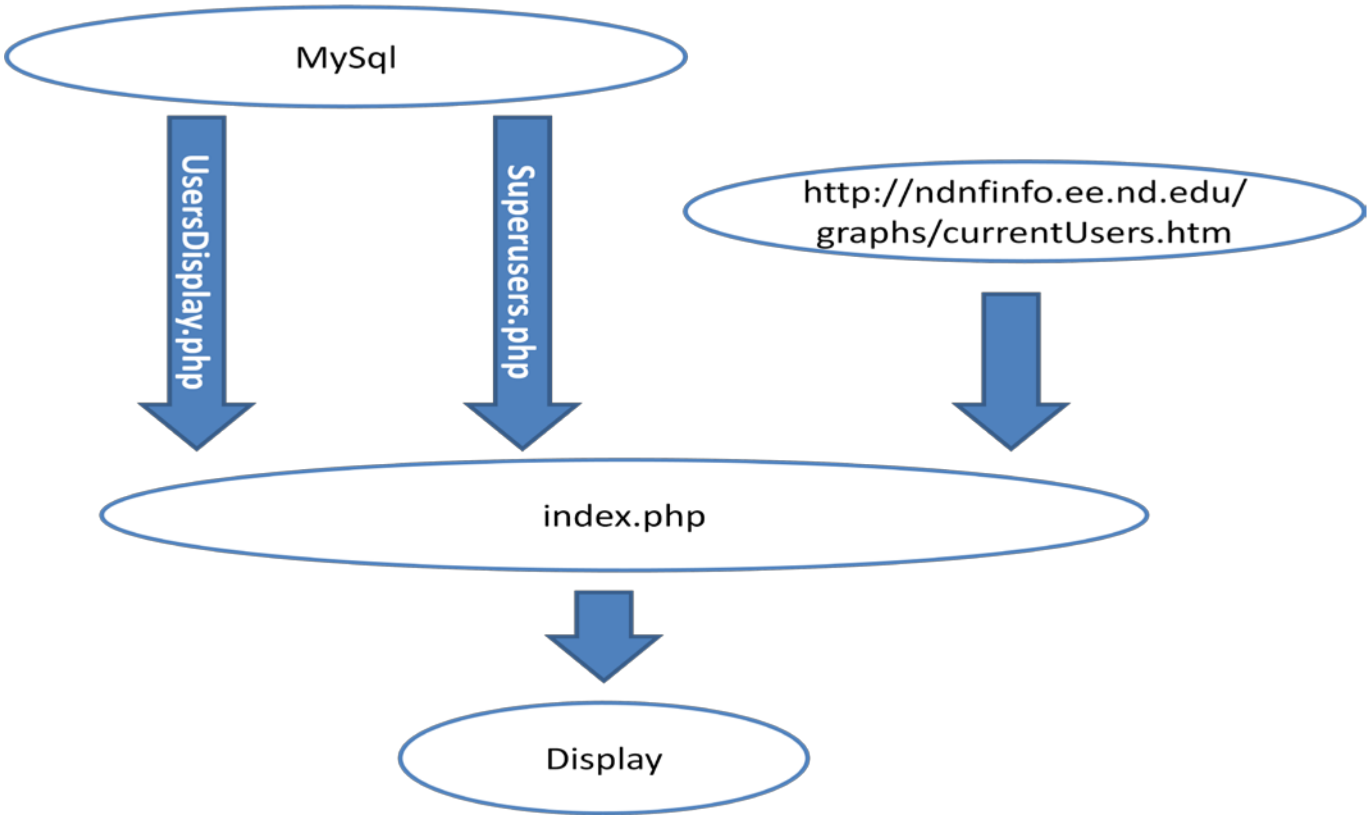
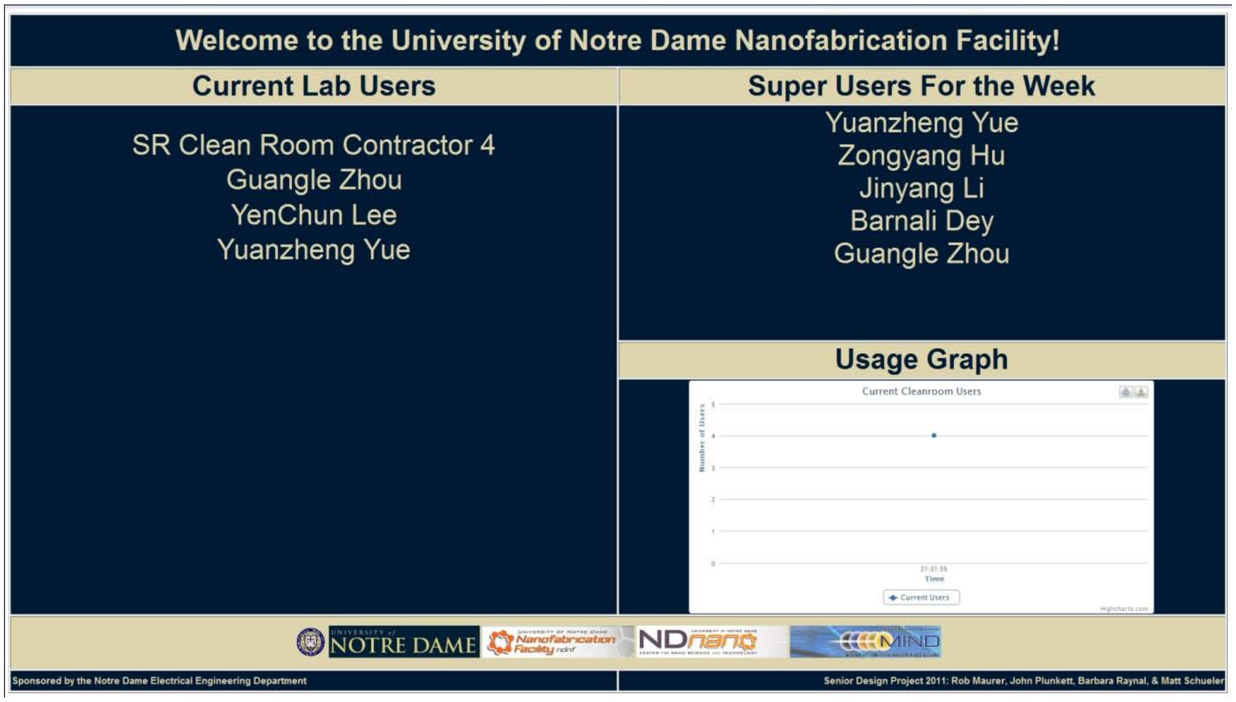


Figure 11. Block diagram of the main display.


Welcome to the University of Notre Dame Nanofabrication Facility!

Current Lab Users	Super Users For the Week
SR Clean Room Contractor 4 Guangle Zhou YenChun Lee Yuanzheng Yue	Yuanzheng Yue Zongyang Hu Jinyang Li Barnali Dey Guangle Zhou

Usage Graph

Sponsored by the Notre Dame Electrical Engineering Department

Senior Design Project 2011: Rob Maurer, John Plunkett, Barbara Raynal, & Matt Schueler

Figure 12. Screen Shot of the main display.

4.10 Interfaces and Sensors:

There are several interfaces in between subsystems in our system. First, there is the WiFi link between the Lenel server and the ndnfinfo email account on Notre Dame's server. There is also the Ethernet cable connecting the Zotac hardware to the internet, which gives the main program access to the internet so it can download emails. A VGA cable provides the link between the main system hardware and the LCD display. Additionally, WiFi provides the link between the Android and iPhone apps and the rest of the system.

In addition to these hardware links, there are many software links between subsystems that allow the system to function. The Apache webserver software running on the Zotac provides the basis for the website. The website is able to communicate with both the MySQL and PSQL databases with PHP scripts that allow embedded queries. The main program is able to access the Notre Dame server via the email client Fetchmail, and it is able to modify the MySQL database with several PHP scripts. Finally, the data encoding format JSON generated from PHP scripts allows the Android and iPhone apps to have access to information from the rest of our system.

5. System Integration Testing

5.1 Testing subsystems

Subsystem Testing Procedures: The subsystems were almost entirely software, so they were all tested using the traditional method of commenting out unnecessary functions while others are tested individually until the program runs cohesively.

Main Program: The main program was developed before any of the other parts of the system, so it was initially isolated from the database code and website, making it easier to debug. The portion of the program dealing with decoding base64 was tested on a sample email saved for this purpose. Initially, the main program maintained a list of current users in an arraylist that grew and shrunk as users entered and left the lab. This information was then sent to the terminal and eventually, to the website in the form of an html file. This is how the program was debugged- by reading the terminal ouput and verifying that it much the users currently in the lab. Later, however, it was determined that it was much simpler for the main program to not maintain a list of users and simply make any changes dictated by the emails to the database. This allowed the website, display, and apps to access the information without having to interface directly with the main program.

Database: The main PHP script was tested with manual individual entries from the command line of the terminal. Small pieces of the code were tested at first to make sure that the database was being manipulated in the correct way. When the code reached its first “complete” version, the unread mail was backed up, and then fed into the database through the main code. Getting this step to work was probably the most difficult milestone. This was where we figured out that we needed to remove crosshatches, periods, and quotes from names to avoid errors. After this step, additions were tested by attempting to build the database correctly from the ground-up using backed-up email logs.

5.2 How does it meet overall requirements

First and most importantly, the customer was extremely satisfied with the product. The functionality exceeds the original requirements, and is extremely user-friendly. The easy-to read display maximizes utility, while the website allows for highly customizable and in-depth reports and usage club. The smart-phone apps make it even easier to use on-the-go. Implementation of this product will increase efficiency of the lab, impress visitors, and provide a fun activity for users.

6. User's Manual/Admin's Manual

6.1 How to install the product

Installation manual assuming user has a blank Zotac Mag and all of our code. How to install NDNFinfo: The first thing the user must do is install an operating system on the device. We used Ubuntu v.10.10 due to the fact that it is free of cost, it has a wide array of free and open source software available, it comes packaged with a C++ compiler, and it is simple and easy to execute system commands. The most recent version of Ubuntu can be obtained from the website ubuntu.com, downloaded to a flash drive, and installed after booting to the flash drive. Next, several software packages must be installed on top of the OS in order to provide enough functionality for the system to run. Apache, Fetchmail, MySQL, and PHP can all be installed with the Advanced Packaging Tool provided with Ubuntu (sudo apt-get install *x*). The Firefox Kiosk Add-on allows the browser to enter into display mode for the main screen, hiding the URL bar and search bar. Additionally, a copy of all the source code that is necessary for operation is included with the project

The following plot shows all the necessary directories in the Linux file system, and files that each directory needs to contain. Directories that already exist or that should exist after installing the programs listed above (such as Apache or Fetchmail) are listed in black. Directories that must be created manually using the mkdir command are listed in red. A number with a circle around it indicates a table listed below, which contains the names of all the files that should be included in this directory.

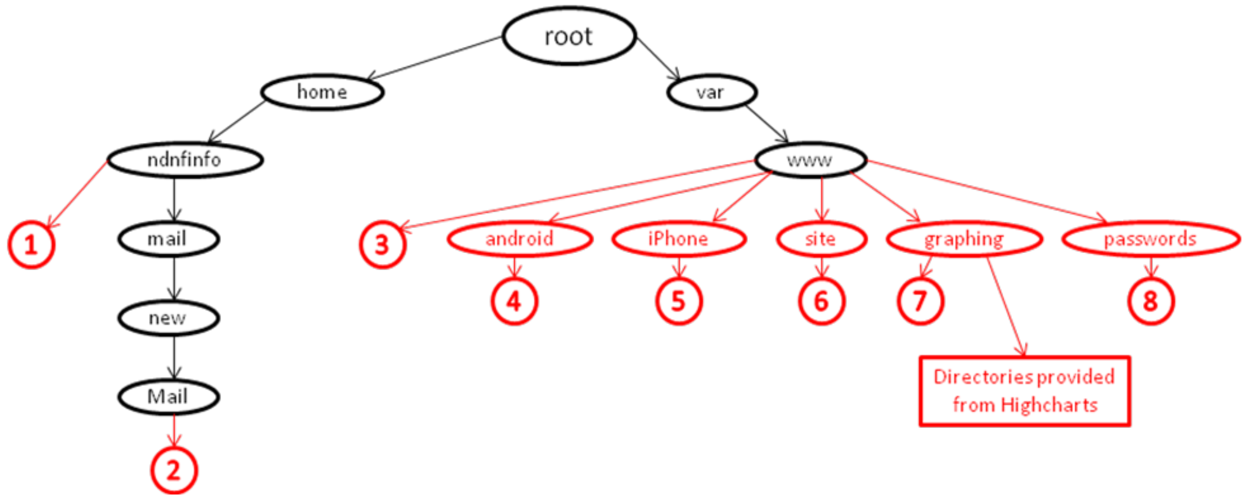


Figure 13. Flow Chart of Linux directories and files

1) Ndnfinfo
database_code.php
coral_psql.php
databasereset.php
daychange.php

2) Mail
ndnf (executable)

3) www
500px-NotreDameSeal.png
MindBanner.png
ndnanoBanner.png
ndnfBanner.png
UniversityBanner.png
allTime.php
index.html
index.php
index2.php
Machines_in_use.php
machine_statuses.php
Superusers.php
title.php
Users.php
UsersDisplay.php
UsersDisplay2.php

4) Android
machinenames.php
machine_statuses.php
machres.php
rankings.php
userhis.php
usernames.php
users.php

5) iPhone
AppBackground.jpg
homepage.png
index.htm

login.php
mach_data_by_user.csv
machines.php
machines_by_user.php
machres.php
readCSV3.htm
reservations.css
reservations.php
super.php
topusers.php
Users.php

6) Site
allTime.php
createAccount.php
createGroup.php
createNew.css
createNew.php
del.php
delete.php
disp.php
edit.php
enterSite.php
help.php
Machines_in_use.php
machine_statuses
machine_statuses.php
main.css
main.js
main.php
reservations.css
reservations.php
saveAccount.php
showRes.php
stdtheme.css
users.php
Users.php

7) Graphing
autocomplete.js
byUser.php
currentUser.htm

currentUsers.php
Custom_plot.csv
Custom_plot.htm
Custom_Plots.htm
Custom_Plots.php
groupPie.htm
groupPieChart.csv
groups.php
index.html
mach_data_by_user.csv
machine.php
machines2.php
machines3.php
readcsv2.htm
readcsv3.htm
user_data_by_mach.csv
Highcharts directories

8) Passwords
checkpassword.php

To initialize a blank database with the correct tables (empty), run the code "databasereset.php".
WARNING: Running databasereset.php will delete all records from the database. This file should only be run if you are trying to start clean.

6.2 How to set up product

The processor has USB ports for a keyboard and a mouse. Due to the nature of our static IP address, it must be hard connected with an Ethernet cable to the Notre Dame network. This is also necessary to access the Coral database. When all the files are in the correct directories and the database has been initialized, all you need to do is run the executable 'ndnf' in the directory '/home/ndnfinfo/mail/Mail/new'. This will fetch all user log emails from the mailbox and organize all data on the database ndnfinfo. While the main program loop is running, incoming data logs will be processed and added to the database. The display monitor should be set up at the appropriate location, and it should be displaying the address 'ndnfinfo.ee.nd.edu/index.php'. The display and web site do not work on Internet explorer. The display will only be accurate while the main program loop is running. When a new lab technician, contractor, or visitor card appears in the system, their "super" column in the table users2 will need to be changed to 'NO', indicating that they are not super user-eligible.

6.3 How the user can tell if it is working

The display will accurately show who is in the lab, and the web site will be functional. If a user has been in the lab for more than 24 hours, their name should be removed from the display. Website users should be able to login using accounts they have created, and use the plotting utilities provided. These utilities should provide the correct information, which can be verified using their knowledge of their own lab activity, or by reports generated from the billing. The Android and iPhone applications should provide login to registered users, and they should display the correct information upon request.

6.4 How the user can troubleshoot the product

If you need to access the MySQL database manually to edit information view information directly, enter into the command line “mysql -u root -p”. It will ask to enter password. The current password is “WafersD11”, but it can be changed. The current root password for Ubuntu is “WafersD11” as well, in case administrative privileges are required. If the web site goes down, you should first check the hard connection to the network before checking the programming. In the rare case you find a programming bug, it should not be too difficult to locate and fix the problem based on what goes wrong.

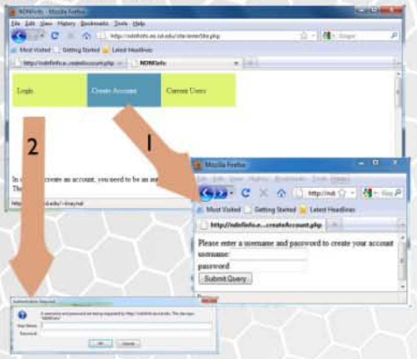
NDNF info
Users Help Guide

ndnfinfo.ee.nd.edu

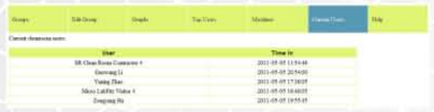
2010-2011
EE Senior Design Project

Robert Maurer
John Plunket
Barbara Raynal
Matthew Schueler

ndnfinfo.ee.nd.edu: This page shows you a list of current cleanroom users, a brief explanation for the site, and links to related pages. The link "Graphing Central" allows you to generate a graph for machine or cleanroom use.

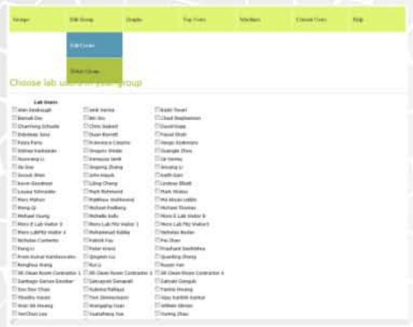


First Time Users: Click "Create Account" at the top of the graphing central main page. You will be asked to provide your netid and password. If you are an authorized user, you will be asked to create a user name and a password for the web site that you can use to log in from now on. Once you create an account, click "Login" at the top to use the page.



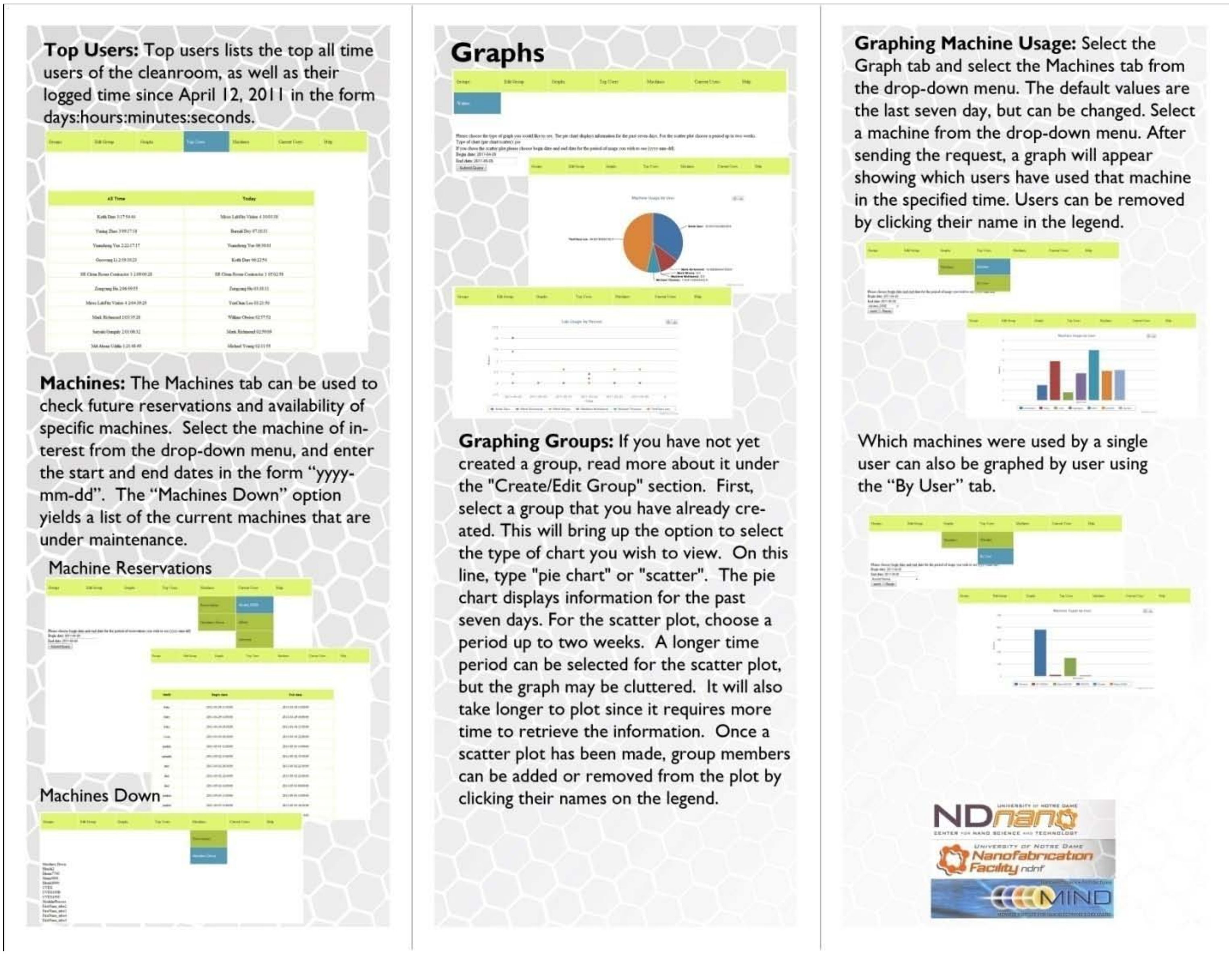
User	Time in
SR Clean Room Contractor 4	2011-05-05 11:54:46
Guowang Li	2011-05-05 20:54:00
Yuning Zhao	2011-05-05 17:16:05
Micro Lab/Fitz Visitor 4	2011-05-05 18:46:05
Zongyang Hu	2011-05-05 19:55:45

Login: Once you log in, you will see the tabs at the top change to graphing options. If you click current users on the far right, you will see a table that has the users that are currently in the cleanroom, and the times at which they logged in.



Create/Edit Group : If you would like to create a group to monitor, you can select "Edit Group" from the drop-down menu at the top of the page. Select "Edit/Create Group". This will lead you to a selection screen with all cleanroom users listed alphabetically. Check all of the users you would like in your group and pick a name. Group names must be limited to one word. If your group name is invalid, you will be asked to enter a different name.
Once a group is created, you (and only you) will be able to see and select your group from the "Groups" tab at the top left of the page. If you wish to edit an existing group, click the "Edit/Create Group" tab again, and check off the names of the users you wish to have in the new version of your group. Then put the name of your group in the text line at the bottom of the screen and submit your query. The changes you have made to your group will now be reflected in the plots. Finally, if you wish to delete a group, select the "Delete Group" tab. You will be prompted to select which group you wish to delete.

