Zacharia Fadika
CIS 480 Capstone project.
Professor: Dr. Timothy Fossum

S.U.N.Y. POTSDAM
SPRING 2006

ROOM SCHEDULING APPLICATION

By Zacharia Fadika

Table of Contents

ABSTRACT

The room scheduling interface was written as a means to allow room managers or simply individuals having rooms at their disposal to store and manage information concerning their occupancy without having to deal with standard database access, paperwork management, manual data storage or mechanical data entry and updating. Institutions of education, businesses and conference centers use rooms to conduct their activities, but more than often rely on a volatile piece of paper posted outside of a given room as an indicator of its schedule. The room scheduler provides a much more effective and robust way for room management. The program is endowed with an easy to use, user friendly interface with intuitive features that allow for speed and data integrity. The application can store a multitude of rooms, retrieve and update them with just a few mouse clicks. Room schedules are saved in an orderly manner and can be either viewed, printed out or deleted when no longer needed. The room scheduler uses a web interface making it accessible from anywhere one can have a connection to the internet and a regular browser. The ability for the scheduler to be available online makes the application manageable from any location, and at any time of the day or the night. It can be used by independent managers, or as a centralized system used by departments or offices within the same institution. This last option would make the application widely available and thus vulnerable. To protect the integrity of the data and of the operations performed on the system, a log in process with session management of account information was integrated into the program. The system is fronted by a portal that prevents unauthorized or unwanted access by prompting a given user for a login username and a password. User access levels and credentials are available for that purpose and allow for the regulation of the system by managing its users. User management is completely automated and does not require any technical knowledge of database operations and password encryption methods. It is a functionality that is built into the application and that only allows privileged account holders or administrator users to view or remove or grant access to the application. This protection measure allows for the safeguarding of the stored information, and the control of the operations performed on the system.

INTRODUCTION

The room schedule interface is a web application that allows users to enter a week schedule for rooms, and to have the ability to view, or delete a previously entered schedule. After being entered, a schedule can be viewed according to the current date, or viewed for a particular date inputted by the user. For the later to be possible, the program checks the current date against the academic calendar to determine school recesses .If a schedule is being requested during a week that in which a recess will start, the concerned days will be marked as break days, and their schedule will not be displayed. Similarly, if a date which corresponds to a recess day is entered, the days displayed are shown as break days. The application should be widely available on the World Wide Web, and as such is password protected. An administrative side was developed and allows for user levels, and users with admin access level to manage account creating and deletion, and thus define which users have access to the application and to what extent their access allows them to perform diverse operations. The interface was coded using HTML and PHP and SQL, and can be hosted on any system supporting the above standards within the software requirement boundaries.

I. SPECIFICATIONS

The capstone project sought to be done will consist of a room scheduling or personal scheduling multi-layered web application. The project will be done in PHP/HTML/MYSQL, and allow users to enter room identifiers, and schedules for those rooms from Monday to Saturday. The room schedules will comprise the time and matter taught for each day of the week in a particular room. The room identifiers will be available on a web page that shall be generated as rooms are created or deleted.

The data entered via the web page will be stored in a database, allowing it to be retrieved, modified, or if wanted deleted. Upon creating a room, the web page should dynamically generate subsequent web pages including the newly entered room identifier, so the user can have an updated list of available rooms and their schedules to perform operations on. Similarly, upon deleting a room, the room erased from the database should instantly disappear from the web page containing the list of available rooms. The web page will also display room schedules according to the current academic calendar, meaning that if a room schedule where to be requested during spring break for instance, the web application should clearly display the days on which classes will be out of session instead of the subject taught on that particular day in the room selected. In the event of an upcoming break, let's say for instance a break starting on a Tuesday and ending on the following Thursday, if a room schedule were to be requested on the Monday of the same week, the application should display the normal schedule for Monday and Friday, but instead of doing the same for Tuesday, Wednesday and Thursday, it should instead display in the column characterizing those days the relevant on going break, as to inform the user. The web page will also give the user the possibility to query it for a future date. Users will be given the opportunity to enter a date, past or future, and view the room schedule for that date. The user will also have the option to modify or update a schedule instead of erasing it and remaking it. The user will also have the option to print room schedules in a printer friendly format. Since the application could serve as a live database used to manage room schedules at SUNY POTSDAM or any other institution, it will be available on the World Wide Web, and as such should be protected from unauthorized access. A log in screen will be available

allowing for verification and authentication of users before giving access to the room scheduling operations. For that matter, a default administrative account will be built into the application, allowing for a first login, and also for creation, modification or deletion of accounts, including the default account itself. The project is planned to be realized using SQL, PHP, HTML, and will necessitate the installation of a Server operating system with a Web daemon, a MYSQL daemon and a PHP daemon configured and running on it.