Figure 14a. User Help Guide Front Side



Top Users: Top users lists the top all time users of the cleanroom, as well as their logged time since April 12, 2011 in the form days:hours:minutes:seconds.

Machines: The Machines tab can be used to check future reservations and availability of specific machines. Select the machine of interest from the drop-down menu, and enter the start and end dates in the form "yyyy-mm-dd". The "Machines Down" option yields a list of the current machines that are under maintenance.

Machine Reservations

Machines Down

Graphs

Graphing Groups: If you have not yet created a group, read more about it under the "Create/Edit Group" section. First, select a group that you have already created. This will bring up the option to select the type of chart you wish to view. On this line, type "pie chart" or "scatter". The pie chart displays information for the past seven days. For the scatter plot, choose a period up to two weeks. A longer time period can be selected for the scatter plot, but the graph may be cluttered. It will also take longer to plot since it requires more time to retrieve the information. Once a scatter plot has been made, group members can be added or removed from the plot by clicking their names on the legend.

Graphing Machine Usage: Select the Graph tab and select the Machines tab from the drop-down menu. The default values are the last seven day, but can be changed. Select a machine from the drop-down menu. After sending the request, a graph will appear showing which users have used that machine in the specified time. Users can be removed by clicking their name in the legend.

Which machines were used by a single user can also be graphed by user using the "By User" tab.

Figure 14b. User Help Guide Back Side

7. To-Market Design Changes

To send this product to market, we would need to greatly improve system security and create automatically generated backup files with easy implementation. The Notre Dame-specific design templates for the display and the website would need to be modified into all-purpose customizable templates. Web usage reports should probably be restricted to advisors and administrators only, due to individual security issues. The machine-related functions will only be available to facilities with Coral a similar system installed. The code is all functional, but there is certainly room for improvement in terms of processing efficiency and code simplification. The product would also need a few months of in-depth testing before being sold commercially. It has only been finalized for a few weeks, and there are sure to be bugs that will surface over time.

Additionally, hardware failure is certain to occur over time, so redundancy and software backup would definitely need to be extended. This could involve having more than one processor in case one fails so that the display will not be down for an extended period of time. If the website is expected to have high traffic, especially in larger facilities, it would be preferable to have multiple servers to avoid an overload. It would also be best to separate the server component and the display component of the system. As it stands, the entire system runs on the Zotac single board computer. If it were to fail, the entire system would fail. Using the Zotac only for the display component, and implementing the website component on a dedicated Apache server would be optimal. In fact, in its future implementation at Notre Dame, the software will most likely run on a dedicated remote server.

The main program might also be modified to take inputs in a more efficient manner. Collecting inputs through email is somewhat clunky. If the system were improved and backed-up more thoroughly, it could skip the Lenel server all together, and serve as the lab’s official record storage. It would be necessary to somehow link to the current facilities method of generating reports of user login and logout activity.

8. Conclusions

When we heard this project idea, we knew that it would take us way outside of our comfort zone, but the chance to make something useful was what drew us to it.

We originally decided upon this project because we wanted our project to be used. We also recognized that the project would be software-based, and that the project was technically more of a computer science project than an EE project. Ultimately, our lack of programming experience proved to be our greatest challenge, but we learned much more because of it. This project required us to become much more familiar with C++ and learn Java, PHP, MySQL, and HTML. Down the road, the programming skills we picked up during this project could come in handy for many applications in electrical engineering.

We feel that our final product not only meets the original project requirements, but also provides many extra functions and diversions without getting in the way of showing the most important information. The web site has a lot of depth in terms of customization of graphs and reports, and the iPhone and Android apps allow users to see whatever information they need very quickly and easily. We think that the project can serve the department well for at least 2-3 years and helps to highlight NDNF to visitors.

Implementing the system will require some upgrading of security and backup files. The backing up of information in the database needs to be more automated in case the system runs into a bug or a problem. If somehow an email gets read with fetchmail, but doesn't get fed through the system, that particular scan will be lost, and some of the data in the database will be slightly inaccurate. Website security needs to be upgraded because passwords for accounts are not encrypted. Someone with programming experience who really wanted to hack into the system could do it very easily at this point. We figured that functionality was more important than security until we knew for sure that it was going to be implemented.

9. Appendices

Link to Zotac Mag Datasheets:

http://www.zotac.com/index.php?option=com_wrapper&view=wrapper&Itemid=100082&lang=us

http://download.intel.com/design/processor/datashts/320528.pdf

1.main_program.cpp

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <fstream>
#include <vector>
#include <algorithm>
#include <sys/time.h>
#include "base64decoder.h"
#include "functions.h"

using namespace std;

vector<string> users;

int main()
{
int hourincounter = 0;
int houroutcounter = 0;
int dayincounter = 0;
int dayoutcounter = 0;
int date1;
int year1;
int hour1;
string month1;
string day1;
string zone1;
string daycheck;
while(1)
{
// Check for email updates
system("fetchmail");
string file;

// List all the files that contain emails
system("ls > files.txt");
ifstream input("files.txt", ifstream::in);
input>>file;

// While there are more files to take in, keep reading them in
```

```
while(file != "a.out")
{
   const char *c = file.c_str();
   ifstream stream(c, ifstream::in);
   string status;
   string code64;
   string email_body;
   string weekday;
   string day;
   string month;
   string year;
   string time;
   string monthnum;
   string stat;
   stream>>status;
   while (status!="Date:")
      stream>>status;
   stream>>weekday;  // Get the date
   stream>>day;
   stream>>month;
   stream>>year;
   stream>>time;
   while (status!="Remick")
      stream>>status;
   stream>>status;  // status now equals "entering" or "leaving"
   if (status == "Entering")
      stat = "y";
   else
      stat = "n";

   stream>>code64;
   while (code64!="base64")
      stream>>code64;
   stream>>code64;

   // Decode the base64 email
   while (!stream.eof())
   {
      email_body += base64_decode(code64);
      stream>>code64;
   }

   int i;
   int newline_num=0;
   string name;

   // Remove the newline characters
   for(i = 0; i < email_body.size(); i++)
   {
      if(email_body[i]=='\n')
        newline_num++;
      if(newline_num==2)
```

```
         name+=email_body[i];
    }

    if (day.length() == 1)
       day = "0" + day;



    if(month == "Jan")
       monthnum = "01";
    if(month == "Feb")
       monthnum = "02";
    if(month == "Mar")
       monthnum = "03";
    if(month == "Apr")
       monthnum = "04";
    if(month == "May")
       monthnum = "05";
    if(month == "Jun")
       monthnum = "06";
    if(month == "Jul")
       monthnum = "07";
    if(month == "Aug")
       monthnum = "08";
    if(month == "Sep")
       monthnum = "09";
    if(month == "Oct")
       monthnum = "10";
    if(month == "Nov")
       monthnum = "11";
    if(month == "Dec")
       monthnum = "12";

    int ind;
    string newname;

    // Remove the carriage return and newlines
    for (ind=0;ind<name.length();ind++)
    {
       if
 (name[ind]!='\n'&&name[ind]!='\r'&&name[ind]!='\''&&name[ind]!='"')
         newname+=name[ind];
    }

    string command("");

    // Call daychange
    if (day != daycheck){
    command = "php /home/ndnfinfo/daychange.php '" + year + "-" +
 monthnum + "-" + day + "'";
    system(command.c_str());
    }
```

```
   daycheck = day;
   command = "php /home/ndnfinfo/database_code.php '" + newname + "' "
+ stat + " '" + year + "-" + monthnum + "-" + day + " " + time + "'";
   // Call database_code
   system(command.c_str());

   string::iterator it;
   name.erase(name.begin());
   name.erase(name.end()-1);

   if (status == "Entering")
   {
      bool exists=false;
      for (int index=0; index<users.size(); index++)
      {
         if(users[index]==name)
            exists=true;
      }
      if(exists==false)
         users.push_back(name);
   }
   if (status == "Leaving")
   {
      for (int index=0; index<users.size(); index++)
      {
         if(users[index]==name)
            users.erase(users.begin()+index);
      }
   }

   stream.close();
   char temp[100];
   char temp2[100];
   sprintf(temp,"cp %s ./backup",file.c_str());
// RM FUNCTION!!!
   system((char *)temp);
   sprintf(temp2,"rm %s",file.c_str());
   system((char *)temp2);
   input>>file;
   string newhour1;
   int newdate;
   int newyear;
   string newmonth;
   string newday;
   string newzone;
   string file1;
   const char *d = file1.c_str();
   ifstream timefile(d, ifstream::in);
   ofstream outfile1;
   outfile1.open("day.csv", ios::app);
   ofstream outfile2;
   outfile2.open("hour.csv", ios::app);
   system("rm timedata.txt");
   system("date > timedata.txt");
   timefile.open("timedata.txt");
   timefile >> newday;
   timefile >> newmonth;
   timefile >> newdate;
   timefile >> newhour1;
   timefile >> newzone;
   timefile >> newyear;
   int newhour = atoi(newhour1.substr(0,2).c_str());

   if (hour1 != newhour)
   {
      outfile2 << newmonth << " " << newdate << " " << newyear << " "
<< newhour << ", " << hourincounter << ", " << houroutcounter << endl;
      hourincounter = 0;
      houroutcounter = 0;
      outfile2.close();
   }

   if (day1 != newday)
   {
      outfile1 << newmonth << " " << newdate << " " << newyear << ", "
<< dayincounter << ", " << dayoutcounter << endl;
      dayincounter = 0;
      dayoutcounter = 0;
      outfile1.close();
   }

   if (status == "Entering")
   {
      hourincounter += 1;
      dayincounter += 1;
   }

      if (status == "Leaving")
   {
      houroutcounter += 1;
      dayoutcounter += 1;
   }

   hour1 = newhour;
   day1 = newday;
   zone1 = newzone;
   month1 = newmonth;
   year1 = newyear;


}

system("clear");
```

```
// Print users to the terminal
for (int index=0; index<users.size(); index++)
{
   cout << users[index]<<endl;
}

// Pause before executing again
pause(5);

}

return 0;

}
```

2. database_code.php

```
<?php
/*
This is the main code for automatically entering entries into the
database, and organizing them.  The code takes 3 argument: The user's name
('John Doe'), a character variable ('y' indicates that the user is
entering, 'n' indicates that the user is leaving), and the timestamp of
the card swipe ('yyyy-MM-dd hh:mm:ss').
*/



//connect to mysql database ndnfinfo, and assign name, time, and status
variables
      mysql_connect("localhost","root","WafersD11");

//connect to db ndnfinfo
      mysql_select_db("ndnfinfo");
//Take in the name variable, and use mysql logic to get rid of quotes,
apostrophes, crosshatches, and hyphens.  The name is eventually tagged as
$name
      $name1 = $argv[1];
      $nametemp = mysql_query("SELECT REPLACE ('$name1', '-','')") or
die(mysql_error());
      $nametemp1 = mysql_fetch_array($nametemp);
      $name2 = $nametemp1[0];
      $nametemp2 = mysql_query("SELECT REPLACE ('$name2', '\'','')") or
die(mysql_error());
      $nametemp21 = mysql_fetch_array($nametemp2);
      $name3 = $nametemp21[0];
      $nametemp3 = mysql_query("SELECT REPLACE ('$name3', '#','')") or
die(mysql_error());
      $nametemp31 = mysql_fetch_array($nametemp3);
      $name4 = $nametemp31[0];
      $nametemp4 = mysql_query("SELECT REPLACE ('$name4', '\"','')") or
die(mysql_error());
      $nametemp41 = mysql_fetch_array($nametemp4);
      $name5 = $nametemp41[0];
      $nametemp5 = mysql_query("SELECT REPLACE ('$name5', '.','')") or
die(mysql_error());
      $nametemp51 = mysql_fetch_array($nametemp5);
      $name = $nametemp51[0];
// If entering, $stat = 'y'.  If leaving, $stat ='n'.
      $stat = $argv[2];
// $timein is a timestamp.  It is important that it stays in the form
'yyyy-mm-dd hh:mm:ss'.  Remember that $timein refers to the time that the
data came in, and it does not mean that the user is entering the cleanroom
at that time.
      $timein = $argv[3];
```

```
//$dateofscan is a date variable ('yyyy-mm-dd') that is used for
calculations later.  It is simply the date from the timestamp variable
$timein.
      $dateofscan1 = mysql_query("SELECT DATE('$timein')");
      $dateofscan2 = mysql_fetch_array($dateofscan1);
      $dateofscan = $dateofscan2[0];



//Since tables cannot have names with spaces those need to be removed for
a variable for a table name.  A second table, UserName_By_Day is also used
to store the total time that the user spent in the cleanroom each day.

      $nametemp11 = mysql_query("SELECT REPLACE ('$name', ' ','')") or
die(mysql_error());
      $nametemp1111 = mysql_fetch_array($nametemp11);
      $nametable = $nametemp1111[0];
      $nametable2 = $nametable . "_By_Day";



//Get date of the last time this user scanned in.  It is stored as
$lastdate
      $lastdatedata = mysql_query("SELECT lastdate FROM users2 WHERE user
= '$name'") or die(mysql_error());
      $lastdatedata1 = mysql_fetch_array($lastdatedata);
      $lastdate = $lastdatedata1['lastdate'];



//see if user is already logged in.  If the user is not already
technically signed in, $data1 will be a null.
      $query1 = "SELECT * FROM users WHERE user = '$name'";
      $data = mysql_query($query1) or die(mysql_error());
      $data1 = mysql_fetch_array($data);


//see if user is already in database system.  If they are not, $data2 will
be a null.
      $query2 = "SELECT * FROM users2 WHERE user = '$name'";
      $datatemp = mysql_query($query2) or die(mysql_error());
      $data2 = mysql_fetch_array($datatemp);

//If user is not already logged in, log them in.
      if ($data1 == "")
          {
         mysql_query("INSERT INTO users (user, tin) VALUES('$name',
'$timein' ) ") or die(mysql_error());
          }


//If user IS logged in already, delete that session, and add a new one
with the same user, and the new log in time.  The deleted session time is
still recorded, but it is done later.
```

```
      else
         {
         mysql_query("DELETE FROM users WHERE user = '$name'") or
die(mysql_error());
         mysql_query("INSERT INTO users (user, tin) VALUES('$name',
'$timein' ) ") or die(mysql_error());
         }


//If user is not already in database system, create the 2 user log tables,
and put them into the users list, a table called users2.
      if ($data2 == "")
         {
         mysql_query("CREATE TABLE $nametable (tin TIMESTAMP, tout
TIMESTAMP)") or die(mysql_error());
         mysql_query("CREATE TABLE $nametable2 (date DATE, time TIME)") or
die(mysql_error());
         mysql_query("INSERT INTO users2 (user, time, day, days, super)
VALUES('$name', '00:00:00', '00:00:00','0','YES')") or die(mysql_error());
         }


// Find the entry where the user has not yet logged out. $data31 is the
timestamp of the beginning of the session that has not ended.  If no such
session exists, it is a null.
      $datatemp3 = mysql_query("SELECT tin FROM $nametable WHERE tout =
'0000-00-00 00:00:00'") or die(mysql_error());
      $data3 = mysql_fetch_array($datatemp3);
      $data31 = $data3['tin'];



//If there is an entry that has not yet been logged out, figure out the
time between when the session started and now.  If the date is different,
divide the session at midnight into 2 seperate day's times.  Note that the
date difference cannot be more than 1 or the session will be deleted, and
not recorded.
          if($data31 != "")
              {
      //$date11 is the date that they signed in
               $date1 = mysql_query("SELECT DATE('$data31')");
             $date111 = mysql_fetch_array($date1);
             $date11 = $date111[0];
      //$date22 is the date of the card swipe
               $date2 = mysql_query("SELECT DATE('$timein')");
             $date222 = mysql_fetch_array($date2);
             $date22 = $date222[0];
      //$timeadd2 is the time difference between the card swipe, and the
beginning of the unfinished session.  In other words, it is the total time
of the session.
             $timedata = mysql_query("SELECT
TIMEDIFF('$timein','$data31')") or die(mysql_error());
               $timeadd21 = mysql_fetch_array($timedata);
               $timeadd2 = $timeadd21[0];
      //The unfinished session is removed from the log table.
               mysql_query("DELETE FROM $nametable WHERE tin = '$data31'")
or die(mysql_error());
      //$oldtime is the total time that the user had spent in the
cleanroom before this session.
               $oldtimedata = mysql_query("SELECT time FROM users2 WHERE
user = '$name'") or die(mysql_error());
               $oldtime11 = mysql_fetch_array($oldtimedata);
               $oldtime = $oldtime11['time'];
      //$newtime is the sum of $oldtime and the latest session's time.
               $newtimedata = mysql_query("SELECT
ADDTIME('$timeadd2','$oldtime')") or die(mysql_error());
               $newtime3 = mysql_fetch_array($newtimedata);
             $newtime = $newtime3[0];
      //the time in the users2 table is updated to $newtime.
             mysql_query("UPDATE users2 SET time = '$newtime' WHERE user =
'$name'") or die(mysql_error());
      //If the dates of the beginning and ending of the session are the
same, then update the current day's total log time.
               if($date11 == $date22)
               {
            //$oldday is the day's total log time before the session
                 $olddaydata = mysql_query("SELECT day FROM users2 WHERE
user = '$name'") or die(mysql_error());
                       $oldday11 = mysql_fetch_array($olddaydata);
                 $oldday = $oldday11[0];
            //$newday9 is the sum of the session time and $oldday
                 $newdaydata = mysql_query("SELECT
ADDTIME('$timeadd2','$oldday')") or die(mysql_error());
                 $newday = mysql_fetch_array($newdaydata);
            $newday9 = $newday[0];
            //update the day's total log time to $newday9
                 mysql_query("UPDATE users2 SET day = '$newday9' WHERE
user = '$name'");
               }
      //If the dates are NOT equal, divide them at midnight, the time from
the first day gets added to the old day's total log time, and sent to the
UserName_By_Day table for that user for that day.  The part from the
second day becomes the new day's total log time.  Remember, it is
impossible for $date22 and $date11 to be different by more than 1, or the
session would have already been removed.
               if($date11 != $date22)
               {
            //$referencedate one day after the beginning of the session,
and $reference is a timestamp that is midnight on that day.  This is used
as a divider.
                $referencedate1 = mysql_query("SELECT INTERVAL 1 DAY +
'$date11'") or die (mysql_error());
            $referencedate2 = mysql_fetch_array($referencedate1);
            $referencedate = $referencedate2[0];
                $reference = $referencedate . " " . '00:00:00';
            //$timeaddyesterday is the time that needs to be added to
yesterday's total log time.
```

```
                $timeaddyesterday1 = mysql_query("SELECT
TIMEDIFF('$reference','$data31')");
                $timeaddyesterday3 =
mysql_fetch_array($timeaddyesterday1);
            $timeaddyesterday = $timeaddyesterday3[0];
            //$yesterday is yesterday's total log time before the session.
                $yesterday1 = mysql_query("SELECT day FROM users2 WHERE
user = '$name'");
            $yesterday3 = mysql_fetch_array($yesterday1);
            $yesterday = $yesterday3[0];
            //Add the times together, and the sum is called
$totalyesterday
                $totalyesterday1 = mysql_query("SELECT
ADDTIME('$yesterday','$timeaddyesterday')");
                $totalyesterday3 = mysql_fetch_array($totalyesterday1);
            $totalyesterday = $totalyesterday3[0];
            //Insert this value into the user's _By_Day log table
                mysql_query("INSERT INTO $nametable2 (date,time) VALUES
('$date11','$totalyesterday')") or die(mysql_error());
            //$reference2 is the date of the end of the session at
midnight, and $todaytime is the amount of time from the session that
happened today.
            $reference2 = $date22 . " " . '00:00:00';
            $todaytime1 = mysql_query("SELECT
TIMEDIFF('$timein','$reference2')") or die(mysql_error());
            $todaytime3 = mysql_fetch_array($todaytime1);
            $todaytime = $todaytime3[0];
            //Set the current day's log time to $todaytime
            mysql_query("UPDATE users2 SET day = '$todaytime' WHERE user =
'$name'") or die(mysql_error());
            }

}



//If there is NOT a session that has not yet ended, and the newest scan is
on a different day than the last scan time, send their day's log time to
their _By_Day table.
             else{
      if($dateofscan != $lastdate && $lastdate != ''){
      $dayta1 = mysql_query("SELECT day FROM users2 WHERE user = '$name'")
or die(mysql_error());
      $dayta2 = mysql_fetch_array($dayta1);
      $dayta = $dayta2['day'];
      mysql_query("INSERT INTO $nametable2 (date,time) VALUES
('$lastdate','$dayta')") or die(mysql_error());
      mysql_query("UPDATE users2 SET day = '00:00:00' WHERE user =
'$name'") or die(mysql_error());
      }
}

//If they are going IN to the cleanroom, insert the timestamp into the
time in column of their log table.
```

```
          if($stat == 'y')
          {
          mysql_query("INSERT INTO $nametable (tin) VALUES ('$timein')")
or die(mysql_error());
          }
//Otherwise, insert the new timestamp into time out, and leave the old one
in time in.
          else
          {
          if($data31 != ""){
          mysql_query("INSERT INTO $nametable (tin,tout) VALUES
('$data31','$timein')") or die(mysql_error());
          mysql_query("DELETE FROM users WHERE user = '$name'") or
die(mysql_error());
          }
          }

//Update lastdate so it can be seen next scan.
mysql_query("UPDATE users2 SET lastdate = '$dateofscan' WHERE user =
'$name'");

/*
There is a maximum value of ~830 hours for total time, so we are including
a day counter to supplement the time counter, so people's log times don't
max out.
*/
$oldtimedata = mysql_query("SELECT time FROM users2 WHERE user = '$name'")
or die(mysql_error());
$oldtime11 = mysql_fetch_array($oldtimedata);
$oldtime = $oldtime11['time'];
$oldtimehoursdata = mysql_query("SELECT HOUR('$oldtime')") or
die(mysql_error());
$oldtimehours1 = mysql_fetch_array($oldtimehoursdata);
$oldtimehours = $oldtimehours1[0];
//While $oldtime is greater than 24 hours, subtract 24 hours and add one
to days.
while($oldtimehours > 23){
      mysql_query("UPDATE users2 SET days = days + 1 WHERE user =
'$name'") or die(mysql_error());
      mysql_query("UPDATE users2 SET time = TIMEDIFF(time,'24:00:00')
WHERE user = '$name'") or die(mysql_error());
      $oldtimedata = mysql_query("SELECT time FROM users2 WHERE user =
'$name'") or die(mysql_error());
      $oldtime11 = mysql_fetch_array($oldtimedata);
      $oldtime = $oldtime11['time'];
      $oldtimehoursdata = mysql_query("SELECT HOUR('$oldtime')") or
die(mysql_error());
      $oldtimehours1 = mysql_fetch_array($oldtimehoursdata);
      $oldtimehours = $oldtimehours1[0];
}
//update the total column
$daysdata1 = mysql_query("SELECT days from users2 where user = '$name'")
or die(mysql_error());
$daysdata = mysql_fetch_array($daysdata1);
```

```
$days = $daysdata[0];
$timedata1 = mysql_query("SELECT time FROM users2 where user = '$name'")
or die(mysql_error());
$timedata = mysql_fetch_array($timedata1) or die(mysql_error());
$time = $timedata[0];
$total = $days . ":" . $time;
mysql_query("UPDATE users2 SET total = '$total' where user = '$name'") or
die(mysql_error());




//close mysql
      mysql_close();
?>
```

3.databasereset.php

```
<?php
           mysql_connect("localhost","root","WafersD11");


           mysql_select_db("ndnfinfo");

           mysql_query("drop database ndnfinfo");
      mysql_query("create database ndnfinfo");
      mysql_select_db("ndnfinfo");
      mysql_query("create table Groups (Name VARCHAR(40),owner
VARCHAR(40))");
      mysql_query("create table coral (Machine VARCHAR(40), User
VARCHAR(40), Time_On TIMESTAMP)");
      mysql_query("create table users (user varchar(40),tin TIMESTAMP)");
      mysql_query("create table users2 (user VARCHAR(40), time TIME, day
TIME, lastdate DATE, days INT, super VARCHAR(3))");
      mysql_query("update table users2 set super = 'NO'");
      mysql_query("create table logins (user VARCHAR(40), password
varchar(40))");

mysql_close();

?>
```

4.daychange.php

```
<?php
/*This code is run the first time a scan is detected after every midnight.
If a session has gone on longer than 24 hours at this point, it will be
removed.  Note that the user will still be billed for the extra time here.
The reason they are removed is that we know they are not in there, so in
case of emergency, the accurate number of users is displayed.  If you are
implementing this system, you might consider adding a script that notifies
someone automatically if their name is removed.*/

//connect to the database
           mysql_connect("localhost","root","WafersD11");


           mysql_select_db("ndnfinfo");
//The argument that must be passed is the current date in the format
'yyyy-mm-dd'.  It is set to equal $today.
      $today = $argv[1];
//Check on all current sessions. $user is the array of users, and $Timein
is the array of log-in times.
      $data1 = mysql_query("SELECT * FROM users");
      while ($arr = mysql_fetch_array($data1)){
        $user = $arr['user'];
      $Timein = $arr['tin'];
//Create a variable for the user's log table - $nametable
      $nametemp11 = mysql_query("SELECT REPLACE ('$user', ' ','')") or
die(mysql_error());
        $nametemp1111 = mysql_fetch_array($nametemp11);
        $nametable = $nametemp1111[0];
//$date is the date that is extracted from each login timestamp.
        $date2 = mysql_query("SELECT DATE('$Timein')");
      $date21 = mysql_fetch_array($date2);
      $date = $date21[0];
//If the difference in days between the current date and the login date is
2 or more, the session is deleted.  As is, no records will be stored of
the expired session.  That can be changed since it is only a matter of
preference.
      $datediff1 = mysql_query("SELECT DATEDIFF('$today','$date')");
      $datediff2 = mysql_fetch_array($datediff1);
      $datediff = $datediff2[0];
      if($datediff >= 2){
      mysql_query("DELETE FROM users WHERE user = '$user'");
      mysql_query("DELETE FROM $nametable WHERE tin = '$Timein'");
      }
      }
      mysql_close();

?>
```

5. index.php

```
<html>
<head>
<title>Display</title>
<script language="javascript">
function reloadIframes()
      {
             frm=document.getElementsByName("current")[0];//we get the
iframe object
             frm.src=frm.src;
             frm=document.getElementsByName("super")[0];
             frm.src=frm.src;
             setTimeout("reloadIframes()",25000);  // Set refresh rate in
mSec here

      }
</script>
</head>

<body onload="reloadIframes()">
<table border="1" width=100% height=screen.height bgcolor="#001b35">
<tbody>

<html>
<body>

<table border="1" width="100%" height="100%">
<tr>
<th colspan=2 align="center" bgcolor="#001B35" ><font color="#DED5AE"
face="arial" size="25px"><strong>
Welcome to the University of Notre Dame Nanofabrication Facility!
</strong></font></th>
</tr>

<tr>
    <th width="41%" height="25" align="center"
bgcolor="#DED5AE"><font color="#001B35" face="arial"
size="20px"><strong>Current Lab
Users</strong></font></th>
    <th width="41%" height="25" align="center"
bgcolor="#DED5AE"><font color="#001B35" face="arial"
size="20px"><strong>Super
Users For the Week</strong></font></th>
</tr>

<tr>
    <td width="41%" rowspan="3" align="center"
bgcolor="#001B35"><div align="center"><iframe name = "current"
style="WIDTH: 100%; HEIGHT: 800px"
src="UsersDisplay.php" align="center" scrolling="no"
marginwidth="0" marginheight="0" frameborder="0" vspace="0"
hspace="0"></iframe></div></td>
```

```
    <td width="41%"height="204" align="center" bgcolor="#001B35"><iframe
name= "super" style="WIDTH: 100%; HEIGHT: 400px" src="Superusers.php"
scrolling="no"
marginwidth="0" marginheight="0" frameborder="0" vspace="0"
hspace="0"></iframe></td>
</tr>

<tr>
    <th width="41%" height="25" align="center"
bgcolor="#DED5AE"><font color="#001B35" face="arial"
size="25px"><strong>Usage
Graph</strong></font></th>
</tr>

<tr>
    <td height="204" align="center" bgcolor="#001B35">
<iframe style="WIDTH: 100%; HEIGHT: 400px"
- Show quoted text -
src="http://ndnfinfo.ee.nd.edu/graphs/currentUsers.htm" scrolling="no"
marginwidth="0" marginheight="0" frameborder="0" vspace="0"
hspace="0"></iframe>
</td>
</tr>

<tr>
    <th colspan="2" align="center" bgcolor="#DED5AE">

<img src="http://ndnfinfo.ee.nd.edu/500px-NotreDameSeal.svg.png"
width="54" height="54" alt=""
longdesc="http://ndnfinfo.ee.nd.edu/500px-NotreDameSeal.svg.png">

<img src="http://ndnfinfo.ee.nd.edu/UniversityBanner.png" width="260"
height="55" alt=""
longdesc="http://ndnfinfo.ee.nd.edu/UniversityBanner.png">

<img src="http://ndnfinfo.ee.nd.edu/ndnfBanner.png" width="260"
height="55" alt=""
longdesc="http://ndnfinfo.ee.nd.edu/ndnfBanner.png">

<img src="http://ndnfinfo.ee.nd.edu/ndnanoBanner.png" width="260"
height="55" alt=""
longdesc="http://ndnfinfo.ee.nd.edu/ndnanoBanner.png">

<img src="http://ndnfinfo.ee.nd.edu/MindBanner.png" width="260"
height="55" alt=""
longdesc="http://ndnfinfo.ee.nd.edu/MindBanner.png">
</th>

</tr>

<tr>

   <td align="left" bgcolor="#001B35"><font color="#DED5AE"
face="arial"><strong>
```

```
Sponsored by the Notre Dame Electrical Engineering Department
</strong></font></td>

   <td align="right" bgcolor="#001B35"><font color="#DED5AE"
face="arial"><strong>

Senior Design Project 2011: Rob Maurer, John Plunkett, Barbara Raynal,
& Matt Schueler</strong></font></td>

</tr>

</table>

</body>
</html>
```

6. UsersDisplay.php

```
<?php
      mysql_connect("localhost","root","WafersD11");

      mysql_select_db("ndnfinfo");

      $q=mysql_query("SELECT * FROM users");

      while($e=mysql_fetch_assoc($q))
      {
              $name[]=$e['user'];
              $time[]=$e['tin'];
      }
echo'<font face="arial" color="#DED5AE">'; //<marquee direction="up"
scrollamount="1" loop=true height="100%">';
echo '<div align="center">';
echo '<table id="Top Users">';

echo '</tr>', '</thead>';
echo '<tbody>';
echo '<div align="center">';
      $i=0;

      for ($i; $i<sizeof($name); $i++)
      {
          echo '<tr>';
        echo '<td>', '<font size="15px" color="#DED5AE">','<div
align="center">', $name[$i], '</div>','</font>','</td>';
          echo '</tr>';

      }
echo '</div>';
echo '</tbody>', '</table>','</div>';
//echo'</marquee></font>';
echo '</font>';

    mysql_close();
?>
```

7. Superusers.php

```
<?php

$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";

$link = @mysql_connect($db_host, $db_user, $db_pwd) or
die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");

$result = mysql_query("SELECT lastdate FROM users2 ORDER BY lastdate
DESC") or die ("Can't select table");
$data = mysql_fetch_array($result);
$lastdate = $data['lastdate'];

$result2 = mysql_query("SELECT user FROM users2 where super =
'YES'") or die (mysql_error());
while($resultdata = mysql_fetch_assoc($result2))
      $users[] = $resultdata['user'];

for($i=0;$i<count($users);$i++){
      $result3 = mysql_query("SELECT day FROM users2 WHERE user =
'$users[$i]' and lastdate = '$lastdate'") or die(mysql_error());
      $data3 = mysql_fetch_array($result3);
      $userstime[$i] = $data3['day'];
      if ($userstime[$i] == ''){
      $userstime[$i] = '00:00:00';
      }
      $temp3 = $users[$i];
      $nametemp = mysql_query("SELECT REPLACE ('$temp3',' ','')") or
die(mysql_error());
        $nametemp1 = mysql_fetch_array($nametemp);
        $nametable = $nametemp1[0];
        $nametable2 = $nametable . "_By_Day";
      for($j=1;$j<7;$j++){
            $result4 = mysql_query("SELECT time from $nametable2
where date = date_sub('$lastdate',interval $j day)") or
die(mysql_error());
            $data4 = mysql_fetch_array($result4);
            $temp = $userstime[$i];
            $temp2 = $data4['time'];
            if($temp2 == ''){
            $temp2 = '00:00:00';
            }
            $result5 = mysql_query("SELECT
ADDTIME('$temp','$temp2')") or die(mysql_error());
            $data5 = mysql_fetch_array($result5);
            $userstime[$i] = $data5[0];
}

}
```

```
array_multisort($userstime, SORT_DESC,$users);
echo '<div align="center">';
echo '<font color="#DED5AE" face="arial" size="15px"">';

for ($k = 0;$k<5;$k++){
echo $users[$k] ;
echo '<br>';

}
echo '</font>';
echo '</div>';
?>
```

Site

8. enterSite.php

```
<html>
<!--This file creates the login site for the graphing central-->
      <head>

            <title>NDNFinfo</title>
                     <script type=text/JavaScript>
                  //This functions reloads iframes every minute to make
sure information is updated
                      function reloadIframes()
                      {
                         frm=document.getElementsByName("users")[0];//we
get the iframe object
                         frm.src=frm.src;
                         setTimeout("reloadIframes()",60000);
                      }
                     </script>
            <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
            <!-- main.css must be included to take care of the formatting-
->
                <link rel="stylesheet" type="text/css" href="main.css"/>


      </head>
      <body onload="reloadIframes()">

<!--the following lines create the dropdown menu-->
<ul id="Menu" >
<!-- main.php is the faile for the main part of the graphing central-->
<li><a href="main.php" title="Enter" class="selected" >Login</a></li>
<!--check NetID and password with htaccess file on webfile and if correct
display the contents of createAccount.php-->
<li><a href="http://www.nd.edu/~ndnfinfo" title="Create Account"
target="main">Create Account</a></li>
<!--Users.php contains the names of all users currently working in the
cleanroom-->
<li><a href="Users1.php" title="Current Users" target="main">Current
Users</a></li>
</ul>
      </body>
<br></br>
<br></br>
<br></br>
<br></br>
<br></br>
<p> In order to create an account, you need to be an authorized cleanroom
user.  We will ask you to login with your netID and password.  Thank
you!</p>
<div align="center"><iframe name="users" class="border" src="Users.php"
width="100%" height="700"
```

```
frameborder="0?" scrolling="no" align ="left" name="main"
align="middle"></iframe></div>
</html>
```

9. createAccount.php

```
<html>
<body>
<!-- all the inputs from the user will be sent to saveAccount.php-->
<form action="saveAccount.php" method = "post">
<?php
//ask for username and password for their website account
echo 'Please enter a username and password to create your account','<br>';
echo 'username:','<input type="text" name="username">','<br>';
echo 'password','<input type="password" name="password">','<br>';
echo '<input type="submit">';

?>
</body>
</html>
```

10. saveAccount.php

```
<?php
//get the username and password from createAccount.php
$username = $_POST['username'];
$password = $_POST['password'];

$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";
//connect to database
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");
//check to see if an account with the desired username already exists
$query1 = mysql_query("SELECT user from logins where user = '$username'")
or die (mysql_error());
$userdata = mysql_fetch_array($query1);

// If username already exists, tell them to make a new one.
if ($userdata != ""){
echo 'Sorry, the username you chose already exists, please choose a
different one.','<br>';
echo '<a href="createAccount.php">Try again</a>';
}
// if not then create a table for this user
else{
mysql_query("insert into logins (user, password) values
('$username','$password')");
echo 'Thank you! Your account has been created.  Click the login tab to
access it.';

}
```

```
?>
```

11. main.php

```
<html>
<!--main file for the graphing central-->
      <head>
<!-- ask for username and password-->
    <?php if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="NDNFinfo"');
    header('HTTP/1.0 401 Unauthorized');
    echo '<meta HTTP-EQUIV="REFRESH" content="0;
url=http://ndnfinfo.ee.nd.edu/site/enterSite.php">';
    exit;
  }
// if a username and password were entered
else {
$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";

//connect to to mysql database
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");

//obtain the username and password entered
$User = $_SERVER['PHP_AUTH_USER'];
$password = $_SERVER['PHP_AUTH_PW'];

//check the database to see if the username and passwords are correct
$result = mysql_query("SELECT password FROM logins WHERE user='$User'") or
die ("Can't select table");
$e = mysql_fetch_assoc($result);
$checkpasswd=$e['password'];

// if username and password are correct then continue to show the contents
of the site
if ($checkpasswd==$_SERVER['PHP_AUTH_PW']){



            echo '<title>', 'NDNFinfo', '</title>';
            echo '<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />';
            //include main.css for formatting
                echo '<link rel="stylesheet" type="text/css"
href="main.css"/>';


      echo '</head>', '<body>';

// dropdown menu
echo '<ul id="Menu" >';
echo '<li>', '<a href="#" title="Groups" class="selected">', 'Groups',
'</a>';
echo '<ul>';


// select all the groups that the logged in user has created and display
them on the drop down menu under groups
$result = mysql_query("SELECT name FROM Groups WHERE owner='$User'") or
die ("Can't select table");
while($e=mysql_fetch_assoc($result))
   $output[]=$e['name'];

sort($output);


$size = sizeof($output);
$i=1;


//This loop displays the groups on the dropdown menu and if a group is
selected it sends the name of the group as a variable to groups.php which
asks for input about the graphs to display for the chosen group

foreach ($output as $value){
    echo '<li>', '<a href="../graphs/groups.php?group=', "{$value}", '"',
'title="Group" target="main">';
    echo $value, '</a></li>';}

mysql_close();

      echo '</ul>', '</li>';
echo '<li>', '<a href="#" title="Edit Group">', 'Edit Group', '</a>';
      echo '<ul>';
                  //When edit/create is selected the contents of
createNew.php are displayed in the iframe
            echo '<li>', '<a href="createNew.php" title="Edit"
target="main">', 'Edit/Create', '</a>', '<li>';
                  //when delete croup is created the contents of delete.php
are displayed on the iframe
            echo '<li>', '<a href= "delete.php" title="Edit"
target="main">', 'Delete Group', '</a>', '<li>';
      echo '</ul>', '</li>';
echo '<li>', '<a href="#" title="Graphs">', 'Graphs', '</a>';
      echo '<ul>';
            echo '<li>', '<a href="#" title="Machines" >', 'Machines',
'</a>';
                  echo '<ul>';
//when the option to graph machines is selected the contents of
machine.php in var/www/graphs/machine.php are displayed
                        echo '<li>', '<a href="../graphs/machine.php",
title="Machine" target="main">', 'Machine', '</a>', '</li>';
//when the option to graph machines is selected the contents of byUser.php
in var/www/graphs/byUser.php are displayed
```

```
                        echo '<li>', '<a href="../graphs/byUser.php",
title="By User" target="main">', 'By User', '</a>', '</li>';
                  echo '</ul>', '</li>';
      echo '</ul>', '</li>';

//when top users is selected the contents of allTime.php are displayed in
the iframe
echo '<li>', '<a href="allTime.php" title="Top Users" target = "main">',
'Top Users', '</a>', '</li>';


echo '<li>', '<a href="#" title="Machines">', 'Machines', '</a>';
      echo '<ul>';
            echo '<li>', '<a href="#" title="Reservations">',
'Reservations', '</a>';
            echo '<ul>';
                  //access psql database to obtain the names of all the
machines
                  $dbconn = pg_connect("host=anemone.nano.nd.edu
dbname=coral user=display password=gOiRiSh") or die('Could not connect: '
. pg_last_error());
                  $query = 'select distinct item from resmgr.reservation';
                  $querydata = pg_query($dbconn, "$query");
                  while($e=pg_fetch_assoc($querydata))
                  {
                        $output2[]=$e['item'];
                  } sort($output2);
                  //Display the names of all machines under reservations on
the dropdown menu
                  foreach ($output2 as $value){
                        //if a machine is clicked the name of the machine
is sent as a variable to showRes.php
                        echo '<li>', '<a
href="showRes.php?machine=',"{$value}",'"','title="mach" target="main">';
                        echo $value, '</a></li>';}
pg_close($dbconn);
            echo '</ul>','</li>';

            //when machines down is pressed the contents of
machine_statuses.php are displayed in the iframe
            echo '<li>', '<a href="machine_statuses.php" title="Machines
Down" target="main">', 'Machines Down', '</a>', '</li>';
      echo '</ul>', '</li>';
//when Current Users is pressed the contents of Users.php are displayed in
the iframe
echo '<li>', '<a href = "Users1.php" title = "Current Users"
target="main">', 'Current Users', '</a>', '</li>';
//when help is selected the contents of help.php are diplayed on the
iframe
echo '<li>', '<a href="help.php" title="Help"
target="main">','Help','</a>';
echo '</li>';

echo '</ul>';
```

```
      echo '</body>';
echo '<br>', '</br>', '<br>', '</br>', '<br>', '</br>', '<br>', '</br>',
'<br>', '</br>';
// the iframe is the window that allows the information displayed to
change based on what the user wants to see
echo '<div align="center">', '<iframe class="border" src="#" width="100%"
height="700"
frameborder="0?" scrolling="yes" align ="left" name="main"
align="middle">', '</iframe>', '</div>';

}

//if the username and password were incorrect then do no enter site
else echo 'Sorry, incorrect username and password.  Close the browser and
try again.';}

?>
</html>
```

12. groups.php

```
<?php
//This file prepares csv file for pie chart as well as sends information
to Custom_Plots.php in case scatter plot was chosen
$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";
//connect to database
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");
//get current day
$query = mysql_query("SELECT CURDATE()");
$day1 = mysql_fetch_array($query);
//get seven days ago
$query = mysql_query("select DATE_SUB(curdate(), INTERVAL 6 DAY)");
$day7 = mysql_fetch_array($query);
//the default values for begin date and end date of the plot are for the
past week but if the user changes them by inputing different numbers they
will change
$bdate = $day7[0];
$edate = $day1[0];
//get the name of group from main.php
$group  = $_GET['group'];
//all inputs will be senf to Custom_Plots.php as variables as well as the
name of the group selected
echo '<form action="Custom_Plots.php?group=', "{$group}", '"', '" method =
"post">';
echo 'Please choose the type of graph you would like to see.  The pie
chart displays information for the past seven days.  For the scatter plot
choose a period up to two weeks.', '<br>';
echo 'Type of chart (pie chart/scatter):', '<input type = "text" name =
"chart"/>', '<br>';
echo 'If you chose the scatter plot please choose begin date and end date
for the period of usage you wish to see (yyyy-mm-dd).','<br>','Begin
date:','<input type="text" name="bdate"
value="',"{$bdate}",'"','/>','<br>','End date:','<input type="text"
name="edate" value = "',"{$edate}",'"','/>','<br>';

            echo '<input  type="submit"/>';
            echo '</form>';


//prepare csv file for the pie chart
$result = mysql_query("SELECT user FROM $group") or die ("Can't select
table");
while($e=mysql_fetch_assoc($result))
   $output[]=$e['user'];

$query = mysql_query("SELECT CURDATE()");
$day1 = mysql_fetch_array($query);
```

```
$query = mysql_query("select DATE_SUB(curdate(), INTERVAL 6 DAY)");
$day7 = mysql_fetch_array($query);
// appropriate values will be writen in groupPieChart.csv
$myFile = "groupPieChart.csv";
$fh = fopen($myFile, 'w') or die("can't open file");
fwrite($fh, "Group Member,Time\n");

foreach ($output as $value){
      $name = str_replace(" ","",$value);
      $name = $name.'_By_Day';
//get times worked during the dates selected for each user of the group
from the database
      $result1 = mysql_query("SELECT SUM(TIME_TO_SEC(time)) FROM $name
WHERE date>='$day7[0]' and date<='$day1[0]'") or die ("Can't select
table");
      $resul = mysql_fetch_array($result1);

      $output1[]=$resul[0];

}

$sum=0;
$i=0;
foreach ($output1 as $value)
   $sum += $value;
foreach($output1 as $value){
$percentage = $value*100/$sum;
//write usernames and percentages in csv file
fwrite($fh, "$output[$i]");
fwrite($fh, ",");
fwrite($fh, "$percentage");
fwrite($fh, "\n");
$i++;
}
fclose($fh);
?>
```

13. Custom_Plots.php