II. REQUIREMENTS

The room scheduling application is a web application, and as such, it should be available on a web server, giving users the ability to access it from anywhere they can have a browser running and an internet connection. For this purpose, a web server needs to be running on the host computer to be able to render the HTML used to make the interface. The project was programmed in HTML v 1.1. The application also makes use of Hypertext preprocessor (PHP), so a PHP daemon should also be available on the server, and should be able to perform message passing with the web server as the PHP daemon would generate HTML pages, and HTML pages would send their content to the PHP daemon. The PHP module should be version 4.3 or greater. For this project, we also used a database system for storing information, more precisely room schedules and account information. MySQL server v4.4 was used as the database server, and the Structured Query Language (SQL) as the language used to perform the required operations by system to store, access and retrieve the needed information. The Database server is accessed from the web by the PHP daemon, so the PHP server and the MySQL module should be able to work in concert. For this to be possible, the web server must be installed with PHP functionality, and the installation parameter of the PHP module for the web server should contain SQL extensions inclusion. In an effort to save memory, one could restrict the SQL module extension installation to the database operation module without the administrative module and the development. However, the required component should be installed with the optional debugging module for PHP and SQL, as the system performs error checking on itself and adjusts certain parts of the PHP, especially variable sizes for memory leak avoidance during runtime. On the system should also be installed a C compiler or simply a C module for PHP. Even though this last step could be avoided, following it would greatly increase the overall speed of certain demanding functions. To finish, as this should be accessible at anytime of the day or the night, the server should be able to run continuously, and should have a static IP address, or a domain name, and for certain server systems running firewalls, the relevant high-ports should be fingerprinted, and opened, as the daemons use a certain group of ports in the thousand range.

III. MODUS OPERANDI

Introduction

The application works by a simple process of data exchange and creation between PHP, HTML and MySQL. For the account creating section on the administrative side of the application, only the authorized users (Administrators) gave access to the available functionalities. To be authenticated, a user needs to enter his or her credentials on the log in page. At that level, the HTML server takes the entered information and sends it to the PHP daemon which in turns connects to the MySQL database, and checks the information obtained from the HTML server against the information stored in the database, precisely in the accounts table. It is worthwhile to note that the accounts table contains usernames, passwords and access levels of authorized users. The usernames are stored in plain text, the access levels are designated by flags, and the passwords are stored encrypted, so the PHP daemon needs to encrypt the given password prior to comparing with the hash stored in the table. The encryption algorithm used in this case was MD5 with randomly generated keys. The randomization algorithm was written for this purpose so it wouldn't match any algorithm publicly available. Once the comparison returns a successful match, the access level is retrieved and checked to ensure that the authenticated user has administrative privileges. If the test is negative, the user is informed of his or her lack of credentials to access the administrative panel and returned to the log in page. If the test is positive, the user is logged into the panel; a three minute session is created in PHP, and the user is given access to the operations page. Whenever the user performs an operation, the session validity is checked, and if no longer valid, the user is sent back to the log in page, where they are required to identify themselves. The same process takes place whenever a user logs out, as a logout link is available on almost every link. In that case, the session is "killed", and as such no longer valid. Once on the operation page, the user has three options: The first is to create accounts for future users, view the current accounts in the system, or simply remove accounts. When creating accounts, the user has the option to determine which type of access a user has, and as such create other administrators, who would also in turn be able to create accounts. This allows for users to regulate who has and

who does have access to the system.

Once user accounts are created, authorized users (users and administrators) can freely access the scheduling page. The scheduling page, like the administrative page requires log in information to allow a user to perform operations. In this particular case, both administrators and users can log into the system and use the application. The authentication phase is similar to the previous process (For the administration side), to the only difference that the access level is not checked, since both types of users can access the system. A longer session is however created as managing schedules can be more time consuming than creating accounts. The user's session is valid for six minutes in this case. Much like on the administrative side, a logout function accesses the session information on the client and server side, and "kills" it, thus making it invalid.

Once in the system, a user has three options. Schedule creation, schedule deletion and schedule consultation.

1. Schedule creation

Creating an account on the system is nothing less than entering the desired information in a HTML form. Once a user is granted access to the system, the user is presented with a page displaying the three options aforementioned. Upon clicking on the account creation alternative, the user is sent a PHP form containing a schedule skeleton, and a room identifier box. To enter a schedule, a user needs to enter the room number first, and then fill out the matter taught along with the times at which the classes are taught using input and drop down boxes. Upon submission, the form is sent to a second PHP (hidden) form, whose role is to contact the MySQL database in order to store the information entered. The PHP module has an account on the MySQL system, and used its credentials to log into the system; it then accesses the room table, stores the room identifier in it, and then accesses the schedule table to store the schedule in it. It is worthwhile to note that as an entry is store in the room table, a self generated number is issued, and identifies the entry.