```
<?php
//this file creates the csv file to create the scatter plot and based ont
he user's input shows the correct graph
//get the name of group, begin date, end date, and type of graph from
groups.php
$group = $_GET['group'];
$bdate = $_POST['bdate'];
$edate = $_POST['edate'];
$chart = $_POST['chart'];
//show appropriate graph depending on input
if ($chart == 'scatter'){
$site = 'http://ndnfinfo.ee.nd.edu/graphs/Custom_plot.htm';}
else $site = 'http://ndnfinfo.ee.nd.edu/graphs/groupPie.htm';
//connect to database
      mysql_connect("localhost","root","WafersD11");
      mysql_select_db("ndnfinfo");
      //prepare csv file for scatter plot
      $file = "Custom_plot.csv";
      $fh = fopen($file, 'w') or die("can't open file");
      fwrite($fh, "Categories,Users\n");

//get names of users in group
      $groupmemberdata = mysql_query("SELECT user FROM $group");
      while ($i=mysql_fetch_assoc($groupmemberdata)){
      $user[]=$i['user'];
      }
      $size = count($user);
//ad _By_Day after all names to be able to call their tables in the
database
      for($f=0;$f<$size;$f++){
      $nametemp11[$f] = mysql_query("SELECT REPLACE ('$user[$f]', '
','')") or die(mysql_error());
        $nametemp = mysql_fetch_array($nametemp11[$f]);
        $nametable[$f] = $nametemp[0];
        $nametable2[$f] = $nametable[$f] . "_By_Day";

      }

      $daydiff1 = mysql_query("select datediff('$edate','$bdate')") or
die(mysql_error());
      $daydiff2 = mysql_fetch_array($daydiff1);
      $daydiff = $daydiff2[0];

        for($n=0;$n<$daydiff;$n++){
           $datetemp = mysql_query("select date_add('$bdate',interval $n
day)") or die(mysql_error());
         $datetemp1 = mysql_fetch_array($datetemp);
         $date = $datetemp1[0];
           fwrite($fh,"$date,");

        }
```

```
//write to csv file
        fwrite($fh,"\n");
      for($p=0;$p<$size;$p++){
      for($n=0;$n<$daydiff;$n++){
            $datetemp = mysql_query("select date_add('$bdate',interval $n
day)") or die(mysql_error());
            $datetemp1 = mysql_fetch_array($datetemp);
            $date = $datetemp1[0];
            $datatemp = mysql_query("select time from $nametable2[$p]
where date = '$date'") or die(mysql_error());
            $datatemp1 = mysql_fetch_array($datatemp);
            $data = $datatemp1[0];
            if ($data == ''){
               $data = '00:00:00';
               }
                if ($n==0)
                {
                   fwrite($fh,"$user[$p],");
            }
            fwrite($fh,"$data,");
            }
          if ($p!=($size-1))
          fwrite($fh,"\n");
      }
//go to correct graph
echo '<meta HTTP-EQUIV="REFRESH" content="0; url=', "{$site}", '">';


?>
```

14. Custom_plot.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Machines Example</title>
		<!-- 1. Add these JavaScript inclusions in the head of your
page -->
		<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></sc
ript>
		<script type="text/javascript"
src="./js/highcharts.js"></script>

		<!-- 1a) Optional: add a theme file -->

		  <script type="text/javascript"
src="../js/themes/gray.js"></script>


		<!-- 1b) Optional: the exporting module -->
		<script type="text/javascript"
src="./js/modules/exporting.js"></script>


		<!-- 2. Add the JavaScript to initialize the chart on document
ready -->
		<script type="text/javascript">

var options = {
    chart: {
        renderTo: 'container',
        defaultSeriesType: 'line'
    },
    title: {
        text: 'Lab Usage by Person'
    },
    xAxis: {
        title: {
            text: 'Time'
      },
        categories: []
    },
    yAxis: {
        title: {
            text: 'hours'
        }

    },
    series: []
};

$.get('Custom_plot.csv', function(data) {
    // Split the lines
    var lines = data.split('\n');
                var series = {
                data: []
            };

    var dates = new Array();

    // Iterate over the lines and add categories or series
    $.each(lines, function(lineNo, line) {
        var items = line.split(',');

        if (lineNo == 0) {
         //   if it is the first line, do nothing
        }

        // The second line contains the dates, which are to be included on
the x-axis
        else if (lineNo == 1) {
            $.each(items, function(itemNo, item) {
                options.xAxis.categories.push(item);
            });
        }

        // the rest of the lines contain data with the user name in the
first position
        else {
            var series = {
                data: []
            };
            $.each(items, function(itemNo, item) {
                if (itemNo == 0) {
                    series.name = item;
                } else {
                    series.data.push(parseFloat(item));
                }
            });

           // Push this series onto the chart
           options.series.push(series);

        }

    });

    // Create the chart
    var chart = new Highcharts.Chart(options);
});

</script>
```

```
</head>
      <body>

            <!-- 3. Add the container -->
            <div id="container" style="width: 800px; height: 400px;
margin: 0 auto"></div>


      </body>
</html>
```

15. groupPie.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
      <head>
            <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
            <title>Lab Usage by Group Member</title>


            <!-- 1. Add these JavaScript inclusions in the head of your
page -->
            <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></sc
ript>
            <script type="text/javascript"
src="./js/highcharts.js"></script>

            <!-- 1a) Optional: add a theme file -->
            <!--
                  <script type="text/javascript"
src="../js/themes/gray.js"></script>
            -->

            <!-- 1b) Optional: the exporting module -->
            <script type="text/javascript"
src="./js/modules/exporting.js"></script>


            <!-- 2. Add the JavaScript to initialize the chart on document
ready -->
            <script type="text/javascript">

var options = {
    chart: {
      renderTo: 'container',
      plotBackgroundColor: null,
      plotBorderWidth: null,
      plotShadow: false
    },
    title: {
        text: 'Machine Usage by User'
    },
      tooltip: {
      formatter: function() {
            return '<b>'+ this.point.name +'</b>: '+ this.y +' %';
            }
      },
      plotOptions: {
      pie: {
      allowPointSelect: true,
      cursor: 'pointer',
      dataLabels: {
            enabled: true,
            color: '#000000',
            connectorColor: '#000000',
            formatter: function() {
            return '<b>'+ this.point.name +'</b>: '+ this.y +' %';
            }
      }
      }
},
    series: []
};

   $.get('groupPieChart.csv', function(data) {
      var user;
      var lines = data.split('\n');
      var series = {
           data: []
      };
      series.type = 'pie';
    // Iterate over the lines and add categories or series
    $.each(lines, function(lineNo, line) {
        var items = line.split(',');

        // header line containes categories
        if (lineNo == 0) {
       //     $.each(items, function(itemNo, item) {
        //        if (itemNo > 0) options.plotOptions.push(item);
        //    });
        }
        // the rest of the lines contain data with their name in the first
position
        else {

            $.each(items, function(itemNo, item) {
                if (itemNo == 0) {
                    user = item;
                } else {
                    series.data.push({y:parseFloat(item),name:user});
                }
            });



        }

    });
                      options.series.push(series);
    // Create the chart
    var chart = new Highcharts.Chart(options);

});
           </script>
      </head>
      <body>

            <!-- 3. Add the container -->
            <div id="container" style="width: 800px; height: 400px;
margin: 0 auto"></div>


      </body>
</html>
```

16. createNew.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="en-US" xml:lang="en-US" xmlns="http://www.w3.org/1999/xhtml">

 <head>
  <title>PHP Test</title>
<!--include css files for formatting-->
<link rel="stylesheet" type="text/css" href="createNew.css"/>
<link rel="stylesheet" type="text/css" href="stdtheme.css"/>

 </head>
 <body>

<h1><span class="color_h1">Choose lab users in your group</span></h1>
<hr/>
<p class="intro"></p>
<table id="users">
<tr>
      <th>Lab Users</th>
</tr>



 <?php

$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";
//connect to database
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");
//get names of all users and sort them
$result = mysql_query("SELECT user FROM users2") or die ("Can't select
table");
while($e=mysql_fetch_assoc($result))
   $output[]=$e['user'];

sort($output);

$size = sizeof($output);

$i=0;
$k=0;
$l=0;
$times = floor($size/3);

$residue = $size%3;

//display all users in a table with checkboxes
```

```
echo '<form action="createGroup.php" method = "post">';
  echo '<tr>';
for($l;$l<$times;$l++){

      for($k=0;$k<3;$k++)
        {
            echo '<td>', '<input type = "checkbox" class="styled" name =
"checkbox[]" value=',"{$i}",'>';
            echo $output[$i];
            echo '</td>';
            $i=$i+1;

        }
      echo '</tr>';
        if (($l+1)%2 !=0){echo '<tr class="alt">';}
      else echo '<tr>';
}
$l=0;
echo '<tr>';
for ($l;$l<$residue;$l++){

      echo '<td>', '<input type = "checkbox" class="styled" name =
"checkbox[]" value=',"{$i}",'>';
      echo $output[$i];
      echo '</td>';
      $i=$i+1;}
echo '</tr>';
//take name of group and send it to createGroup.php allong with the users
chosen
echo '</table>','<hr />';
echo 'Name of group:','<input type="text" name="groupName"/>';
echo '<input type="submit"/>';
echo '</form>';

    mysql_close();



       ?>


 </body>
</html>
```

17. createGroup.php

```
<html>
<body>

<?php
//this file creates a table for each group and makes sure existing groups
do not get overwritten
$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";
//connect to database
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");

$result = mysql_query("SELECT user FROM users2") or die ("Can't select
table");
while($e=mysql_fetch_assoc($result))
   $output[]=$e['user'];

sort($output);
//get all names selected and group name chosen from createNew.php
$checked= $_POST['checkbox'] ;
$groupName= $_POST['groupName'];
$arr[0]=$groupName;
$m=1;


foreach ($checked as $key=>$value){
      $arr[$m]=$output[$value];
      $m=$m+1;
}

      //check to see if the group already exists
        $query = "SHOW TABLES LIKE '$arr[0]'";
        $data = mysql_query($query) or die(mysql_error());
      $data1 = mysql_fetch_array($data);
        $size = count($arr);
        $start = 1;

      //if it doesn't then create it
        if($data1 == ""){
           //insert group and owner to table Groups
           $User = $_SERVER['PHP_AUTH_USER'];
                mysql_query("INSERT INTO Groups (Name,owner) VALUES
('$arr[0]' ,'$User')");
           mysql_query("CREATE TABLE $arr[0] (user VARCHAR(30))");
                $start = 1;
                for ($start;$start < $size;$start++)
                 mysql_query("INSERT INTO $arr[0] VALUES
('$arr[$start]')");
```

```
            echo 'Congratulations! Your group has been created.';
                 }
      // if group already exists, check to see if it belongs to the user
signed in
        else{
                 $checkGroup=mysql_query("SELECT owner FROM Groups WHERE
Name = '$groupName'");
            $User = $_SERVER['PHP_AUTH_USER'];
            if ($User == $checkGroup){

                 //if it does belong to the user tell them Group has
already been created, do they want to overwrite?
            echo '<form action="edit.php" method = "post">';
            echo 'This group already exists, do you wish to overwrite?
enter y/n:','<input type="text" name="overwrite"/>';
            echo '<input type="submit"/>';
            echo '</form>';}
            else {
            //if it does not belong to the user ask them to choose a
different name
            echo 'Name already taken. Please select a different name for
your group';}

            }



?>

</body>
</html>
```

18. edit.php

```
<?php
//this file edits an existing group
$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";
//connect to database
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");
//get input from createGroup.php.  did they want to overwrite?(y/n)
$over= $_POST['overwrite'];


                //If yes, $over = y.  If no, $over = n.
                if($over == 'y'){
           mysql_query("DROP TABLE $arr[0]");
           mysql_query("INSERT INTO Groups (Name) VALUES ('$arr[0]')");
           mysql_query("CREATE TABLE $arr[0] (user VARCHAR(30))");
                $start = 1;
                for ($start;$start < $size;$start++)
                 mysql_query("INSERT INTO $arr[0] VALUES
('$arr[$start]')");
           echo 'Congratulations! Your group has been created.';
                }
           if($over =='n'){
           echo '<br>','your group was not created','</br>';
           echo '<a href="createNew.php">','Try again','</a>';}

?>
```

19. delete.php

```
<html><body>
<!--this file allows the user to choose which of his groups to delete-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="en-US" xml:lang="en-US" xmlns="http://www.w3.org/1999/xhtml">

 <head>
  <title>PHP Test</title>
<!--add css files for formatting-->
<link rel="stylesheet" type="text/css" href="createNew.css"/>
<link rel="stylesheet" type="text/css" href="stdtheme.css" />

 </head>
 <body>

<h1><span class="color_h1">Choose the groups you would like to
delete</span></h1>
<hr/>
<p class="intro"></p>
<table id="users">




 <?php

$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";
//connect to database
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");
//get name of user logged in
$User = $_SERVER['PHP_AUTH_USER'];
//get all groups owned by user
$result = mysql_query("SELECT name FROM Groups WHERE owner='$User'") or
die ("Can't select table");
while($e=mysql_fetch_assoc($result))
   $output[]=$e['name'];

sort($output);

$size = sizeof($output);

$i=0;
$k=0;
$l=0;
$times = floor($size/3);
```

```
$residue = $size%3;

//send all inputs to del.php
echo '<form action="del.php" method = "post">';
  echo '<tr>';
for($l;$l<$times;$l++){
      //display all groups that belong to user
      for($k=0;$k<3;$k++)
        {
            echo '<td>', '<input type = "checkbox" class="styled" name =
"checkbox[]" value=',"{$i}",'>';
            echo $output[$i];
            echo '</td>';
            $i=$i+1;

        }
      echo '</tr>';
        if (($l+1)%2 !=0){echo '<tr class="alt">';}
      else echo '<tr>';
}
$l=0;
echo '<tr>';
for ($l;$l<$residue;$l++){

      echo '<td>', '<input type = "checkbox" class="styled" name =
"checkbox[]" value=',"{$i}",'>';
      echo $output[$i];
      echo '</td>';
      $i=$i+1;}
echo '</tr>';

echo '</table>','<hr />';
echo '<input type="submit"/>';
echo '</form>';


?>
</body>
</head>
```

20. del.php

```
<?php
//this file deletes chosen groups
$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";
//log in to database
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");
//get name of user logged in
$User = $_SERVER['PHP_AUTH_USER'];
$result = mysql_query("SELECT name FROM Groups WHERE owner='$User'") or
die ("Can't select table");
while($e=mysql_fetch_assoc($result))
   $output[]=$e['name'];

sort($output);
//get names of groups to delete from delete.php
$checked= $_POST['checkbox'] ;
$m=0;
//delete groups
foreach ($checked as $key=>$value){
      $arr[$m]=$output[$value];

      mysql_query("DROP TABLE $arr[$m]");
      mysql_query("DELETE from Groups WHERE Name='$arr[$m]'");
      $m=$m+1;
}

echo 'The group(s) you selected have been deleted';


    mysql_close();
?>
```

21. machine.php

```
<html><head>
<!--this website gets the inputs to graph user times for a given machine,
all inputs are sent to machines3.php-->
</head>

<!--inputs will be sent to machines3.php-->
<form action="machines3.php" method="post">



<?php
$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";
//connect to database
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");
//set default dates to the past week
$query = mysql_query("SELECT CURDATE()");
$day1 = mysql_fetch_array($query);
$query = mysql_query("select DATE_SUB(curdate(), INTERVAL 6 DAY)");
$day7 = mysql_fetch_array($query);

$bdate = $day7[0];

$edate = $day1[0];

//ask for begin date and end date in case they want a different time
period
echo 'Please choose begin date and end date for the period of usage you
wish to see (yyyy-mm-dd)','<br>','Begin date:','<input type="text"
name="bdate" value="',"{$bdate}",'"','/>','<br>','End date:','<input
type="text" name="edate" value = "',"{$edate}",'"','/>','<br>';
//get all machines so the user can pick one
echo '<div align="left">';
echo '<select name="machines" >';


//connect to psql database
                 $dbconn = pg_connect("host=anemone.nano.nd.edu
dbname=coral user=display password=gOiRiSh") or die('Could not connect: '
. pg_last_error());
                 $query = 'select distinct item from resmgr.reservation';
                 $querydata = pg_query($dbconn, "$query");
                 while($e=pg_fetch_array($querydata))
                 {
                       $output2[]=$e['item'];
                 }sort($output2);
//echo names of all machines
```

```
                 foreach ($output2 as $value){
                       echo '<option value="', "{$value}", '" name =
"machine"','>';
                       echo $value;
                       echo '</option>';}
//exit out of database
pg_close($dbconn);

echo '</select>';


echo '</div>';

echo '<input type="submit" value ="send">', '<input type="reset"';


           echo '</form>';


?>
</html>
```

22. machines3.php

```
<?php

//get machine and dates to graph
$machine=$_POST['machines'];
$bdate=$_POST['bdate'];
$edate=$_POST['edate'];



//connect to psql database
$dbconn = pg_connect("host=anemone.nano.nd.edu dbname=coral user=display
password=gOiRiSh") or die('Could not connect: ' . pg_last_error());



//prepare csv file for graph
$myFile = "mach_data_by_user.csv";
$fh = fopen($myFile, 'w') or die("can't open file");


$query1 = "select agent from eqmgr.eq_activity where stale=0 and
bdate>'$bdate' and edate<'$edate' and item ='$machine'" ;

$query2 = "select bdate from eqmgr.eq_activity where stale=0 and
bdate>'$bdate' and edate<'$edate' and item = '$machine'" ;

$query3 = "select edate from eqmgr.eq_activity where stale=0 and
bdate>'$bdate' and edate<'$edate' and item = '$machine'" ;


$querydata1 = pg_query($dbconn, "$query1");
$querydata2 = pg_query($dbconn, "$query2");
$querydata3 = pg_query($dbconn, "$query3");

while($e=pg_fetch_assoc($querydata1))
{
  $agent[]=$e['agent'];
}

while($e=pg_fetch_assoc($querydata2))
{
  $btime[]=$e['bdate'];
}

while($e=pg_fetch_assoc($querydata3))
{
  $etime[]=$e['edate'];
}

for($index=0;$index<count($btime);$index++)
```

```
{
      $time[$index]=(strtotime($etime[$index])-
strtotime($btime[$index]))/3600;
}


$found=0;
$end=0;
for($i=0;$i<count($agent);$i++)
{

   for($index=0;$index<$end;$index++)
   {
      if($nagent[$index]==$agent[$i])
      {
         $ntime[$index]=$ntime[$index]+$time[$i];
         $found=1;
      }
   }
   if ($found==0)
   {
      $nagent[$end]=$agent[$i];
      $ntime[$end]=$time[$i];
      $end++;
   }
   $found=0;
}
//write all info to mach_data_by_user.csv
fwrite($fh, "Categories,Machines\n");

for($i=0;$i<count($ntime);$i++)
{
  fwrite($fh, "$nagent[$i]");
  fwrite($fh, ",");
  fwrite($fh, "$ntime[$i]");
  if ($i!=(count($ntime)-1))
    fwrite($fh, "\n");
}

fclose($fh);



?>
<html>
<meta HTTP-EQUIV="REFRESH" content="0;
url=http://ndnfinfo.ee.nd.edu/graphs/readcsv3.htm">
</html>
```

23. readcsv3.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Machines Example</title>
            <!-- 1. Add these JavaScript inclusions in the head of your
page -->
            <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></sc
ript>
            <script type="text/javascript"
src="./js/highcharts.js"></script>

            <!-- 1a) Optional: add a theme file -->

              <script type="text/javascript"
src="../js/themes/gray.js"></script>


            <!-- 1b) Optional: the exporting module -->
            <script type="text/javascript"
src="./js/modules/exporting.js"></script>


            <!-- 2. Add the JavaScript to initialize the chart on document
ready -->
            <script type="text/javascript">

var options = {
    chart: {
        renderTo: 'container',
        defaultSeriesType: 'column'
    },
    title: {
        text: 'Machine Usage by User'
    },
    xAxis: {
        categories: []
    },
    yAxis: {
        title: {
            text: 'hours'
        }
    },
    series: []
};

$.get('mach_data_by_user.csv', function(data) {
    // Split the lines
    var lines = data.split('\n');

    // Iterate over the lines and add categories or series
    $.each(lines, function(lineNo, line) {
        var items = line.split(',');

        // header line containes categories
        if (lineNo == 0) {
            $.each(items, function(itemNo, item) {
                if (itemNo > 0) options.xAxis.categories.push(item);
            });
        }

        // the rest of the lines contain data with their name in the first
position
        else {
            var series = {
                data: []
            };
            $.each(items, function(itemNo, item) {
                if (itemNo == 0) {
                    series.name = item;
                } else {
                    series.data.push(parseFloat(item));
                }
            });

           options.series.push(series);

        }

    });

    // Create the chart
    var chart = new Highcharts.Chart(options);
});

</script>
</head>
      <body>

            <!-- 3. Add the container -->
            <div id="container" style="width: 800px; height: 400px;
margin: 0 auto"></div>


      </body>
</html>
```

24. byUser.php

```
<html><head>
<!--this file allows the user to select a lab user and dates to plot
machines the lab user has worked on-->
</head>

<!--all inputs will be sent to machines2.php-->
<form action="machines2.php" method="post">
<?php
$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";
//connect to database
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");
//set default dates
$query = mysql_query("SELECT CURDATE()");
$day1 = mysql_fetch_array($query);
$query = mysql_query("select DATE_SUB(curdate(), INTERVAL 6 DAY)");
$day7 = mysql_fetch_array($query);

$bdate = $day7[0];

$edate = $day1[0];
//ask for dates in case they want to change default
echo 'Please choose begin date and end date for the period of usage you
wish to see (yyyy-mm-dd)','<br>','Begin date:','<input type="text"
name="bdate" value="',"{$bdate}",'"','/>','<br>','End date:','<input
type="text" name="edate" value = "',"{$edate}",'"','/>','<br>';

echo '<div align="left">';
echo '<select name="user" >';


//connect to psql database
$dbconn = pg_connect("host=anemone.nano.nd.edu dbname=coral user=display
password=gOiRiSh") or die('Could not connect: ' . pg_last_error()); //get
al user netIDs so they can select which user to plot for
                 $query = 'select name from rscmgr.member';
                 $querydata = pg_query($dbconn, "$query");
                 while($e=pg_fetch_array($querydata))
                 {
                       $output2[]=$e['name'];
                 }sort($output2);
                 foreach ($output2 as $value){
                       echo '<option value="', "{$value}", '" name =
"user"','>';
                       echo $value;
                       echo '</option>';}
pg_close($dbconn);
```

```
echo '</select>';


echo '</div>';

echo '<input type="submit" value ="send">', '<input type="reset"';


           echo '</form>';


?>
</html>
```

25. machines2.php

```
<?php

// This script displays the times one user has spent on various machines
for a given time period
$dbconn = pg_connect("host=anemone.nano.nd.edu dbname=coral user=display
password=gOiRiSh") or die('Could not connect: ' . pg_last_error());

// Get the info the user has selected
$user=$_POST['user'];
$bdate=$_POST['bdate'];
$edate=$_POST['edate'];

// The name of the csv file used to plot the info
$myFile = "user_data_by_mach.csv";
$fh = fopen($myFile, 'w') or die("can't open file");

// Create db queries
$query1 = "select item from eqmgr.eq_activity where stale=0 and
bdate>'$bdate' and edate<'$edate' and agent ='$user'" ;

$query2 = "select bdate from eqmgr.eq_activity where stale=0 and
bdate>'$bdate' and edate<'$edate' and agent = '$user'" ;

$query3 = "select edate from eqmgr.eq_activity where stale=0 and
bdate>'$bdate' and edate<'$edate' and agent = '$user'" ;

// Query the db
$querydata1 = pg_query($dbconn, "$query1");
$querydata2 = pg_query($dbconn, "$query2");
$querydata3 = pg_query($dbconn, "$query3");

// fetch the results
while($e=pg_fetch_assoc($querydata1))
{
  $item[]=$e['item'];
}
```

```
while($e=pg_fetch_assoc($querydata2))
{
  $btime[]=$e['bdate'];
}

while($e=pg_fetch_assoc($querydata3))
{
  $etime[]=$e['edate'];
}

// the total time in hours, found by subtracting the begin time from the
end time
for($index=0;$index<count($btime);$index++)
{
      $time[$index]=(strtotime($etime[$index])-
strtotime($btime[$index]))/3600;
}

// This part calculates the time for each machine
$found=0;
$end=0;
// item[] contains all the machines from all the reservations, possibly
with duplicates
// nitem[] will have no duplicates
// time[] contains all the times from all reservations
// ntime[] contains all the times from the reservations for each machine
for($i=0;$i<count($item);$i++)  // loop through all the items listed
{
   // for each item, check if it is nitem[]
   for($index=0;$index<$end;$index++)
   {
      if($nitem[$index]==$item[$i])  // if the item is found, add the time
to total time
      {
         $ntime[$index]=$ntime[$index]+$time[$i];
         $found=1;
      }
   }

   // If the item was not found (the item wasn't already listed in
nitem[]), add it to nitem[]
   if ($found==0)
   {
      $nitem[$end]=$item[$i];
      $ntime[$end]=$time[$i];
      $end++;
   }
   $found=0;
}

fwrite($fh, "Categories,Machines\n");

// Write to the csv file
```

```
for($i=0;$i<count($ntime);$i++)
{
  fwrite($fh, "$nitem[$i]");
  fwrite($fh, ",");
  fwrite($fh, "$ntime[$i]");
  if ($i!=(count($ntime)-1))
    fwrite($fh, "\n");
}

fclose($fh);

?>
<html>
<meta HTTP-EQUIV="REFRESH" content="0;
url=http://ndnfinfo.ee.nd.edu/graphs/readcsv2.htm">
</html>
```

26. readcsv2.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Machines Example</title>
           <!-- 1. Add these JavaScript inclusions in the head of your
page -->
           <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></sc
ript>
           <script type="text/javascript"
src="./js/highcharts.js"></script>

           <!-- 1a) Optional: add a theme file -->

             <script type="text/javascript"
src="../js/themes/gray.js"></script>


           <!-- 1b) Optional: the exporting module -->
           <script type="text/javascript"
src="./js/modules/exporting.js"></script>


           <!-- 2. Add the JavaScript to initialize the chart on document
ready -->
           <script type="text/javascript">


var options = {
    chart: {
        renderTo: 'container',
        defaultSeriesType: 'column'
    },
    title: {
        text: 'Machine Usage by User'
    },
    xAxis: {
        categories: []
    },
    yAxis: {
        title: {
            text: 'hours'
        }
    },
    series: []
};

$.get('user_data_by_mach.csv', function(data) {
    // Split the lines
    var lines = data.split('\n');

    // Iterate over the lines and add categories or series
    $.each(lines, function(lineNo, line) {
        var items = line.split(',');

        // header line containes categories
        if (lineNo == 0) {
            $.each(items, function(itemNo, item) {
                if (itemNo > 0) options.xAxis.categories.push(item);
            });
        }

        // the rest of the lines contain data with their name in the first
position
        else {
            var series = {
                data: []
            };
            $.each(items, function(itemNo, item) {
                if (itemNo == 0) {
                    series.name = item;
                } else {
                    series.data.push(parseFloat(item));
                }
            });

           options.series.push(series);

        }

    });

    // Create the chart
    var chart = new Highcharts.Chart(options);
});

</script>
</head>
      <body>

            <!-- 3. Add the container -->
            <div id="container" style="width: 800px; height: 400px;
margin: 0 auto"></div>


      </body>
</html>
```

27. allTime.php

```
<html>
<!--include reservations.css for formatting-->
<link rel="stylesheet" type="text/css" href="reservations.css" />

<?php
/*This file is supposed to be placed in the directory /var/www/site.  It
is called to calculate and display top users for the day, week, and of all
time on the web site.*/


$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";
//connect to database
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");


//$totaldays is an array containing the number of days that super user
candidates have spent logged in in descending order.  There will be
repeats.
$result = mysql_query("SELECT days FROM users2 where super = 'YES' ORDER
BY days DESC") or die ("Can't select table");
while($resultdata = mysql_fetch_assoc($result))
   $totaldays[] = $resultdata['days'];
//Since there are repeats, for each value in $totaldays, there needs to be
an ordered array of the time column in users2.

//$h is initialized at zero.  For the following for loop, $h tracks the
index of the array $tops, which contains the top all time users in order.
$h = 0;

//The for loop starts at $d = the top user's total number of days, and
goes down to zero.
for ($d = $totaldays[0];$d>=0;$d--){
      //This command returns the time variable for super user candidates
who have spent $d days in the cleanroom in descending order.  $TotalTime
is that array.
   $result = mysql_query("SELECT time FROM users2 WHERE days = $d and
super = 'YES' order by time desc") or die ("Can't select table");
   while ($e = mysql_fetch_array($result))
     $totalTime[]=$e['time'];
   rsort($totalTime);
//users with the top days + time values get transferred into $tops array.
$tops is the ordered list of top users' names.
   for ($g=0;$g < count($totalTime); $g++){
      $result = mysql_query("SELECT user FROM users2 where time =
'$totalTime[$g]' and days = $d and super = 'YES'") or die(mysql_error());
      $resultdata = mysql_fetch_array($result);
      $tops[$h]=$resultdata[0];
      $h = $h + 1;
      }
//$totalTime is reset so it is empty for the next value of $d.
unset($totalTime);
}

//$today is the current date.  This is used for the day's top user.
$query = mysql_query("SELECT CURDATE()");
$today = mysql_fetch_array($query);


//The top users of the day are selected, omitting users with no time
logged, then the array is sorted.
$result = mysql_query("SELECT day FROM users2 WHERE lastdate = '$today[0]'
and day > '00:00:00' and super = 'YES'") or die ("Can't select table");
while ($e = mysql_fetch_array($result))
  $dailyTime[]=$e['day'];

rsort($dailyTime);

//$lastdate is the date of the last scan.
$result = mysql_query("SELECT lastdate FROM users2 ORDER BY lastdate
DESC") or die ("Can't select table");
$data = mysql_fetch_array($result);
$lastdate = $data['lastdate'];

//$users is an array of all super user candidates.
$result2 = mysql_query("SELECT user FROM users2 where super = 'YES'") or
die (mysql_error());
while($resultdata = mysql_fetch_assoc($result2))
      $users[] = $resultdata['user'];

//For each super user candidate, Their total time for the last 7 days is
calculated.
for($i=0;$i<count($users);$i++){
      //First their time for the current day is collected.  If it is null,
it is changed to 00:00:00 so the summing functions still work.
      $result3 = mysql_query("SELECT day FROM users2 WHERE user =
'$users[$i]' and lastdate = '$lastdate'") or die(mysql_error());
      $data3 = mysql_fetch_array($result3);
      $userstime[$i] = $data3['day'];
      if ($userstime[$i] == ''){
      $userstime[$i] = '00:00:00';
      }
      //A variable is created for the user's '_By_Day' table.
      $temp3 = $users[$i];
      $nametemp = mysql_query("SELECT REPLACE ('$temp3',' ','')") or
die(mysql_error());
        $nametemp1 = mysql_fetch_array($nametemp);
        $nametable = $nametemp1[0];
        $nametable2 = $nametable . "_By_Day";
      //For each of the past 6 days, add it to the total time sum.
      for($j=1;$j<7;$j++){
            $result4 = mysql_query("SELECT time from $nametable2 where
date = date_sub('$lastdate',interval $j day)") or die(mysql_error());
            $data4 = mysql_fetch_array($result4);
            $temp = $userstime[$i];
            $temp2 = $data4['time'];
            if($temp2 == ''){
            $temp2 = '00:00:00';
            }
            $result5 = mysql_query("SELECT ADDTIME('$temp','$temp2')") or
die(mysql_error());
            $data5 = mysql_fetch_array($result5);
            $temp4 = $data5[0];
            if($j==6)
                  $userstime[$i] = $temp4;
}

}
//Sort the parallel arrays of users and their times for the last week by
$userstime in descending order.
array_multisort($userstime, SORT_DESC,$users);








//This displays the top 10 of each category on a table.

echo '<table id="Top Users">';
echo '<thead>', '<tr>', '<th scope="col" id="user">All Time</th>';
echo '<th>', 'Today', '</th>';
echo '<th>', 'Weekly Superusers';
echo '</tr>', '</thead>';
echo '<tbody>';

for($i=0;$i<10;$i++){
$result = mysql_query("SELECT total FROM users2 WHERE user ='$tops[$i]' ")
or die ("Can't select table");
$user1=mysql_fetch_assoc($result);
$userTot = $user1['total'];
$result2 = mysql_query("SELECT user FROM users2 WHERE lastdate
='$today[0]' and day = '$dailyTime[$i]'") or die ("Can't select table");
$user2=mysql_fetch_assoc($result2);
$userDay = $user2['user'];

      //for($i;$i<$size;$i++){

            echo '<tr>';
            echo '<td>', $tops[$i], ' ', $userTot, '</td>';
            echo '<td>', $userDay, ' ', $dailyTime[$i], '</td>';
            echo '<td>', $users[$i], '<td>';
            echo '</tr>';
      //}

//echo $user, ' ', $totalTime[$i], '<br>';
}
echo '</tbody>', '</table>';
?>
</html>
```

28. showRes.php

```
<?php
//this file asks the user to select the time period for which they wish to
see a machine's reservations
$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";
//connect to database
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");
//set default dates
$query = mysql_query("SELECT CURDATE()");
$day1 = mysql_fetch_array($query);
$query = mysql_query("select DATE_SUB(curdate(), INTERVAL 6 DAY)");
$day7 = mysql_fetch_array($query);

$bdate = $day7[0];

$edate = $day1[0];
//get machine for which to show reservations from main.php
$mach  = $_GET['machine'];

// send machine  name to reservations.php
echo '<form action="reservations.php?machine=',"{$mach}",'"', 'method =
"post">';
//ask for dates in case they want to change the default
echo 'Please choose begin date and end date for the period of reservations
you wish to see (yyyy-mm-dd)','<br>','Begin date:','<input type="text"
name="bdate" value="',"{$bdate}",'"','/>', '<br>','End date:','<input
type="text" name="edate" value = "',"{$edate}",'"','/>','<br>';
            echo '<input type="submit"/>';
            echo '</form>';


?>
```

29. reservations.php

```
<html>
```

```
<!--this file displays reservations for the machine and time period
specified by the user-->
<!--include reservations.css for formatting-->
<link rel="stylesheet" type="text/css" href="reservations.css" />
<?php
//get machine name from showress.php
$mach  = $_GET['machine'];


//Connect to database
    $dbconn = pg_connect("host=anemone.nano.nd.edu dbname=coral
user=display password=gOiRiSh") or die('Could not connect: //' .
pg_last_error());
//get dates for which to show reservations
$bdate = $_POST['bdate'];
$edate  = $_POST['edate'];


$query = "select agent,bdate,edate from resmgr.reservation where stale=0
and bdate>'$bdate' and edate<'$edate' and item = '$mach'";

//get name of person who made reservation and date for reservation
$querydata = pg_query($dbconn, "$query");
while($e=pg_fetch_assoc($querydata))
{
   $output[]=$e['agent'];
$output1[]=$e['bdate'];
$output2[]=$e['edate'];

}

$size = sizeof($output);

$i=0;
//display reservations in a table
echo '<table id="reservations">';
echo '<thead>', '<tr>', '<th scope="col" id="user">NetID</th>';
echo '<th>', 'Begin date', '</th>';
echo '<th>', 'End date', '</th>', '</tr>', '</thead>';
echo '<tbody>';

for($i;$i<$size;$i++){

      echo '<tr>';
      echo '<td>', $output[$i], '</td>';
      echo '<td>', $output1[$i], '</td>';
      echo '<td>', $output2[$i], '</td>';

      echo '</tr>';

}


echo '</tbody>', '</table>';
```

```
//close database
pg_close($dbconn);

?>
</html>
```

30. machine_statuses.php

```
<?php
//this file gets the names of all machines that are down
//connect to database
$dbconn = pg_connect("host=anemone.nano.nd.edu dbname=coral user=display
password=gOiRiSh") or die('Could not connect: ' . pg_last_error());
//get names of machines that are down
$query = "SELECT name FROM eqmgr.equipment WHERE shutdowns = '1'";
$querydata = pg_query($dbconn,"$query");
//display names of machines
echo 'Machines Down:', '<br>';
while ($machine = pg_fetch_array($querydata)){
echo $machine[0] , '<br>';
}
// close database
pg_close($dbconn);
?>
```

31. Users1.php

```
<html>
<!--this file creates a table with the names of all users currently inside
cleanroom-->
<!-- style formats the table-->
<style type="text/css">
table
{
border-collapse:collapse;
}
table, td, th
{
border:1px solid #E0F574;
}
table
{
width:50%;
}
th
{
height:25px;
}
td
{
text-align:center;
}
td
{
height:25px;
vertical-align:bottom;
color:black;
}
table, td, th
{
border:1px solid #E0F574;
}
th
{
background-color:#E0F574;
color:black;
}
thead
{
font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
}
</style>

<?php
      echo '<p>','Current cleanroom users:','</p>';
//connect to database
      mysql_connect("localhost","root","WafersD11");
      mysql_select_db("ndnfinfo");

      $q=mysql_query("SELECT * FROM users");
//get users currently working in cleanroom
      while($e=mysql_fetch_assoc($q))
      {
              $name[]=$e['user'];
              $time[]=$e['tin'];
      }

echo '<table id="Top Users">';
echo '<thead>', '<tr>', '<th scope="col" id="user">User</th>';
echo '<th>', 'Time in', '</th>';
echo '</tr>', '</thead>';
echo '<tbody>';

      $i=0;
//display the names intable
      for ($i; $i<sizeof($name); $i++)
      {
          echo '<tr>';
        echo '<td>', $name[$i], '</td>';
        echo '<td>', $time[$i], '</td>';
          echo '</tr>';

      }
echo '</tbody>', '</table>';

    mysql_close();
?>
```

32. help.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="en-US" xml:lang="en-US" xmlns="http://www.w3.org/1999/xhtml">

 <head>
  <title>PHP Test</title>
<link rel="stylesheet" type="text/css" href="createNew.css"/>
<link rel="stylesheet" type="text/css" href="stdtheme.css"/>




<style type="text/css">

</style>


 </head>
 <body>
```

```
<h1><span class="color_h1">Thank you for visiting NDNFINFO Graphing
Central!</span></h1>
<hr/>
<p><font size="4"><span class="color_h1"> Groups </span></font>
<font size="3"><blockquote>If you have not yet created a group, read more
about it under the "Edit Group" section.  First, select a group that you
have already created. This will bring up the option to select the type of
chart you wish to view.  On this line, type "pie chart" or "scatter".  The
pie chart displays information for the past seven days. For the scatter
plot, choose a period up to two weeks.  A longer time period can be
selected for the scatter plot, but the graph may be cluttered.  It will
also take longer to plot since it requires more time to retrieve the
information from our database.
</blockquote></font>

<font size="4"><span class="color_h1"> Edit Group </span></font>
<font size="3"><blockquote>If you would like to create a group to monitor,
you can select “Edit Group” from the drop-down menu at the top of the
page.  Select "Edit/Create Group". This will lead you to a selection
screen with all cleanroom users listed alphabetically.  Check all of the
users you would like in your group and pick a name for your group.  Group
names must be limited to one word.  If your group name is invalid, you
will be asked to enter a different name.  Once a group is created, you
(and only you) will be able to see and select your group from the “Groups”
tab at the top left of the page.  If you wish to edit an existing group,
simply click the "Edit/Create Group" tab again, and check off the names of
the users you wish to have in the new version of your group.  Then put the
name of your group in the text line at the bottom of the screen and submit
your query.  The changes you have made to your group will now be reflected
in the plots.  Finally, if you wish to delete a group, select the "Delete
Group" tab.  You will be prompted to select which group you wish to
delete.
</blockquote></font>

<font size="4"><span class="color_h1"> Graphs </span></font>
<font size="3"><blockquote>There are two options for plotting machine
usage: by machine and by user.  The default time frame is for the past
seven days, but this can be changed.  Selecting "Machine" lets you view
how many hours each cleanroom user has spent working on the machine of
interest.  Selecting "By User" lets you see how many hours one user has
spent on each machine.
</blockquote></font>


<font size="4"><span class="color_h1"> Top Users </span></font>
<font size="3"><blockquote>Top users lists the users that have spent the
most time in the cleanroom for the past week and for the current day.
Their times are listed in the form days:hours:minutes:seconds.
</blockquote></font>


<font size="4"><span class="color_h1"> Machines </span></font>
<font size="3"><blockquote>The Machines tab can be used to check future
reservations and availability of specific machines.  Select the machine of
```

```
interest from the drop-down menu, and enter the start and end dates in the
form “yyyy-mm-dd”.  Additionally, the “Machines Down” option yields a list
of the current machines that are under maintenance.
</blockquote></font>


<font size="4"><span class="color_h1"> Current Users </span></font>
<font size="3"><blockquote>This options list the current lab users, as
well as the time that they entered the lab.
</blockquote></font>

</p>


</body>
</html>
```

33. base64decoder.cpp

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <fstream>
#include "base64decoder.h"


static const std::string base64_chars =
             "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
             "abcdefghijklmnopqrstuvwxyz"
             "0123456789+/";

static inline bool is_base64(unsigned char c) {
  return (isalnum(c) || (c == '+') || (c == '/'));
}

std::string base64_encode(unsigned char const* bytes_to_encode,
unsigned int in_len) {
  std::string ret;
  int i = 0;
  int j = 0;
  unsigned char char_array_3[3];
  unsigned char char_array_4[4];

  while (in_len--) {
    char_array_3[i++] = *(bytes_to_encode++);
    if (i == 3) {
      char_array_4[0] = (char_array_3[0] & 0xfc) >> 2;
      char_array_4[1] = ((char_array_3[0] & 0x03) << 4) +
((char_array_3[1] & 0xf0) >> 4);
      char_array_4[2] = ((char_array_3[1] & 0x0f) << 2) +
((char_array_3[2] & 0xc0) >> 6);
```

```
      char_array_4[3] = char_array_3[2] & 0x3f;

      for(i = 0; (i <4) ; i++)
        ret += base64_chars[char_array_4[i]];
      i = 0;
    }
  }

  if (i)
  {
    for(j = i; j < 3; j++)
      char_array_3[j] = '\0';

    char_array_4[0] = (char_array_3[0] & 0xfc) >> 2;
    char_array_4[1] = ((char_array_3[0] & 0x03) << 4) +
((char_array_3[1] & 0xf0) >> 4);
    char_array_4[2] = ((char_array_3[1] & 0x0f) << 2) +
((char_array_3[2] & 0xc0) >> 6);
    char_array_4[3] = char_array_3[2] & 0x3f;

    for (j = 0; (j < i + 1); j++)
      ret += base64_chars[char_array_4[j]];

    while((i++ < 3))
      ret += '=';

  }

  return ret;

}

std::string base64_decode(std::string const& encoded_string) {
  int in_len = encoded_string.size();
  int i = 0;
  int j = 0;
  int in_ = 0;
  unsigned char char_array_4[4], char_array_3[3];
  std::string ret;

  while (in_len-- && ( encoded_string[in_] != '=') &&
is_base64(encoded_string[in_])) {
    char_array_4[i++] = encoded_string[in_]; in_++;
    if (i ==4) {
      for (i = 0; i <4; i++)
        char_array_4[i] = base64_chars.find(char_array_4[i]);

      char_array_3[0] = (char_array_4[0] << 2) + ((char_array_4[1] &
0x30) >> 4);
      char_array_3[1] = ((char_array_4[1] & 0xf) << 4) +
((char_array_4[2] & 0x3c) >> 2);
```

```
      char_array_3[2] = ((char_array_4[2] & 0x3) << 6) +
char_array_4[3];

      for (i = 0; (i < 3); i++)
        ret += char_array_3[i];
      i = 0;
    }
  }

  if (i) {
    for (j = i; j <4; j++)
      char_array_4[j] = 0;

    for (j = 0; j <4; j++)
      char_array_4[j] = base64_chars.find(char_array_4[j]);

    char_array_3[0] = (char_array_4[0] << 2) + ((char_array_4[1] &
0x30) >> 4);
    char_array_3[1] = ((char_array_4[1] & 0xf) << 4) +
((char_array_4[2] & 0x3c) >> 2);
    char_array_3[2] = ((char_array_4[2] & 0x3) << 6) +
char_array_4[3];

    for (j = 0; (j < i - 1); j++) ret += char_array_3[j];
  }

  return ret;
}
```

34. base64decoder.h

```
#ifndef FILE_BASE64DECODER_SEEN
#define FILE_BASE64DECODER_SEEN

static inline bool is_base64(unsigned char c);
std::string base64_encode(unsigned char const* bytes_to_encode,
unsigned int in_len);
std::string base64_decode(std::string const& encoded_string);

#endif
```

35. functions.cpp

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <fstream>
#include <vector>
```

```
#include <algorithm>
#include <sys/time.h>
#include "functions.h"

using namespace std;
// A pause function that uses the system clock to wait a specified
period
int pause(int secs)
{
    long tm1,test;
    time(&tm1);
    test = tm1 + secs;
    while(tm1 < test)
    {
         time(&tm1);
    }
    return 0;
}

void update_html(vector<string> users)
{
   ofstream myfile;
   system("rm users.html");
   myfile.open ("users.html");
   myfile <<
   "<html> <head> <script type=\"text/JavaScript\"> function
timedRefresh(timeoutPeriod)
{setTimeout(\"location.reload(true);\",timeoutPeriod);} </script>
</head> <body onload=\"JavaScript:timedRefresh(5000);\"> ";
   myfile << "Users: <br />" << endl;
   for (int i=0; i < users.size(); i++)
   {
         myfile << users[i] << "<br />" << endl;
   }
   myfile << "</html>" << endl;
   myfile << "</body>" << endl;
   myfile.close();
   system("./sftp.sh");
}
```

36. functions.h