This feature known as a primary key makes the entry unique, as two entries cannot have the same identification number. The id number is a simple integer, which starts at 0, and is incremented as entries are added. Should any of the above operations fail (Database access, table access), the user is automatically notified. Once the operations are successfully completed, a third PHP page is sent and displays a message indicating the status of the action.

1. Schedule consultation

Consulting an account operates in a different manner than creating an account. Upon clicking on the option, the user is sent to a PHP page that automatically accesses the database and the indexed table to retrieve the available rooms. The PHP module accesses the room table store and writes the HTML code accordingly, so the user can have the option to select the one he or she wants to view. Once the choice is made, the data is sent from the HTML generated page back to the PHP module which goes back to the database, but this time to access a different table using the id number gotten from room table. The PHP module accesses the room table, uses the room identifier to retrieve the primary key, and then switches to the room schedule table to pull all of the rows containing the same key. The information is then passed back to the PHP module which is responsible for assessing it, and displaying it accordingly.

a. Schedule consultation by date

When consulting a room schedule, the user also has the option to view the displayed schedule for a different date, be it past or future. The reason for this feature is simply so for a meeting or a one time event, a room manager might need a room on a weekday, but at a certain date on which school might not be in session. The room manager would need to check the academic calendar, then a regular calendar, and compare the dates. Giving the ability to the user to query the system for such date is a means to ensure that the user gets the same output accurately and fast. For this purpose, once

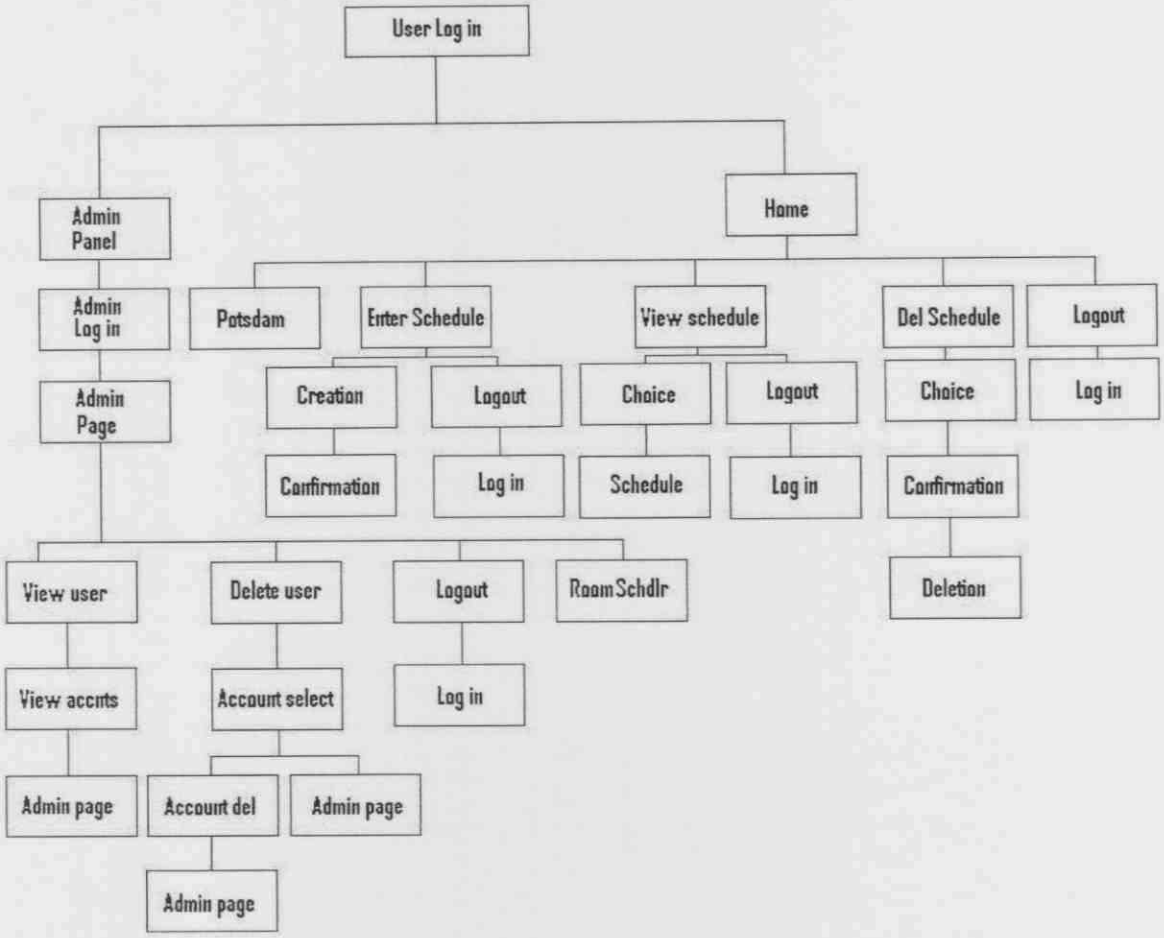
a schedule is displayed, it also comes with an input box for a date. As soon as the user enters the date, the information is sent to the PHP module that accesses the calendar table which is synchronized with the academic calendar. The PHP application passes the baton to a second PHP application whose role is to check the current calendar, retrieve the current date, and make the comparison. The second application also locates the relevant days in the current week and overwrites their content with the information pertinent to the current break. The result is then generated in HTML by the second PHP application and displayed.