```
#ifndef FILE_FUNCTIONS
#define FILE_FUNCTIONS

int pause(int secs);
void update_html(std::vector<std::string> users);

#endif
```

Android

37. machinenames.php

```
<?php

     // This script is used to show the names of all the machines
currently in the coral database to the user
     $dbconn = pg_connect("host=anemone.nano.nd.edu dbname=coral
user=display password=gOiRiSh") or die('Could not connect: ' .
pg_last_error());

     $query = 'select distinct item from resmgr.reservation';
     $querydata = pg_query($dbconn, "$query");
     while($e=pg_fetch_assoc($querydata))
     {
      $output[]=$e;
     }

     print(json_encode($output));

     pg_close($dbconn);
?>
```

38. machine_statuses.php

```
<?php

// This script is used to show the names of all the machines currently
shutdown

$dbconn = pg_connect("host=anemone.nano.nd.edu dbname=coral user=display
password=gOiRiSh") or die('Could not connect: ' . pg_last_error());
$query = "SELECT name FROM eqmgr.equipment WHERE shutdowns = '1'";
$querydata = pg_query($dbconn,"$query");

while ($e = pg_fetch_assoc($querydata)){
 $output[]=$e;
}

print(json_encode($output));
pg_close($dbconn);


?>
```

39. machres.php

```
<?php

// This script shows all the reservations from the start date to end date
for a given machine
    $dbconn = pg_connect("host=anemone.nano.nd.edu dbname=coral
user=display password=gOiRiSh") or die('Could not connect: ' .
pg_last_error());

// Get the start date, end date, and machine from the url passed by the
user
$sdate = $_GET['sdate'];
$edate = $_GET['edate'];
$mach  = $_GET['mach'];

$query = "select agent,bdate,edate from resmgr.reservation where stale=0
and bdate>'$sdate' and edate<'$edate' and item = '$mach'";


$querydata = pg_query($dbconn, "$query");
while($e=pg_fetch_assoc($querydata))
{
   $output[]=$e;
}

print(json_encode($output));
pg_close($dbconn);

?>
```

40. rankings.php

```
<?php
      // This script shows all the top lab users for the day
      mysql_connect("localhost","root","WafersD11");

      mysql_select_db("ndnfinfo");

      // Get the daily time for each user that has been in today
      $result = mysql_query("SELECT day FROM users2 WHERE lastdate =
CURDATE()") or die ("Can't select table");
      while ($e = mysql_fetch_array($result))
         $totalTime[]=$e['day'];

      // sort the times
      rsort($totalTime);

      // select the users whose times correspond to the times given
      for($i=0;$i<10;$i++){
         $result = mysql_query("SELECT user FROM users2 WHERE day
='$totalTime[$i]' ") or die ("Can't select table");
         $user1=mysql_fetch_assoc($result);
         $userTot[] = $user1['user'];
      }

      // This section encodes the information in JSON, a form
understandable by the Android.
      echo '[';
      for($i=0;$i<10;$i++){
      if ($userTot[$i]!="" && $totalTime[$i]!='00:00:00')
      {
        echo '{';
        echo '"user":"';
        echo "$userTot[$i]";
        echo '","';
        echo 'time":"';
        echo "$totalTime[$i]";
        echo '"';
        echo '}';
        if ($i!=9)
          echo ',';
      }
      }
      echo ']';
    mysql_close();
?>
```

41. userhis.php

```php
<?php

// This script shows the history of logins and logouts for a given user
mysql_connect("localhost","root","WafersD11");

mysql_select_db("ndnfinfo");

$sdate = $_GET['sdate'];
$edate = $_GET['edate'];
$user  = $_GET['user'];

$query = "select * from $user where tin>'$sdate' and tout<'$edate'";

$querydata = mysql_query($query);

while($e=mysql_fetch_assoc($querydata))
{
   $output[]=$e;
}

print(json_encode($output));
mysql_close();

?>
```

42. usernames.php

```
<?php
      // This script shows all the users currently in our database
      mysql_connect("localhost","root","WafersD11");

      mysql_select_db("ndnfinfo");

      $q=mysql_query("SELECT user FROM users2");

      while($e=mysql_fetch_assoc($q))

              $output[]=$e;

      print(json_encode($output));

    mysql_close();
?>
```

43. users.php

```
<?php
      // This script shows all the users currently in the cleanroom
      mysql_connect("localhost","root","WafersD11");

      mysql_select_db("ndnfinfo");

      $q=mysql_query("SELECT * FROM users");

      while($e=mysql_fetch_assoc($q))

              $output[]=$e;

           print(json_encode($output));

    mysql_close();
?>
```

iPhone

44. index.htm

```
<html>
<head>

<style type ="text/css" >
body {background-image:url('AppBackground.jpg');
width: 100%;
height: 100%;
position: absolute;
top:0;
left:0;}

input {
   font-family: "Times New Roman", Times, serif;
   font-size: 18px;
   font-weight: bold;
   color: #000000;
   background-color: #FFFFFF;
   padding: 5px;
   height: 30px;
   width: 150px;
}
input.myButton {
   font-family: Arial, Helvetica, sans-serif;
   font-size: 20px;
   font-style: italic;
   color: #000000;
   background-color: #CCCCFF;
   padding: 2px;
   height: 50px;
   width: 100px;
   border: 1px solid #000000;
}

</style>
</head>
 <body>

<h1><font size="6" color = 'blue'><pre> <div align="center">NDNFINFO
Login</div></pre></font></h1><br>
<form action="login.php" method = "post">
<br> <font size='5' color='blue'>
<b>Login:</b>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</font><input
type="text" name="username"><br>
<br> </br><font size='5' color='blue'> <b>Password:</b> </font><input
type="password" name="password"><br><br><br><br>
<div align="center"><input type="submit"  name="Submit" value="Submit"
class="myButton" ></div>
</form>
```

```
 </body>

</html>
```

45. login.php

```
<?php

$username = $_POST['username'];
$password = $_POST['password'];

mysql_connect("localhost","root","WafersD11");

mysql_select_db("ndnfinfo");

$q=mysql_query("SELECT * FROM logins");

while($e=mysql_fetch_assoc($q))
{
    $users[]=$e['user'];
    $pwords[]=$e['password'];
}

for ($i=0; $i< sizeof($users); $i++)
{
  if ($username == $users[$i] && $password == $pwords[$i])
  {
     include "homepage.php";
     break;
  }
  else if ($i == sizeof($users)-1)
    echo 'Not a valid username-password combination.  Please go back and 
try again';

}


?>
```

46. machines.php

```
<html>
<style type ="text/css" >
body {background-image:url('AppBackground.jpg');
width: 100%;
height: 100%;
position: absolute;
top:0;
left:0;}
input {
   font-family: "Times New Roman", Times, serif;
   font-size: 12px;
   font-weight: bold;
   color: #000000;
   background-color: #FFFFFF;
   padding: 5px;
   height: 30px;
   width: 100px;
}
input.myButton {
   font-family: Arial, Helvetica, sans-serif;
   font-size: 20px;
   font-style: italic;
   color: #000000;
   background-color: #CCCCFF;
   padding: 2px;
   height: 50px;
   width: 100px;
   border: 1px solid #000000;
}
</style>


<head>

</head>
<body>
<br>
<h1><font size="6"><div align="center">Machine Usage for Previous 7
Days</div></font></h1>
<div align="center">
<br>
<form action="machines_by_user.php" method="post">




<?php
$db_host = "localhost";
$db_user = "root";
$db_pwd = "WafersD11";
$database = "ndnfinfo";
```

```
$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());
$link2 = mysql_select_db($database) or die("Can't select database");
$query = mysql_query("SELECT CURDATE()");
$day1 = mysql_fetch_array($query);
$query = mysql_query("select DATE_SUB(curdate(), INTERVAL 6 DAY)");
$day7 = mysql_fetch_array($query);



echo '<div align="center">';
echo '<select name="machines" >';



                 $dbconn = pg_connect("host=anemone.nano.nd.edu
dbname=coral user=display password=gOiRiSh") or die('Could not connect: '
. pg_last_error());
                 $query = 'select distinct item from resmgr.reservation';
                 $querydata = pg_query($dbconn, "$query");
                 while($e=pg_fetch_array($querydata))
                 {
                       $output2[]=$e['item'];
                 }sort($output2);
                 foreach ($output2 as $value){
                       echo '<option value="', "{$value}", '" name =
"machine"','>';
                       echo $value;
                       echo '</option>';}
pg_close($dbconn);

echo '</select>', '<br>','<br>','<br>';
echo '<input type="submit" value ="submit">';
echo '</div>';

echo '</form>';
/*echo '<form action="../graphs/Custom_Plots.php" method="post">';
echo '<div align="center">';
echo'<select name="group">';
$User = $_SERVER['PHP_AUTH_USER'];
echo $User;
$result = mysql_query("SELECT name FROM Groups WHERE owner='$User'") or
die ("Can't select table");
while($e=mysql_fetch_assoc($result))
   $output[]=$e['name'];

sort($output);


$size = sizeof($output);
$i=1;
```

```
foreach ($output as $value){
    echo '<option value="', "{$value}", '">';
      echo $value;
      echo '</option>';}
echo '</select>','<br>';
echo '<input type="submit" value="submit">';
echo '</div>';
echo '</form>';*/

mysql_close();



?>
</body>
</html>
```

47. machines_by_user.php

```
<?php

$dbconn = pg_connect("host=anemone.nano.nd.edu dbname=coral user=display
password=gOiRiSh") or die('Could not connect: ' . pg_last_error());

$query = pg_query($dbconn, "SELECT current_date");
$day1 = pg_fetch_array($query);
$query = pg_query($dbconn, "select current_date - integer '7'");
$day7 = pg_fetch_array($query);

$bdate = $day7[0];

$edate = $day1[0];

$machine=$_POST['machines'];
//$bdate=$_POST['bdate'];
//$edate=$_POST['edate'];


$myFile = "./mach_data_by_user.csv";
$fh = fopen($myFile, 'w') or die("can't open file");


$query1 = "select agent from eqmgr.eq_activity where stale=0 and
bdate>'$bdate' and edate<'$edate' and item ='$machine'" ;

$query2 = "select bdate from eqmgr.eq_activity where stale=0 and
bdate>'$bdate' and edate<'$edate' and item = '$machine'" ;

$query3 = "select edate from eqmgr.eq_activity where stale=0 and
bdate>'$bdate' and edate<'$edate' and item = '$machine'" ;


$querydata1 = pg_query($dbconn, "$query1");
$querydata2 = pg_query($dbconn, "$query2");
$querydata3 = pg_query($dbconn, "$query3");

while($e=pg_fetch_assoc($querydata1))
{
  $agent[]=$e['agent'];
}

while($e=pg_fetch_assoc($querydata2))
{
  $btime[]=$e['bdate'];
}

while($e=pg_fetch_assoc($querydata3))
{
  $etime[]=$e['edate'];
}
```

```
for($index=0;$index<count($btime);$index++)
{
      $time[$index]=(strtotime($etime[$index])-
strtotime($btime[$index]))/3600;
}


$found=0;
$end=0;
for($i=0;$i<count($agent);$i++)
{

   for($index=0;$index<$end;$index++)
   {
      if($nagent[$index]==$agent[$i])
      {
         $ntime[$index]=$ntime[$index]+$time[$i];
         $found=1;
      }
   }
   if ($found==0)
   {
      $nagent[$end]=$agent[$i];
      $ntime[$end]=$time[$i];
      $end++;
   }
   $found=0;
}

fwrite($fh, "Categories,Machines\n");
for($i=0;$i<count($ntime);$i++)
{
  fwrite($fh, "$nagent[$i]");
  fwrite($fh, ",");
  fwrite($fh, "$ntime[$i]");
  if ($i!=(count($ntime)-1))
    fwrite($fh, "\n");
}

fclose($fh);



?>
<html>
<meta HTTP-EQUIV="REFRESH" content="0;
url=http://ndnfinfo.ee.nd.edu/iphone/readCSV3.htm">
</html>
```

48. machres.php (iPhone)

```
<html>
<style type ="text/css" >
body {background-image:url('AppBackground.jpg');
width: 100%;
height: 100%;
position: absolute;
top:0;
left:0;}
input {
   font-family: "Times New Roman", Times, serif;
   font-size: 20px;
   font-weight: bold;
   color: #000000;
   background-color: #FFFFFF;
   padding: 5px;
   height: 30px;
   width: 100px;
}
input.myButton {
   font-family: Arial, Helvetica, sans-serif;
   font-size: 20px;
   font-style: italic;
   color: #000000;
   background-color: #CCCCFF;
   padding: 2px;
   height: 50px;
   width: 100px;

   border: 1px solid #000000;

}

</style>

<head>


</head>

<body>

<br>

<h1><font size="6"><div align="center">Machine Reservations for Previous
and Next 7 Days</div></font></h1>

<div align="center">

<br>

<form action="reservations.php" method="post">







<?php

$db_host = "localhost";

$db_user = "root";

$db_pwd = "WafersD11";

$database = "ndnfinfo";


$link = @mysql_connect($db_host, $db_user, $db_pwd) or die(mysql_error());

$link2 = mysql_select_db($database) or die("Can't select database");

$query = mysql_query("SELECT CURDATE()");

$day1 = mysql_fetch_array($query);
```

```
$query = mysql_query("select DATE_SUB(curdate(), INTERVAL 6 DAY)");

$day7 = mysql_fetch_array($query);

echo '<div align="center">';

echo '<select name="machines" >';

                 $dbconn = pg_connect("host=anemone.nano.nd.edu
dbname=coral user=display password=gOiRiSh") or die('Could not connect: '
. pg_last_error());

                 $query = 'select distinct item from resmgr.reservation';

                 $querydata = pg_query($dbconn, "$query");

                 while($e=pg_fetch_array($querydata))

                 {

                       $output2[]=$e['item'];

                 }sort($output2);

                 foreach ($output2 as $value){

                       echo '<option value="', "{$value}", '" name =
"machine"','>';

                       echo $value;

                       echo '</option>';}

pg_close($dbconn);


echo '</select>', '<br>','<br>','<br>';

echo '<input type="submit" value ="submit">';

echo '</div>';


echo '</form>';

/*echo '<form action="../graphs/Custom_Plots.php" method="post">';

echo '<div align="center">';

echo'<select name="group">';

$User = $_SERVER['PHP_AUTH_USER'];
```

```
echo $User;

$result = mysql_query("SELECT name FROM Groups WHERE owner='$User'") or
die ("Can't select table");

while($e=mysql_fetch_assoc($result))

   $output[]=$e['name'];


sort($output);




$size = sizeof($output);

$i=1;

foreach ($output as $value){

    echo '<option value="', "{$value}", '">';

      echo $value;

      echo '</option>';}

echo '</select>','<br>';

echo '<input type="submit" value="submit">';

echo '</div>';

echo '</form>';*/


mysql_close();

?>

</body>

</html>
```

49. readCSV3.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Machines Example</title>
           <!-- 1. Add these JavaScript inclusions in the head of your
page -->
           <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></sc
ript>
           <script type="text/javascript"
src="../graphs/js/highcharts.js"></script>

           <!-- 1a) Optional: add a theme file -->

             <script type="text/javascript"
src="../graphs/js/themes/gray.js"></script>


           <!-- 1b) Optional: the exporting module -->
           <script type="text/javascript"
src="../graphs/js/modules/exporting.js"></script>


           <!-- 2. Add the JavaScript to initialize the chart on document
ready -->
           <script type="text/javascript">

var options = {
    chart: {
        renderTo: 'container',
        defaultSeriesType: 'column'
    },
    title: {
        text: 'Machine Usage by User'
    },
    xAxis: {
        categories: []
    },
    yAxis: {
        title: {
            text: 'hours'
        }
    },
    series: []
};

$.get('mach_data_by_user.csv', function(data) {
    // Split the lines
    var lines = data.split('\n');
    // Iterate over the lines and add categories or series
    $.each(lines, function(lineNo, line) {
        var items = line.split(',');

        // header line containes categories
        if (lineNo == 0) {
            $.each(items, function(itemNo, item) {
                if (itemNo > 0) options.xAxis.categories.push(item);
            });
        }

        // the rest of the lines contain data with their name in the first
position
        else {
            var series = {
                data: []
            };
            $.each(items, function(itemNo, item) {
                if (itemNo == 0) {
                    series.name = item;
                } else {
                    series.data.push(parseFloat(item));
                }
            });

           options.series.push(series);

        }

    });

    // Create the chart
    var chart = new Highcharts.Chart(options);
});

</script>
</head>
      <body>

            <!-- 3. Add the container -->
            <div id="container" style="width: 300px; height: 400px;
margin: 0 auto"></div>


      </body>
</html>
```

50. reservations.css

```
table
{
border-collapse:collapse;
}
table, td, th
{
border:1px solid #E0F574;
}
table
{
width:50%;
}
th
{
height:50px;
}
td
{
text-align:center;
}
td
{
height:50px;
vertical-align:bottom;
}
table, td, th
{
border:1px solid #E0F574;
}
th
{
background-color:#E0F574;
color:black;
}
thead
{
font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
}
```

51. reservations.php

```
<html>
<link rel="stylesheet" type="text/css" href="reservations.css" />
<?php



//Connect
    $dbconn = pg_connect("host=anemone.nano.nd.edu dbname=coral
user=display password=gOiRiSh") or die('Could not connect: //' .
pg_last_error());

$query = pg_query($dbconn, "SELECT current_date + integer '7'");
$day1 = pg_fetch_array($query);
$query = pg_query($dbconn, "select current_date - integer '7'");
$day7 = pg_fetch_array($query);

$bdate = $day7[0];
$machine=$_POST['machines'];
$edate = $day1[0];

$query = "select agent,bdate,edate from resmgr.reservation where stale=0
and bdate>'$bdate' and edate<'$edate' and item = '$machine'";


$querydata = pg_query($dbconn, "$query");
while($e=pg_fetch_assoc($querydata))
{
   $output[]=$e['agent'];
$output1[]=$e['bdate'];
$output2[]=$e['edate'];

}

$size = sizeof($output);
//$times = floor($size/3);
//$residue = $size%3;
$i=0;
echo '<table id="reservations">';
echo '<thead>', '<tr>', '<th scope="col" id="user">NetID</th>';
echo '<th>', 'Begin date', '</th>';
echo '<th>', 'End date', '</th>', '</tr>', '</thead>';
echo '<tbody>';

for($i;$i<$size;$i++){

      echo '<tr>';
      echo '<td>', $output[$i], '</td>';
      echo '<td>', $output1[$i], '</td>';
      echo '<td>', $output2[$i], '</td>';

      echo '</tr>';
}

//$i=0;
//for($i;$i<$residue;$i++){
      //echo '<tr>';
      //echo '<td>', $output[$i], '</td>';
      //echo '<td>', $output1[$i], '</td>';
      //echo '<td>', $output2[$i], '</td>';
      //echo '</tr>';
      //$i=$i+1;}

echo '</tbody>', '</table>';




pg_close($dbconn);

?>
</html>
```

52. super.php

```
<?php
 mysql_connect("localhost","root","WafersD11");

      mysql_select_db("ndnfinfo");

$result = mysql_query("SELECT lastdate FROM users2 ORDER BY lastdate
DESC") or die ("Can't select table");
$data = mysql_fetch_array($result);
$lastdate = $data['lastdate'];

$result2 = mysql_query("SELECT user FROM users2 where super = 'YES'") or
die (mysql_error());
while($resultdata = mysql_fetch_assoc($result2))
      $users[] = $resultdata['user'];

for($i=0;$i<count($users);$i++){
      $result3 = mysql_query("SELECT day FROM users2 WHERE user =
'$users[$i]' and lastdate = '$lastdate'") or die(mysql_error());
      $data3 = mysql_fetch_array($result3);
      $userstime[$i] = $data3['day'];
      if ($userstime[$i] == ''){
      $userstime[$i] = '00:00:00';
      }
      $temp3 = $users[$i];
      $nametemp = mysql_query("SELECT REPLACE ('$temp3',' ','')") or
die(mysql_error());
        $nametemp1 = mysql_fetch_array($nametemp);
        $nametable = $nametemp1[0];
        $nametable2 = $nametable . "_By_Day";
      for($j=1;$j<7;$j++){
            $result4 = mysql_query("SELECT time from $nametable2 where
date = date_sub('$lastdate',interval $j day)") or die(mysql_error());
            $data4 = mysql_fetch_array($result4);
            $temp = $userstime[$i];
            $temp2 = $data4['time'];
            if($temp2 == ''){
            $temp2 = '00:00:00';
            }
            $result5 = mysql_query("SELECT ADDTIME('$temp','$temp2')") or
die(mysql_error());
            $data5 = mysql_fetch_array($result5);
            $temp4 = $data5[0];
            if($j==6)
                  $userstime[$i] = $temp4;
}

}

array_multisort($userstime, SORT_DESC,$users);



for($k=0,$k<5;$k++){
echo $users[$k] . "\n";
}


?>
```

53. topusers.php

```
<html>
<link rel="stylesheet" type="text/css" href="reservations.css" />
<?php


      mysql_connect("localhost","root","WafersD11");

      mysql_select_db("ndnfinfo");

      $result = mysql_query("SELECT day FROM users2 WHERE lastdate =
CURDATE()") or die ("Can't select table");
      while ($e = mysql_fetch_array($result))
      $totalTime[]=$e['day'];
      rsort($totalTime);
      for($i=0;$i<10;$i++){
         $result = mysql_query("SELECT user FROM users2 WHERE day
='$totalTime[$i]' ") or die ("Can't select table");
         $user1=mysql_fetch_assoc($result);
         $userTot[] = $user1['user'];
      }

   echo '<b>','Top Users for Today','</b>','<br>';
   echo '<table id="reservations">';
   echo '<thead>', '<tr>', '<th scope="col" id="user">User Name</th>';
   echo '<th>', 'Time (hh:mm:ss)', '</th>';
   echo '</tr>', '</thead>';
   echo '<tbody>';

      for($i=0;$i<10;$i++){
      if ($userTot[$i]!="" && $totalTime[$i]!='00:00:00')
      {
      echo '<tr>';
      echo '<td>', $userTot[$i], '</td>';
      echo '<td>', $totalTime[$i], '</td>';

      echo '</tr>';
      }
      }


    mysql_close();
?>
```