2. Schedule deletion

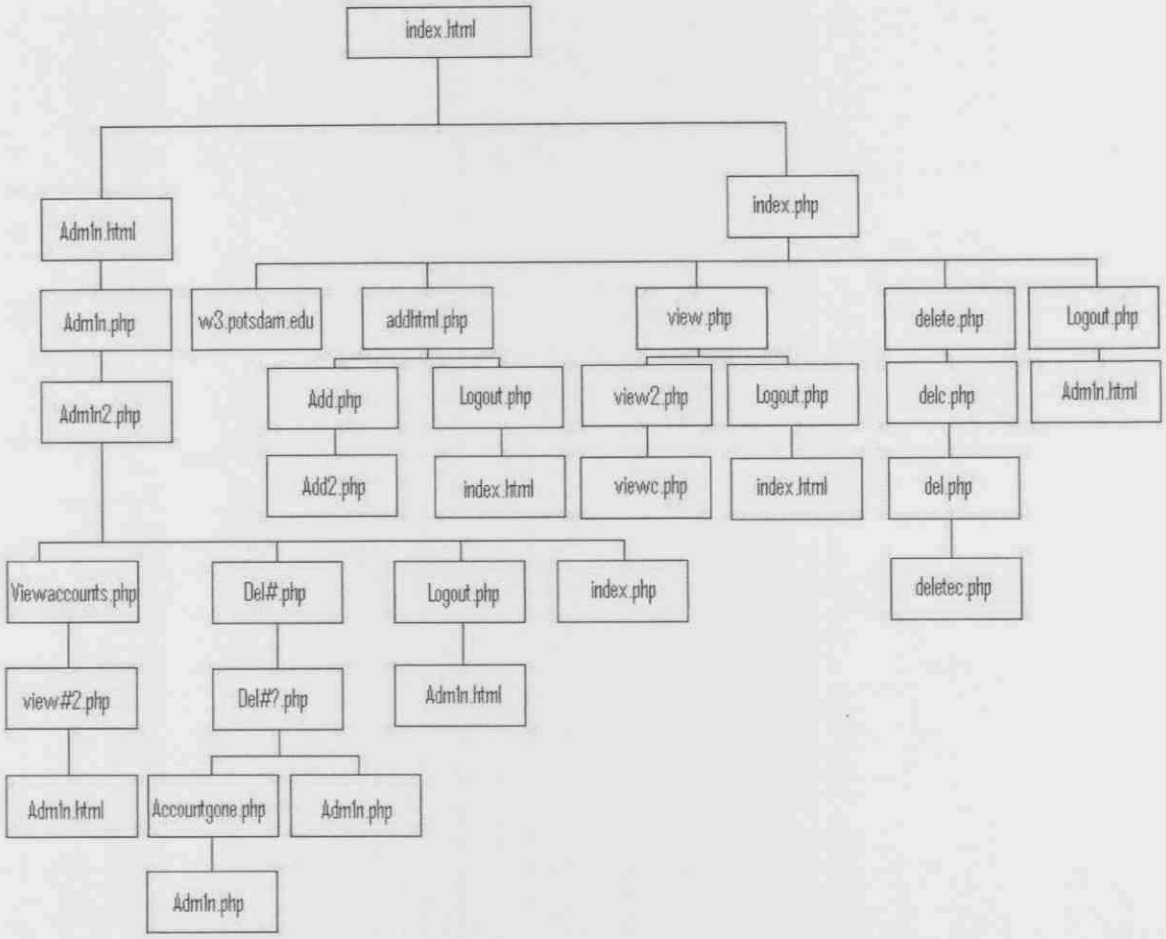
Schedule deletion starts much like schedule consultation in that, a PHP page is automatically sent to the user. That page goes directly to the database and retrieves information from the room table. The information is then displayed using HTML. From that page, the user makes a choice. Upon choosing from the form, the information resulting from the choice is sent back to the database. The room table is accessed using the room identifier, the primary key saved. The user is then sent to a PHP page asking to confirm the deletion. On the same page, the user is also given the option to cancel the deletion. Should the user accept the deletion, the row indexed in the room table is removed, and the key stored used to access the room schedule table and remove all the rows containing that key. The user is then sent to a deletion confirmation page, to indicate that the schedule was successfully deleted.

IV. DESIGN & DEVELOPEMENT

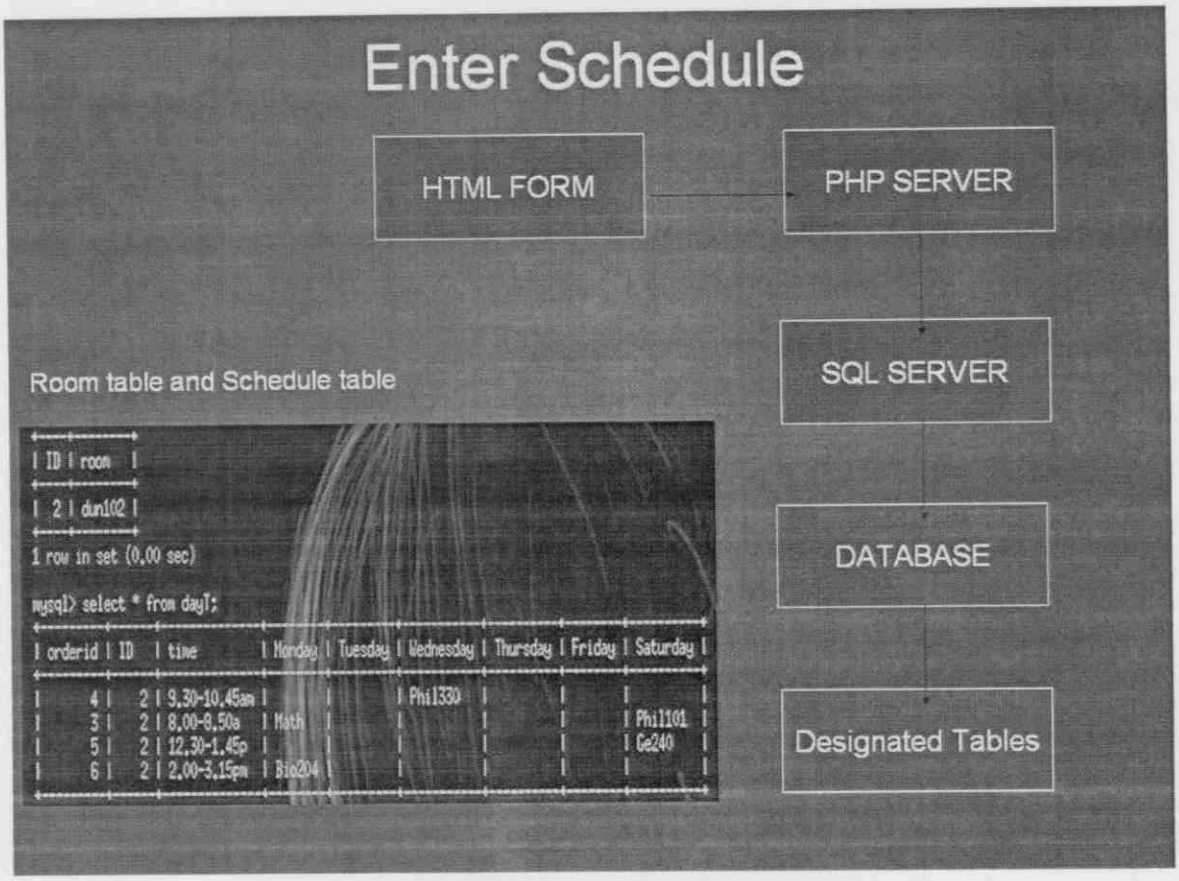
The project was designed in a hierarchical manner, using levels of designs; PHP and HTML pages being set on different levels of design. The design model corresponds to the web development linear model outlined below.