54. Users.php

```
<html>

<?php


      mysql_connect("localhost","root","WafersD11");

      mysql_select_db("ndnfinfo");

      $q=mysql_query("SELECT * FROM users");

      while($e=mysql_fetch_assoc($q))
      {
              $name[]=$e['user'];
              $time[]=$e['tin'];
      }

      $i=0;
     echo'<font color="#CC6633">';
echo '<div align="center">';
echo '<b>';
echo 'Current Users','<br>','<br>';
echo '</b>';

      $i=0;

      for ($i; $i<sizeof($name); $i++)
      {

        echo  $name[$i], '<br>';



      }
echo '</div>';
echo'</font>';

    mysql_close();
?>
```

Passwords

55. checkpassword.php

```
<?php
      // This script allows the android app to check if someone's password
is valid
      mysql_connect("localhost","root","WafersD11");

      mysql_select_db("ndnfinfo");

      $q=mysql_query("SELECT *
                      FROM logins");

      while($e=mysql_fetch_assoc($q))
              $output[]=$e;

           print(json_encode($output));

    mysql_close();
?>
```

Android Source Code

com.seniordesign

56. AndroidManifest.xml

```
  <?xml version="1.0" encoding="utf-8" ?>
- <manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.seniordesign" android:versionCode="1" android:versionName="1.0">
  <uses-sdk android:minSdkVersion="7" />
- <application android:icon="@drawable/icon" android:label="@string/app_name">
- <activity android:name=".Main" android:label="@string/app_name">
- <intent-filter>
  <action android:name="android.intent.action.MAIN" />
  <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    </activity>
  <activity android:name=".rankings" android:label="@string/app_name" />
  <activity android:name=".history" android:label="@string/app_name" />
  <activity android:name=".showhis" android:label="@string/app_name" />
  <activity android:name=".mainpage" android:label="@string/app_name" />
  <activity android:name=".showres" android:label="@string/app_name" />
  <activity android:name=".graphs" android:label="@string/app_name" />
  <activity android:name=".users" android:label="@string/app_name" />
  <activity android:name=".machines" android:label="@string/app_name" />
  <activity android:name=".reservations" android:label="@string/app_name" />
  <activity android:name=".connect" android:theme="@android:style/Theme.NoTitleBar"
    android:label="@string/app_name" />
    </application>

  <uses-permission android:name="android.permission.INTERNET" />

    </manifest>

  <?xml version="1.0" encoding="utf-8" ?>

    </manifest>
```

57. connect.java

```
package com.seniordesign;

import android.app.TabActivity;
import android.content.Intent;
import android.content.res.Resources;
import android.os.Bundle;
import android.widget.TabHost;
import android.widget.TextView;


public class connect extends TabActivity {
/** Called when the activity is first created. */
   TextView txt;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Resources res = getResources();
    TabHost tabHost = getTabHost();
    TabHost.TabSpec spec;
    Intent intent;

    // Create an Intent to launch an Activity for the tab (to be
reused)
    intent = new Intent().setClass(this, users.class);

    // Initialize a TabSpec for each tab and add it to the TabHost
    spec = tabHost.newTabSpec("users").setIndicator("Users",
                      res.getDrawable(R.drawable.group))
                  .setContent(intent);
    tabHost.addTab(spec);

    // Do the same for the other tabs
    intent = new Intent().setClass(this, machines.class);
    spec = tabHost.newTabSpec("machines").setIndicator("Machines",
                      res.getDrawable(R.drawable.machine))
                  .setContent(intent);
    tabHost.addTab(spec);
    intent = new Intent().setClass(this, rankings.class);
    spec = tabHost.newTabSpec("rankings").setIndicator("Rankings",
                      res.getDrawable(R.drawable.ranking))
                  .setContent(intent);
    tabHost.addTab(spec);

    tabHost.setCurrentTab(2);


}

}
```

58. graphs.java

```
package com.seniordesign;

import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebView;

public class graphs extends Activity {
    /** Called when the activity is first created. */
     WebView mWebView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.webview);

        mWebView = (WebView) findViewById(R.id.webview);
        mWebView.getSettings().setJavaScriptEnabled(true);

mWebView.loadUrl("http://ndnfinfo.ee.nd.edu/graphs/readcsv3.htm");
    }
}
```

59. history.java

```
package com.seniordesign;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Calendar;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
```

```java
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.DialogInterface.OnClickListener;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;

public class history extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    private TextView mDateDisplay;
    private Button mPickDate;
    private TextView mDateDisplay2;
    private TextView mPickDate2;
    private int mYear;
    private int mMonth;
    private int mDay;

    private int mYear2;
    private int mMonth2;
    private int mDay2;

    static final int DATE_DIALOG_ID = 0;
    static final int DATE_DIALOG_ID2 = 1;

    ArrayList<String> users = new ArrayList<String>();

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.history);

        // capture our View elements
        mDateDisplay = (TextView) findViewById(R.id.StartTime);
        mPickDate = (Button) findViewById(R.id.pickBDate);
        mDateDisplay2 = (TextView) findViewById(R.id.EndTime);
        mPickDate2 = (Button) findViewById(R.id.pickEDate);


        String KEY_121 =
"http://ndnfinfo.ee.nd.edu/android/usernames.php";
         InputStream is = null;
         String result = "";
         ArrayList<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();
         nameValuePairs.add(new BasicNameValuePair("machine","RIE"));
         try{
                 HttpClient httpclient = new DefaultHttpClient();
                 HttpPost httppost = new HttpPost(KEY_121);
                 httppost.setEntity(new
UrlEncodedFormEntity(nameValuePairs));
                 HttpResponse response = httpclient.execute(httppost);
                 HttpEntity entity = response.getEntity();
                 is = entity.getContent();

         }catch(Exception e){
                 Log.e("log_tag", "Error in http connection
"+e.toString());
         }
         try{
                 BufferedReader reader = new BufferedReader(new
InputStreamReader(is,"iso-8859-1"),8);
                 StringBuilder sb = new StringBuilder();
                 String line = null;
                 while ((line = reader.readLine()) != null) {
                         sb.append(line + "\n");
                 }
                 is.close();
                 result=sb.toString();
         }catch(Exception e){
                 Log.e("log_tag", "Error converting result
"+e.toString());
         }
         //parse json data
         try{
                 JSONArray jArray = new JSONArray(result);
                 for(int i=0;i<jArray.length();i++){
                         JSONObject json_data =
jArray.getJSONObject(i);
                         //Get an output to the screen
                    users.add(json_data.getString("user"));
                 }
         }catch(JSONException e){
                 Log.e("log_tag", "Error parsing data "+e.toString());
         }
        AutoCompleteTextView textView = (AutoCompleteTextView)
findViewById(R.id.selectuser);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
R.layout.list_item, users);
        textView.setAdapter(adapter);



        // add a click listener to the button
        mPickDate.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                showDialog(DATE_DIALOG_ID);
            }
        });

        mPickDate2.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                showDialog(DATE_DIALOG_ID2);
            }
        });
        // get the current date
        final Calendar c = Calendar.getInstance();
        mYear = c.get(Calendar.YEAR);
        mMonth = c.get(Calendar.MONTH);
        mDay = c.get(Calendar.DAY_OF_MONTH);

        mYear2 = c.get(Calendar.YEAR);
        mMonth2= c.get(Calendar.MONTH);
        mDay2 = c.get(Calendar.DAY_OF_MONTH);

        // display the current date (this method is below)
        updateDisplay();

        Button button = (Button)this.findViewById(R.id.button1);
        button.setOnClickListener(new ButtonListener1());
    }

      public void onClick(DialogInterface arg0, int arg1) {
           // TODO Auto-generated method stub

      }
    private class ButtonListener1 implements View.OnClickListener{
      public void onClick(View v){
           AutoCompleteTextView select =  (AutoCompleteTextView)
findViewById(R.id.selectuser);
            String user = select.getText().toString();
               String delims = " ";
               String[] tokens = user.split(delims);
               String name = "";
               for (int i=0;i<tokens.length;i++)
                name += tokens[i];
         String stime = mYear + "-" + (mMonth+1) + "-" + mDay;
```

```
          String etime = mYear2 + "-" + (mMonth2+1) + "-" + mDay2;
          name = name + "!" +stime + "!" + etime;
               Intent myIntent = new Intent();

myIntent.setClassName("com.seniordesign","com.seniordesign.showhis");
                myIntent.putExtra("info",name);
                startActivity(myIntent);
     }
    }


    private void updateDisplay() {
        mDateDisplay.setText(
            new StringBuilder()
                    // Month is 0 based so add 1
                    .append(mMonth + 1).append("-")
                    .append(mDay).append("-")
                    .append(mYear).append(" "));
        mDateDisplay2.setText(
                new StringBuilder()
                        // Month is 0 based so add 1
                        .append(mMonth2 + 1).append("-")
                        .append(mDay2+1).append("-")
                        .append(mYear2).append(" "));
    }

    private DatePickerDialog.OnDateSetListener mDateSetListener =
        new DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker view, int year,
                                  int monthOfYear, int dayOfMonth) {
                mYear = year;
                mMonth = monthOfYear;
                mDay = dayOfMonth;
                updateDisplay();
            }
        };
    private DatePickerDialog.OnDateSetListener mDateSetListener2 =
            new DatePickerDialog.OnDateSetListener() {
                @Override
                public void onDateSet(DatePicker view, int year,
                                      int monthOfYear, int dayOfMonth)
{
                    mYear2 = year;
                    mMonth2 = monthOfYear;
                    mDay2 = dayOfMonth;
                    updateDisplay();
                }
            };

        @Override
        protected Dialog onCreateDialog(int id) {
            switch (id) {
            case DATE_DIALOG_ID:
                return new DatePickerDialog(this,
                           mDateSetListener,
                           mYear, mMonth, mDay);
            case DATE_DIALOG_ID2:
                 return new DatePickerDialog(this,
                              mDateSetListener2,
                              mYear2, mMonth2, mDay2);
            }
            return null;
        }
}
```

60. machines.java

```
package com.seniordesign;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.LinearLayout;
import android.widget.TextView;




public class machines extends Activity{
     TextView txt;
   public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     LinearLayout rootLayout = new
LinearLayout(getApplicationContext());
     txt = new TextView(getApplicationContext());
     rootLayout.addView(txt);
     setContentView(rootLayout);
     txt.setText("Connecting...");
     txt.setText(getServerData(KEY_121));
}


    public static final String KEY_121 =
"http://ndnfinfo.ee.nd.edu/android/machine_statuses.php";
    private String getServerData(String returnString) {

       InputStream is = null;

       String result = "";
       String output = "";
        ArrayList<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();
        nameValuePairs.add(new BasicNameValuePair("user","Matt"));
        try{
                HttpClient httpclient = new DefaultHttpClient();
                HttpPost httppost = new HttpPost(KEY_121);
                httppost.setEntity(new
UrlEncodedFormEntity(nameValuePairs));
                HttpResponse response = httpclient.execute(httppost);
                HttpEntity entity = response.getEntity();
                is = entity.getContent();

        }catch(Exception e){
                Log.e("log_tag", "Error in http connection
"+e.toString());
        }
        try{
                BufferedReader reader = new BufferedReader(new
InputStreamReader(is,"iso-8859-1"),8);
                StringBuilder sb = new StringBuilder();
                String line = null;
                while ((line = reader.readLine()) != null) {
                        sb.append(line + "\n");
                }
                is.close();
                result=sb.toString();
        }catch(Exception e){
                Log.e("log_tag", "Error converting result
"+e.toString());
        }
        //parse json data
        try{
                output += "Machines Currently Down:\n\n";
                JSONArray jArray = new JSONArray(result);
                for(int i=0;i<jArray.length();i++){
                        JSONObject json_data = 
jArray.getJSONObject(i);
                        //Get an output to the screen
                        output += json_data.getString("name")+"\n\n";
                }
        }catch(JSONException e){
                Log.e("log_tag", "Error parsing data "+e.toString());
        }
        return output;
   }
}
```

61. Main.java

```
package com.seniordesign;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.DialogInterface.OnClickListener;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class Main extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.startup);
    Button button = (Button)this.findViewById(R.id.button1);
    button.setOnClickListener(new ButtonListener1());
    }

      public void onClick(DialogInterface arg0, int arg1) {
            // TODO Auto-generated method stub

      }

      public static final String KEY_121 = 
"http://ndnfinfo.ee.nd.edu/passwords/checkpassword.php";
    private class ButtonListener1 implements View.OnClickListener{
      public void onClick(View v){
         Intent myIntent = new Intent();

myIntent.setClassName("com.seniordesign","com.seniordesign.mainpage");
         startActivity(myIntent);
          EditText name =  (EditText) findViewById(R.id.editText1);
          EditText pword =  (EditText) findViewById(R.id.editText2);
         String username = name.getText().toString();
          String password = pword.getText().toString();
          InputStream is = null;

          String result = "";
          ArrayList<NameValuePair> nameValuePairs = new 
ArrayList<NameValuePair>();
          nameValuePairs.add(new BasicNameValuePair("user","Matt"));
          try{
                      HttpClient httpclient = new DefaultHttpClient();
                      HttpPost httppost = new HttpPost(KEY_121);
                      httppost.setEntity(new 
UrlEncodedFormEntity(nameValuePairs));
                      HttpResponse response = 
httpclient.execute(httppost);
                      HttpEntity entity = response.getEntity();
                      is = entity.getContent();

              }catch(Exception e){
                      Log.e("log_tag", "Error in http connection 
"+e.toString());
              }
              try{
                      BufferedReader reader = new BufferedReader(new 
InputStreamReader(is,"iso-8859-1"),8);
                      StringBuilder sb = new StringBuilder();
                      String line = null;
                      while ((line = reader.readLine()) != null) {
                              sb.append(line + "\n");
                      }
                      is.close();
                      result=sb.toString();
```

```
             }catch(Exception e){
                     Log.e("log_tag", "Error converting result 
"+e.toString());
             }
             //parse json data
             try{
                     JSONArray jArray = new JSONArray(result);
                     for(int i=0;i<jArray.length();i++){
                        JSONObject json_data = 
jArray.getJSONObject(i);
                        String user = json_data.getString("user");
                        String paword = 
json_data.getString("password");

                        //if (username.equals(user) && 
password.equals(paword)){
                        //  Intent myIntent = new Intent();
                         // 
myIntent.setClassName("com.seniordesign","com.seniordesign.mainpage");
                         //  startActivity(myIntent);
                        //}
                     }
            }
             catch(JSONException e)
             {}
}
}
}
```

62. mainpage.java

```
package com.seniordesign;

import android.app.Activity;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.DialogInterface.OnClickListener;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class mainpage extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.homepage);
        Button button = (Button)this.findViewById(R.id.button1);
        button.setOnClickListener(new ButtonListener1());
        Button button2 = (Button)this.findViewById(R.id.button2);
        button2.setOnClickListener(new ButtonListener2());
        Button button3 = (Button)this.findViewById(R.id.button3);
        button3.setOnClickListener(new ButtonListener3());
        Button button4 = (Button)this.findViewById(R.id.button4);
        button4.setOnClickListener(new ButtonListener4());
    }
    private class ButtonListener1 implements View.OnClickListener{
     @Override
     public void onClick(View v){
          Intent myIntent = new Intent();

     myIntent.setClassName("com.seniordesign","com.seniordesign.connec
t");
          startActivity(myIntent);
     }
    }

    private class ButtonListener2 implements View.OnClickListener{
     @Override
     public void onClick(View v){
          Intent myIntent = new Intent();

     myIntent.setClassName("com.seniordesign","com.seniordesign.reserv
ations");
          startActivity(myIntent);
     }
    }

    private class ButtonListener3 implements View.OnClickListener{
     @Override
     public void onClick(View v){
          Intent myIntent = new Intent();

     myIntent.setClassName("com.seniordesign","com.seniordesign.histor
y");
          startActivity(myIntent);
     }
    }

    private class ButtonListener4 implements View.OnClickListener{
     @Override
     public void onClick(View v){
          Intent myIntent = new Intent();

     myIntent.setClassName("com.seniordesign","com.seniordesign.graphs
");
          startActivity(myIntent);
     }
    }
     @Override
     public void onClick(DialogInterface dialog, int which) {
          // TODO Auto-generated method stub

     }

}
```

63. rankings.java

```
package com.seniordesign;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.LinearLayout;
import android.widget.TextView;





public class rankings extends Activity{
     TextView txt;
   public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     LinearLayout rootLayout = new
LinearLayout(getApplicationContext());
     txt = new TextView(getApplicationContext());
     rootLayout.addView(txt);
     setContentView(rootLayout);
     txt.setText("Connecting...");
     txt.setText(getServerData(KEY_121));

}


    public static final String KEY_121 =
"http://ndnfinfo.ee.nd.edu/android/rankings.php";
    private String getServerData(String returnString) {

       InputStream is = null;

       String result = "";
       String output = "";
        ArrayList<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();
        nameValuePairs.add(new BasicNameValuePair("user","Matt"));
        try{
                HttpClient httpclient = new DefaultHttpClient();
                HttpPost httppost = new HttpPost(KEY_121);
                httppost.setEntity(new
UrlEncodedFormEntity(nameValuePairs));
                HttpResponse response = httpclient.execute(httppost);
                HttpEntity entity = response.getEntity();
                is = entity.getContent();

        }catch(Exception e){
                Log.e("log_tag", "Error in http connection
"+e.toString());
        }
        try{
                BufferedReader reader = new BufferedReader(new
InputStreamReader(is,"iso-8859-1"),8);
                StringBuilder sb = new StringBuilder();
                String line = null;
                while ((line = reader.readLine()) != null) {
                        sb.append(line + "\n");
                }
                is.close();
                result=sb.toString();
        }catch(Exception e){
                Log.e("log_tag", "Error converting result
"+e.toString());
        }
        //parse json data
        try{
                JSONArray jArray = new JSONArray(result);
                output+="Top Overall Users:\n\n";
                for(int i=0;i<jArray.length();i++){
                        JSONObject json_data =
jArray.getJSONObject(i);
                           output += "Ranking: " + (i+1) + "\n"
                           +"Name: "+json_data.getString("user")+"\n"+
                            "Total time (hours): " +
json_data.getString("time") + "\n\n";

                }
        }catch(JSONException e){
                Log.e("log_tag", "Error parsing data "+e.toString());
        }
        return output;
   }
}
```

64. reservations.java

```
package com.seniordesign;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Calendar;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.DialogInterface.OnClickListener;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;

public class reservations extends Activity implements OnClickListener
{
    /** Called when the activity is first created. */
    private TextView mDateDisplay;
    private Button mPickDate;
    private TextView mDateDisplay2;
    private TextView mPickDate2;
    private int mYear;
    private int mMonth;
    private int mDay;

    private int mYear2;
    private int mMonth2;
    private int mDay2;

    static final int DATE_DIALOG_ID = 0;
    static final int DATE_DIALOG_ID2 = 1;

    ArrayList<String> machines = new ArrayList<String>();

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.date_request);

        // capture our View elements
        mDateDisplay = (TextView) findViewById(R.id.StartTime);
        mPickDate = (Button) findViewById(R.id.pickBDate);

        mDateDisplay2 = (TextView) findViewById(R.id.EndTime);
        mPickDate2 = (Button) findViewById(R.id.pickEDate);


        String KEY_121 =
"http://ndnfinfo.ee.nd.edu/android/machinenames.php";
         InputStream is = null;
         String result = "";
         ArrayList<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();
         nameValuePairs.add(new BasicNameValuePair("machine","RIE"));
         try{
                 HttpClient httpclient = new DefaultHttpClient();
                 HttpPost httppost = new HttpPost(KEY_121);
                 httppost.setEntity(new
UrlEncodedFormEntity(nameValuePairs));
                 HttpResponse response = httpclient.execute(httppost);
                 HttpEntity entity = response.getEntity();
                 is = entity.getContent();
         }catch(Exception e){
                 Log.e("log_tag", "Error in http connection
"+e.toString());
         }
         try{
                 BufferedReader reader = new BufferedReader(new
InputStreamReader(is,"iso-8859-1"),8);
                 StringBuilder sb = new StringBuilder();
                 String line = null;
                 while ((line = reader.readLine()) != null) {
                         sb.append(line + "\n");
                 }
                 is.close();
                 result=sb.toString();
         }catch(Exception e){
                 Log.e("log_tag", "Error converting result
"+e.toString());
         }
         //parse json data
         try{
                 JSONArray jArray = new JSONArray(result);
                 for(int i=0;i<jArray.length();i++){
                         JSONObject json_data =
jArray.getJSONObject(i);
                         //Get an output to the screen
                   machines.add(json_data.getString("item"));
                 }
         }catch(JSONException e){
                 Log.e("log_tag", "Error parsing data "+e.toString());
         }


        AutoCompleteTextView textView = (AutoCompleteTextView)
findViewById(R.id.selectmach);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
R.layout.list_item, machines);
        textView.setAdapter(adapter);



        // add a click listener to the button
        mPickDate.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                showDialog(DATE_DIALOG_ID);
            }
        });

        mPickDate2.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                showDialog(DATE_DIALOG_ID2);
            }
        });
        // get the current date
        final Calendar c = Calendar.getInstance();
        mYear = c.get(Calendar.YEAR);
        mMonth = c.get(Calendar.MONTH);
        mDay = c.get(Calendar.DAY_OF_MONTH);

        mYear2 = c.get(Calendar.YEAR);
        mMonth2= c.get(Calendar.MONTH);
        mDay2 = c.get(Calendar.DAY_OF_MONTH);

        // display the current date (this method is below)
        updateDisplay();

        Button button = (Button)this.findViewById(R.id.button1);
        button.setOnClickListener(new ButtonListener1());
    }

     public void onClick(DialogInterface arg0, int arg1) {
           // TODO Auto-generated method stub

     }
    private class ButtonListener1 implements View.OnClickListener{
     public void onClick(View v){
           AutoCompleteTextView mach =  (AutoCompleteTextView)
findViewById(R.id.selectmach);
            String machine = mach.getText().toString();
         String stime = mYear + "-" + (mMonth+1) + "-" + mDay;
         String etime = mYear2 + "-" + (mMonth2+1) + "-" + mDay2;
         machine = machine + "!" +stime + "!" + etime;
              Intent myIntent = new Intent();

myIntent.setClassName("com.seniordesign","com.seniordesign.showres");
                myIntent.putExtra("info",machine);
                startActivity(myIntent);
     }
    }


    private void updateDisplay() {
        mDateDisplay.setText(
            new StringBuilder()
                    // Month is 0 based so add 1
                    .append(mMonth + 1).append("-")
                    .append(mDay).append("-")
                    .append(mYear).append(" "));
        mDateDisplay2.setText(
                new StringBuilder()
                        // Month is 0 based so add 1
                        .append(mMonth2 + 1).append("-")
                        .append(mDay2+1).append("-")
                        .append(mYear2).append(" "));
    }

    private DatePickerDialog.OnDateSetListener mDateSetListener =
        new DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker view, int year,
                                  int monthOfYear, int dayOfMonth) {
                mYear = year;
                mMonth = monthOfYear;
                mDay = dayOfMonth;
                updateDisplay();
            }
        };
    private DatePickerDialog.OnDateSetListener mDateSetListener2 =
            new DatePickerDialog.OnDateSetListener() {
                @Override
                public void onDateSet(DatePicker view, int year,
                                      int monthOfYear, int dayOfMonth)
{
                    mYear2 = year;
                    mMonth2 = monthOfYear;
                    mDay2 = dayOfMonth;
                    updateDisplay();
                }
            };

        @Override
        protected Dialog onCreateDialog(int id) {
            switch (id) {
            case DATE_DIALOG_ID:
                return new DatePickerDialog(this,
                            mDateSetListener,
                            mYear, mMonth, mDay);
            case DATE_DIALOG_ID2:
                 return new DatePickerDialog(this,
                              mDateSetListener2,
                              mYear2, mMonth2, mDay2);
            }
            return null;
        }
}
```