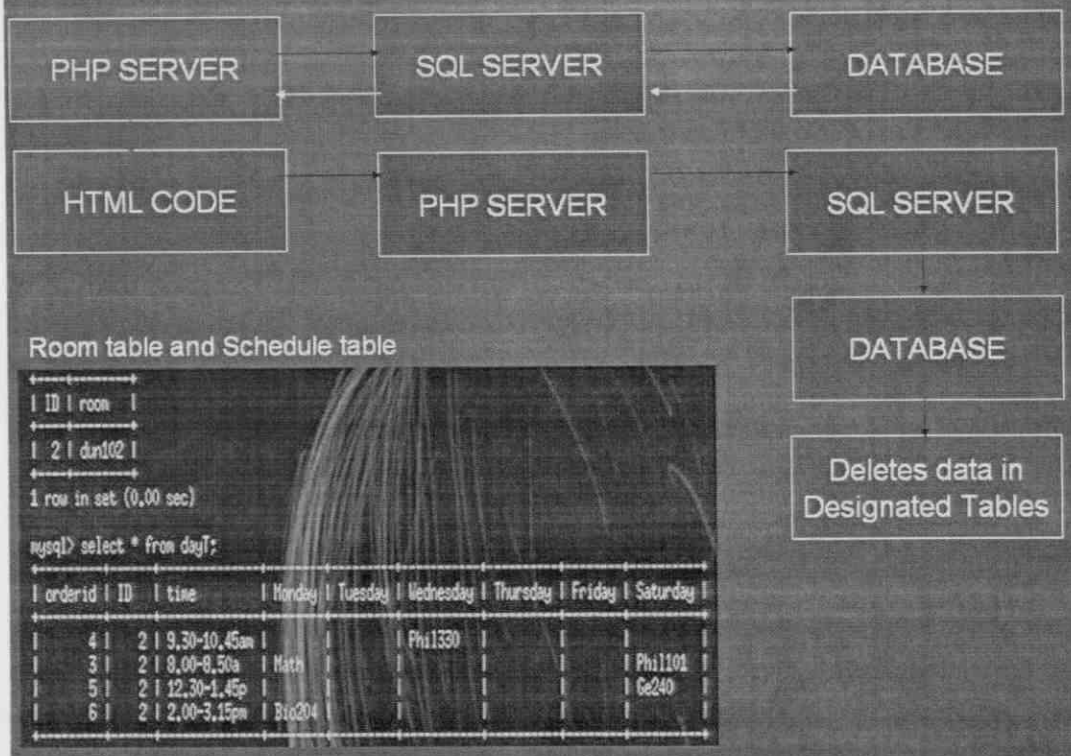
4.1. Technical Organization



4.2. Analysis


Enter Schedule
HTML FORM
PHP SERVER
SQL SERVER
DATABASE
Designated Tables
Room table and Schedule table
| ID | room |
| 2 | dun102 |
1 row in set (0.00 sec)
mysql> select * from dayT;
orderid	ID	time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
4	2	9.30-10.45am			Phil330			
3	2	8.00-8.50a	Math					Phil101
5	2	12.30-1.45p						Ce240
6	2	2.00-3.15pm	Bio204					

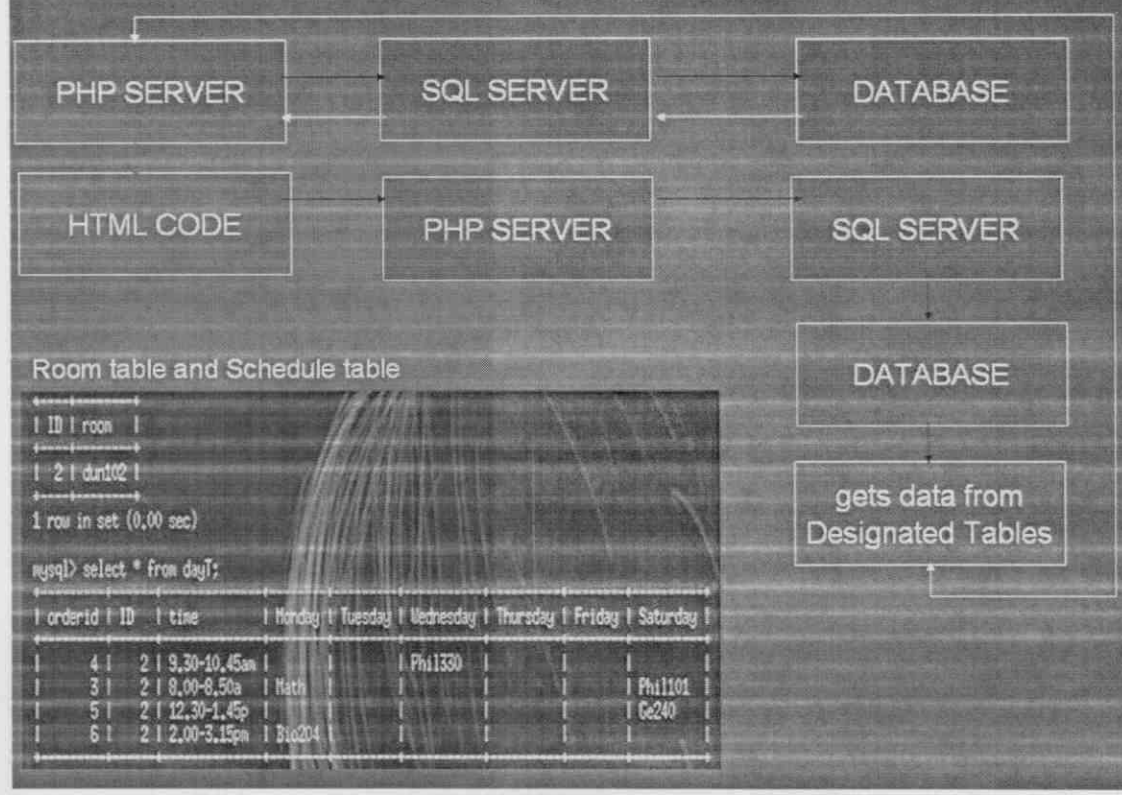
Delete Schedule



Room table and Schedule table

```
+----+--------+
| ID | room   |
+----+--------+
|  2 | dun102 |
+----+--------+
1 row in set (0.00 sec)

mysql> select * from dayT;
+---------+----+----------------+---------+---------+-----------+---------+---------+---------+
| orderid | ID | time           | Monday  | Tuesday | Wednesday | Thursday | Friday  | Saturday |
+---------+----+----------------+---------+---------+-----------+---------+---------+---------+
|       4 |  2 | 9.30-10.45am   |         |         | Phil330   |         |         |         |
|       3 |  2 | 8.00-8.50a     | Math    |         |           |         |         | Phil101 |
|       5 |  2 | 12.30-1.45p    |         |         |           |         |         | Ge240   |
|       6 |  2 | 2.00-3.15pm    | Bio204  |         |           |         |         |         |
+---------+----+----------------+---------+---------+-----------+---------+---------+---------+
```

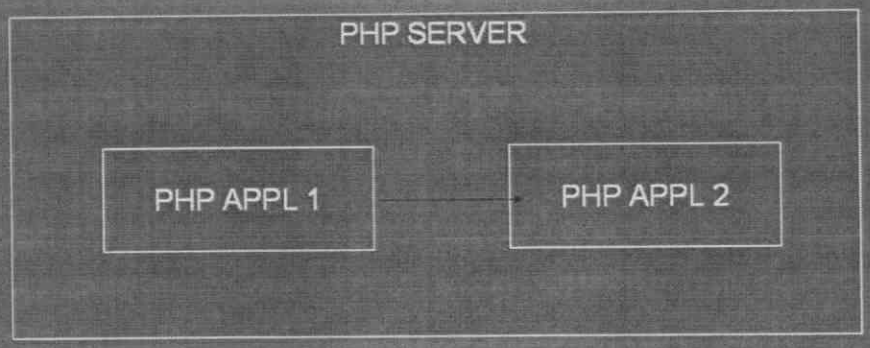
View Schedule



Room table and Schedule table

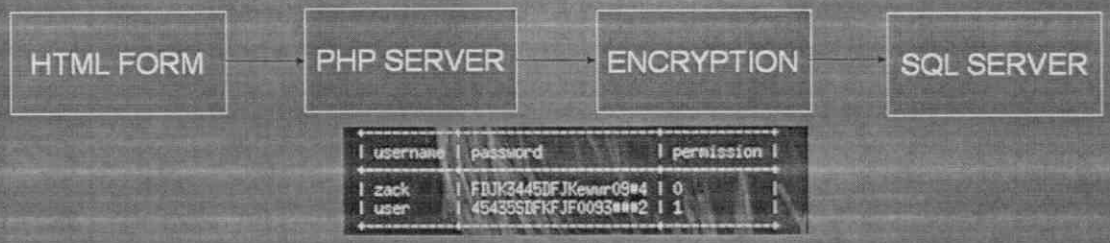
```
+----+--------+
| ID | room   |
+----+--------+
|  2 | dun102 |
+----+--------+
1 row in set (0.00 sec)

mysql> select * from dayT;
+---------+----+----------------+---------+---------+-----------+---------+---------+---------+
| orderid | ID | time           | Monday  | Tuesday | Wednesday | Thursday | Friday  | Saturday |
+---------+----+----------------+---------+---------+-----------+---------+---------+---------+
|       4 |  2 | 9.30-10.45am   |         |         | Phil330   |         |         |         |
|       3 |  2 | 8.00-8.50a     | Math    |         |           |         |         | Phil101 |
|       5 |  2 | 12.30-1.45p    |         |         |           |         |         | Ge240   |
|       6 |  2 | 2.00-3.15pm    | Bio204  |         |           |         |         |         |
+---------+----+----------------+---------+---------+-----------+---------+---------+---------+
```

View Schedule for a specific date



1. Retrieve current date
2. Locate current week
3. Access calendar
4. Retrieve the relevant week
5. Compute the exact position, and write HTML code

User Creation



username	password	permission
zack	FIUK3445DFJKewur09#4	0
user	45435DFKFJF0093###2	1

V. TIME TABLE AND FINAL PRODUCT

The project necessitated a couple thousand lines of code, the core components of which are available in the appendix, section C. The room scheduling system was realized over the span of three months following a particular design plan divided into four principal phases.