65. showhis.java

```
package com.seniordesign;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;

public class showhis extends Activity{
       TextView txt;
       @Override
       public void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           setContentView(R.layout.show_res);
           ScrollView scroller = new
ScrollView(getApplicationContext());
           LinearLayout rootLayout = new
LinearLayout(getApplicationContext());
           txt = new TextView(getApplicationContext());
           rootLayout.addView(txt);
           scroller.addView(rootLayout);


           setContentView(scroller);
            String info= getIntent().getExtras().getString("info");
            String delims = "!";
           String[] tokens = info.split(delims);
            String KEY_121 =
"http://ndnfinfo.ee.nd.edu/android/userhis.php?user=" +tokens[0]+
"&sdate=" +tokens[1] + "&edate=" + tokens[2];
         InputStream is = null;
              String result = "";
              String printable = "";
              ArrayList<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();
              nameValuePairs.add(new
BasicNameValuePair("machine","RIE"));
```

```
               try{
                       HttpClient httpclient = new
DefaultHttpClient();
                       HttpPost httppost = new HttpPost(KEY_121);
                       httppost.setEntity(new
UrlEncodedFormEntity(nameValuePairs));
                       HttpResponse response =
httpclient.execute(httppost);
                       HttpEntity entity = response.getEntity();
                       is = entity.getContent();

               }catch(Exception e){
                       Log.e("log_tag", "Error in http connection
"+e.toString());
               }
               try{
                       BufferedReader reader = new BufferedReader(new
InputStreamReader(is,"iso-8859-1"),8);
                       StringBuilder sb = new StringBuilder();
                       String line = null;
                       while ((line = reader.readLine()) != null) {
                               sb.append(line + "\n");
                       }
                       is.close();
                       result=sb.toString();
               }catch(Exception e){
                       Log.e("log_tag", "Error converting result
"+e.toString());
               }
               //parse json data
               try{
                       JSONArray jArray = new JSONArray(result);
                       for(int i=0;i<jArray.length();i++){
                               JSONObject json_data =
jArray.getJSONObject(i);
                               //Get an output to the screen
                        printable +=
                          "Start Time: "  + json_data.getString("tin")
+ "\n"
                          + "End Time: "  +
json_data.getString("tout") + "\n\n";
                       }
               }catch(JSONException e){
                       Log.e("log_tag", "Error parsing data
"+e.toString());
               }
               txt.setText(printable);
             }
          }
```

66. showres.java

```
package com.seniordesign;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;

public class showres extends Activity{
       TextView txt;
       @Override
       public void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           setContentView(R.layout.show_res);
           ScrollView scroller = new
ScrollView(getApplicationContext());
           LinearLayout rootLayout = new
LinearLayout(getApplicationContext());
           txt = new TextView(getApplicationContext());
           rootLayout.addView(txt);
           scroller.addView(rootLayout);


           setContentView(scroller);
             String info= getIntent().getExtras().getString("info");
             String delims = "!";
           String[] tokens = info.split(delims);
            String KEY_121 =
"http://ndnfinfo.ee.nd.edu/android/machres.php?mach=" +tokens[0]+
"&sdate=" +tokens[1] + "&edate=" + tokens[2];
         InputStream is = null;
               String result = "";
               String printable = "";
               ArrayList<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();
               nameValuePairs.add(new
BasicNameValuePair("machine","RIE"));
               try{
                       HttpClient httpclient = new
DefaultHttpClient();
                       HttpPost httppost = new HttpPost(KEY_121);
                       httppost.setEntity(new
UrlEncodedFormEntity(nameValuePairs));
                       HttpResponse response =
httpclient.execute(httppost);
                       HttpEntity entity = response.getEntity();
                       is = entity.getContent();

               }catch(Exception e){
                       Log.e("log_tag", "Error in http connection
"+e.toString());
               }
               try{
                       BufferedReader reader = new BufferedReader(new
InputStreamReader(is,"iso-8859-1"),8);
                       StringBuilder sb = new StringBuilder();
                       String line = null;
                       while ((line = reader.readLine()) != null) {
                               sb.append(line + "\n");
                       }
                       is.close();
                       result=sb.toString();
               }catch(Exception e){
                       Log.e("log_tag", "Error converting result
"+e.toString());
               }
               //parse json data
               try{
                       JSONArray jArray = new JSONArray(result);
                       for(int i=0;i<jArray.length();i++){
                               JSONObject json_data =
jArray.getJSONObject(i);
                               //Get an output to the screen
                        printable = printable + "Agent:
"+json_data.getString("agent") + "\n"
                          + "Start Time: "  +
json_data.getString("bdate") + "\n"
                          + "End Time: "  +
json_data.getString("edate") + "\n\n";
                       }
               }catch(JSONException e){
                       Log.e("log_tag", "Error parsing data
"+e.toString());
               }
               txt.setText(printable);
```

```
             }
          }
```

67. users.java

```
package com.seniordesign;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.LinearLayout;
import android.widget.TextView;





public class users extends Activity{
     TextView txt;
   public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     LinearLayout rootLayout = new
LinearLayout(getApplicationContext());
     txt = new TextView(getApplicationContext());
     rootLayout.addView(txt);
     setContentView(rootLayout);
     txt.setText(getServerData(KEY_121));

}
```

```
    public static final String KEY_121 =
"http://ndnfinfo.ee.nd.edu/android/users.php";
    private String getServerData(String returnString) {

       InputStream is = null;

       String result = "";
       String output = "";
        ArrayList<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();
        nameValuePairs.add(new BasicNameValuePair("user","Matt"));
        try{
                HttpClient httpclient = new DefaultHttpClient();
                HttpPost httppost = new HttpPost(KEY_121);
                httppost.setEntity(new
UrlEncodedFormEntity(nameValuePairs));
                HttpResponse response = httpclient.execute(httppost);
                HttpEntity entity = response.getEntity();
                is = entity.getContent();

        }catch(Exception e){
                Log.e("log_tag", "Error in http connection
"+e.toString());
        }
        try{
                BufferedReader reader = new BufferedReader(new
InputStreamReader(is,"iso-8859-1"),8);
                StringBuilder sb = new StringBuilder();
                String line = null;
                while ((line = reader.readLine()) != null) {
                        sb.append(line + "\n");
                }
                is.close();
                result=sb.toString();
        }catch(Exception e){
                Log.e("log_tag", "Error converting result
"+e.toString());
        }
        //parse json data
        try{
                JSONArray jArray = new JSONArray(result);
                for(int i=0;i<jArray.length();i++){
                        JSONObject json_data =
jArray.getJSONObject(i);
                        //Get an output to the screen
                        output += "Name:
"+json_data.getString("user")+"\n"+
                        "Time in: "+json_data.getString("tin")+"\n\n";
                }
        }catch(JSONException e){
                Log.e("log_tag", "Error parsing data "+e.toString());
        }
        return output;
    }
}
```

res/layout

68. date_request.xml

```
<?xml version="1.0" encoding="utf-8" ?>
- <RelativeLayout android:id="@+id/widget43" android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
  <TextView android:id="@+id/Start" android:textSize="18sp"
    android:layout_centerVertical="true" android:textStyle="italic" android:text="Start Time:
    \n\n" android:shadowRadius="1.5" android:shadowDx="1" android:shadowDy="1"
    android:shadowColor="#00ccff" android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
  <TextView android:id="@+id/machine" android:textSize="18sp"
    android:layout_above="@+id/Start" android:textStyle="italic" android:text="Machine:
    \n\n" android:shadowRadius="1.5" android:shadowDx="1" android:shadowDy="1"
    android:shadowColor="#00ccff" android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
  <TextView android:id="@+id/End" android:textSize="18sp"
    android:layout_below="@+id/Start" android:textStyle="italic" android:text="End Time:"
    android:shadowRadius="1.5" android:shadowDx="1" android:shadowDy="1"
    android:shadowColor="#00ccff" android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
  <TextView android:id="@+id/StartTime" android:textSize="18sp"
    android:layout_toRightOf="@+id/Start" android:layout_alignBaseline="@+id/Start"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:text="04-01-2011" />
  <TextView android:id="@+id/EndTime" android:textSize="18sp"
    android:layout_toRightOf="@+id/End" android:layout_alignBaseline="@+id/End"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:text="04-04-2011" />
  <AutoCompleteTextView android:id="@+id/selectmach" android:textSize="18sp"
    android:layout_toRightOf="@+id/machine"
    android:layout_alignBaseline="@+id/machine" android:layout_width="150sp"
    android:layout_height="wrap_content" android:singleLine="true" />
```

```
<Button android:id="@+id/pickBDate" android:textSize="14sp"
   android:layout_alignParentRight="true" android:layout_alignBaseline="@+id/StartTime"
   android:layout_width="100sp" android:layout_height="wrap_content"
   android:singleLine="true" android:text="Begin Date" />

<Button android:id="@+id/pickEDate" android:textSize="14sp"
   android:layout_alignParentRight="true" android:layout_alignBaseline="@+id/EndTime"
   android:layout_width="100sp" android:layout_height="wrap_content"
   android:singleLine="true" android:text="End Date" />

<TextView android:id="@+id/title" android:layout_centerHorizontal="true"
   android:textSize="18sp" android:layout_height="wrap_content" android:text="Check
   Reservations" android:shadowRadius="1.5" android:shadowDx="1"
   android:shadowDy="1" android:shadowColor="#00ccff"
   android:layout_width="wrap_content" />

<Button android:id="@+id/button1" android:layout_centerHorizontal="true"
   android:text="Submit" android:textSize="18sp" android:layout_height="wrap_content"
   android:layout_width="wrap_content" android:layout_alignParentBottom="true" />

  </RelativeLayout>
```

69. history.xml

```
 <?xml version="1.0" encoding="utf-8" ?>

- <RelativeLayout android:id="@+id/widget43" android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android">

<TextView android:id="@+id/Start" android:textSize="18sp"
   android:layout_centerVertical="true" android:textStyle="italic" android:text="Start Time:
   \n\n" android:shadowRadius="1.5" android:shadowDx="1" android:shadowDy="1"
   android:shadowColor="#00ccff" android:layout_width="wrap_content"
   android:layout_height="wrap_content" />

<TextView android:id="@+id/user" android:textSize="18sp"
   android:layout_above="@+id/Start" android:textStyle="italic" android:text="User's
   Name: \n\n" android:shadowRadius="1.5" android:shadowDx="1"
   android:shadowDy="1" android:shadowColor="#00ccff"
   android:layout_width="wrap_content" android:layout_height="wrap_content" />

<TextView android:id="@+id/End" android:textSize="18sp"
   android:layout_below="@+id/Start" android:textStyle="italic" android:text="End Time:"
   android:shadowRadius="1.5" android:shadowDx="1" android:shadowDy="1"
   android:shadowColor="#00ccff" android:layout_width="wrap_content"
   android:layout_height="wrap_content" />
```

```
<TextView android:id="@+id/StartTime" android:textSize="18sp"
   android:layout_toRightOf="@+id/Start" android:layout_alignBaseline="@+id/Start"
   android:layout_width="wrap_content" android:layout_height="wrap_content"
   android:text="04-01-2011" />

<TextView android:id="@+id/EndTime" android:textSize="18sp"
   android:layout_toRightOf="@+id/End" android:layout_alignBaseline="@+id/End"
   android:layout_width="wrap_content" android:layout_height="wrap_content"
   android:text="04-04-2011" />

<AutoCompleteTextView android:id="@+id/selectuser" android:textSize="18sp"
   android:layout_toRightOf="@+id/user" android:layout_alignBaseline="@+id/user"
   android:layout_width="150sp" android:layout_height="wrap_content"
   android:singleLine="true" />

<Button android:id="@+id/pickBDate" android:textSize="14sp"
   android:layout_alignParentRight="true" android:layout_alignBaseline="@+id/StartTime"
   android:layout_width="100sp" android:layout_height="wrap_content"
   android:singleLine="true" android:text="Begin Date" />

<Button android:id="@+id/pickEDate" android:textSize="14sp"
   android:layout_alignParentRight="true" android:layout_alignBaseline="@+id/EndTime"
   android:layout_width="100sp" android:layout_height="wrap_content"
   android:singleLine="true" android:text="End Date" />

<TextView android:id="@+id/title" android:layout_centerHorizontal="true"
   android:textSize="18sp" android:layout_height="wrap_content" android:text="View
   Session History" android:shadowRadius="1.5" android:shadowDx="1"
   android:shadowDy="1" android:shadowColor="#00ccff"
   android:layout_width="wrap_content" />

<Button android:id="@+id/button1" android:layout_centerHorizontal="true"
   android:text="Submit" android:textSize="18sp" android:layout_height="wrap_content"
   android:layout_width="wrap_content" android:layout_alignParentBottom="true" />

  </RelativeLayout>
```

70. homepage.xml

```
  <?xml version="1.0" encoding="utf-8" ?>

- <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="@drawable/appbackground">
```

```
<TextView android:layout_width="fill_parent" android:layout_height="wrap_content"
   android:text="Hello, Lab User!" />

<TextView android:id="@+id/textView1" android:layout_height="wrap_content"
   android:layout_centerHorizontal="true" android:text="NDNFINFO\nHomepage"
   android:shadowRadius="1.5" android:shadowDx="1" android:shadowDy="1"
   android:textColor="#4169e1" android:shadowColor="#000000"
   android:textStyle="bold" android:textSize="30sp" android:layout_width="wrap_content"
   />

<Button android:id="@+id/button1" android:layout_height="wrap_content"
   android:text="View Top Users/Machine Statuses" android:layout_width="150sp"
   android:layout_above="@+id/button2" android:layout_centerHorizontal="true" />

<Button android:id="@+id/button2" android:layout_height="wrap_content"
   android:text="View Machine Reservations" android:layout_width="150sp"
   android:layout_centerHorizontal="true" android:layout_centerVertical="true" />

<Button android:id="@+id/button3" android:layout_height="wrap_content"
   android:text="View Session History" android:layout_width="150sp"
   android:layout_below="@+id/button2" android:layout_centerHorizontal="true" />

<Button android:id="@+id/button4" android:layout_height="wrap_content"
   android:text="Graphs" android:layout_width="150sp"
   android:layout_below="@+id/button3" android:layout_centerHorizontal="true" />

  </RelativeLayout>
```

71. list_item.xml

```
<?xml version="1.0" encoding="utf-8" ?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent" android:layout_height="fill_parent"
   android:padding="10dp" android:textSize="16sp" android:textColor="#000" />
```

72. main.xml

```
<?xml version="1.0" encoding="utf-8" ?>

- <TabHost xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@android:id/tabhost" android:layout_width="fill_parent"
   android:layout_height="fill_parent">
```

```
- <ScrollView android:id="@+id/ScrollView01" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
- <LinearLayout android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:padding="5dp">
  <TabWidget android:id="@android:id/tabs" android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
  <FrameLayout android:id="@android:id/tabcontent" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:padding="5dp" />
  </LinearLayout>
  </ScrollView>
  </TabHost>
```

73.main2.xml

```
  <?xml version="1.0" encoding="utf-8" ?>
- <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:stretchColumns="1">
- <TableRow>
  <TextView android:layout_column="1" android:text="Open..." android:padding="3dip" />
  <TextView android:text="Ctrl-O" android:gravity="right" android:padding="3dip" />
  </TableRow>
- <TableRow>
  <TextView android:layout_column="1" android:text="Save..." android:padding="3dip" />
  <TextView android:text="Ctrl-S" android:gravity="right" android:padding="3dip" />
  </TableRow>
- <TableRow>
  <TextView android:layout_column="1" android:text="Save As..." android:padding="3dip" />
  <TextView android:text="Ctrl-Shift-S" android:gravity="right" android:padding="3dip" />
  </TableRow>
  <View android:layout_height="2dip" android:background="#FF909090" />
- <TableRow>
  <TextView android:text="X" android:padding="3dip" />
  <TextView android:text="Import..." android:padding="3dip" />
    </TableRow>
- <TableRow>
  <TextView android:text="X" android:padding="3dip" />
  <TextView android:text="Export..." android:padding="3dip" />
  <TextView android:text="Ctrl-E" android:gravity="right" android:padding="3dip" />
    </TableRow>
  <View android:layout_height="2dip" android:background="#FF909090" />
- <TableRow>
  <TextView android:layout_column="1" android:text="Quit" android:padding="3dip" />
    </TableRow>
    </TableLayout>
```

74.show_res.xml

```
    <?xml version="1.0" encoding="utf-8" ?>
- <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical" android:layout_width="fill_parent"
     android:layout_height="fill_parent">
- <ScrollView android:orientation="vertical" android:layout_width="fill_parent"
     android:layout_height="fill_parent">
  <TextView android:layout_width="fill_parent" android:layout_height="fill_parent"
     android:text="@string/hello" />
    </ScrollView>
    </LinearLayout>
```

75.startup.xml

```
  <?xml version="1.0" encoding="utf-8" ?>
- <RelativeLayout android:id="@+id/widget43" android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/apploginbackground">
  <TextView android:id="@+id/textView1" android:textSize="24sp"
    android:layout_centerVertical="true" android:textStyle="italic" android:text="Username:
    \n\n" android:shadowRadius="1.5" android:shadowDx="1" android:shadowDy="1"
    android:textColor="#4169e1" android:shadowColor="#000000"
    android:layout_width="wrap_content" android:layout_height="wrap_content" />
  <TextView android:id="@+id/textView2" android:textSize="24sp"
    android:layout_below="@+id/textView1" android:textStyle="italic"
    android:text="Password: \n" android:shadowRadius="1.5" android:shadowDx="1"
    android:shadowDy="1" android:textColor="#4169e1" android:shadowColor="#000000"
    android:layout_width="200sp" android:layout_height="wrap_content" />
  <EditText android:id="@+id/editText1" android:textSize="24sp"
    android:layout_toRightOf="@+id/textView1"
    android:layout_alignBaseline="@+id/textView1" android:layout_width="150sp"
    android:layout_height="wrap_content" android:singleLine="true" />
  <EditText android:id="@+id/editText2" android:textSize="24sp"
    android:layout_toRightOf="@+id/textView2"
    android:layout_alignBaseline="@+id/textView2"
    android:layout_alignLeft="@+id/editText1" android:layout_width="150sp"
    android:layout_height="wrap_content" android:singleLine="true"
    android:password="true" />
  <TextView android:id="@+id/textView3" android:layout_centerHorizontal="true"
    android:textSize="30sp" android:layout_height="wrap_content" android:text="Ndnfinfo
    Login" android:shadowRadius="1.5" android:shadowDx="1" android:shadowDy="1"
    android:textColor="#4169e1" android:shadowColor="#000000"
    android:textStyle="bold" android:layout_width="wrap_content" />
  <Button android:id="@+id/button1" android:layout_centerHorizontal="true"
    android:text="Submit" android:textSize="25sp" android:layout_height="wrap_content"
    android:layout_width="wrap_content" android:layout_below="@+id/textView2" />
  </RelativeLayout>
```

76.webview.xml

```xml
 <?xml version="1.0" encoding="utf-8" ?>
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/webview" android:layout_width="fill_parent"
   android:layout_height="fill_parent" />
```

iPhone Source Code

78. WebAppDelegate.h

```
#import <UIKit/UIKit.h>

@interface WebAppDelegate : NSObject <UIApplicationDelegate> {
    UIWindow *window;
      UIWebView *webView;
}

@property (nonatomic, retain) IBOutlet UIWindow *window;
@property (nonatomic, retain) IBOutlet UIWebView *webView;

@end
```

79. WebAppDelegate.m

```
#import "WebAppDelegate.h"

@implementation WebAppDelegate

@synthesize window;
@synthesize webView;


#pragma mark -
#pragma mark Application lifecycle

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [webView loadRequest:[NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://ndnfinfo.ee.nd.edu/iphone"]]];
    // Override point for customization after application launch.

    [self.window makeKeyAndVisible];

    return YES;
}


- (void)applicationWillResignActive:(UIApplication *)application {
    /*
     Sent when the application is about to move from active to
inactive state. This can occur for certain types of temporary
interruptions (such as an incoming phone call or SMS message) or when
the user quits the application and it begins the transition to the
background state.
```

```
     Use this method to pause ongoing tasks, disable timers, and
throttle down OpenGL ES frame rates. Games should use this method to
pause the game.
     */
}


- (void)applicationDidEnterBackground:(UIApplication *)application {
    /*
     Use this method to release shared resources, save user data,
invalidate timers, and store enough application state information to
restore your application to its current state in case it is terminated
later.
     If your application supports background execution, called instead
of applicationWillTerminate: when the user quits.
     */
}


- (void)applicationWillEnterForeground:(UIApplication *)application {
    /*
     Called as part of  transition from the background to the inactive
state: here you can undo many of the changes made on entering the
background.
     */
}


- (void)applicationDidBecomeActive:(UIApplication *)application {
    /*
     Restart any tasks that were paused (or not yet started) while the
application was inactive. If the application was previously in the
background, optionally refresh the user interface.
     */
}


- (void)applicationWillTerminate:(UIApplication *)application {
    /*
     Called when the application is about to terminate.
     See also applicationDidEnterBackground:.
     */
}


#pragma mark -
#pragma mark Memory management

- (void)applicationDidReceiveMemoryWarning:(UIApplication
*)application {
    /*
```

```
     Free up as much memory as possible by purging cached data objects
that can be recreated (or reloaded from disk) later.
     */
}


- (void)dealloc {
    [window release];
    [super dealloc];
}


@end
```

80. main.m

```
#import <UIKit/UIKit.h>

int main(int argc, char *argv[]) {

    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    int retVal = UIApplicationMain(argc, argv, nil, nil);
    [pool release];
    return retVal;
}
```