Research: Version compatibility between the different servers and software components was researched; diverse coding techniques as well as speed-efficient methods were also researched.

Coding: Core programming was implemented using a ground up approach by laying the foundational components, and then following their model to build the components dependant of them. Main parts were outlined, programmed separately, partially tested individually and in concert with other components and then included into the whole.

Testing: Even though testing occurred gradually as components were being created, added and changed to insure optimum compatibility and wide scale surprise, a main testing phase took part after the programming adventure to identify possible parts that needed to be changed or simply improved. Overall system compatibility was also tested, and allowed for latency and variable sharing issues to be fixed in the next phase.

Debugging: Incompatibility issues dwelled at the core of the session and constituted the most recurrent bugs encountered.

The project's time table was laid out as follows.

- February 10th - February 20th Specifications draft, installation of Software
- February 21st - March 10th Final Specifications, Code planning, GUI design.
- March 11th - April 11th Hardcore coding, and GUI integration
- April 15th - April 25th Testing and debugging.
- April 29th Project presentation

The application was up on the server and functional in the early hours of April 26th. The daemons behaved as expected, no latency issues arose. Known bugs were fixed, injection schemes nullified, and error checking on the user part was hardened. The room scheduling system and its add-ons are fully functional.

VI. TESTING, RESULTS, DEBUGGING ISSUES

Testing took ten days and was done from various standpoints, several requests were sent to the server, firewall ports were opened specifically for the application, security test on the confidentiality of stored cookies were also performed. The encryption schemes were reviewed, and tested. Downtime and server-queued requests were also tested. Optimum traffic situations were emulated using specialized software. SQL injection schemes were tested, and proved the system to be injection-proof. Database accessibility and capacity were also tested. As far as debugging, the encryption modules written for My SQL and for PHP seemed incompatible, as a debugging solution, a proprietary model allowing the two aforementioned standards to work was written. Another major debugging issue was related to the schedule by date determination algorithm. The algorithm used was such system demanding for PHP, that the code had to be converted to C, and included as a module. As a measure of security, critical files were removed from the web root directory and placed one branch up. These would include the header files necessary for the database access, and encryption module proper functioning. Sessions were tested, and session hijacking scenarios were implemented to test the authentication process. Upon accessing a table for the second time, the database server would shuffle the rows inside the table, making data retrieval an arduous task as far as schedule viewing. This was appeared according to the My SQL developers a bug in the version, that programmers needed to work around. To fix the issue, an order row consisting of self-generated integers was added to the schedule table. The order number would be issued by order of arrival. Before data retrieval, a sort command by order number is issued, which has the effect to set the table in order, making data retrieval for schedule view possible. Because of the possible multiple requests sent to the database server, a redundant request scheme was implemented, should one request return an accessibility error code. This would ensure that no request was lost, by assigning a time to live value (TTL), upon expiration of which, the software would reissue a request.

VII. CONCLUSION & RECOMMENDATIONS

The project met the set goals and specification, the room scheduling system can be used for any scheduling purposes involving storage of labeled information in a timely fashion. Others applications can be developed from this model, as the underlying frame supporting the application can be maintained and variant programs built upon it. The use of PHP and HTML as well as SQL makes it platform independent, as any operating system supports the aforementioned standards. The room scheduler needs not to be run on a resourceful computer as the application was designed to limit the number of simultaneous requests, and gather the most information from requests, thus preventing the daemons to increase traffic by sending messages back and forth. As far as recommendations, the scheduler should be maintained as one sees fit. Certain functions can be refined with the newer versions of PHP and SQL. The inclusion of a C module makes the application heavy to transfer around and thus could be eliminated as newer versions of the PHP language allow for more execution speed. Also the encryption scheme could be changed for something more elaborate and a little more powerful, as to avoid another external engine. The system could also be expanded to store room schedules by groups, for instance, rooms could be stored by department, and for this feature, another dimension of access level could be added as to grant access to users of a given department only to that department's resources. Even if the underlying database tables could be cleared in an instant to allow for instance for room schedules to be store say at the beginning of a new semester, adding an option to do so automatically from the admin panel could save an administrator the time to go an manually issue the required command from the MySQL daemon interface. As the system would become popular, and serve many users, an operation log sub-system could be implemented and imbedded into the application to track the changes made by users with a description of the operations performed, and a time at which they were performed. The room schedule themselves could be saved and displayed with time and date tags to give the user an idea of how long ago they were created or changed. To finish, certain schedules, such as personal schedules could be reserved, or locked by their owners, so only those would have access to them.