A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Uday Kanth Ranganamyna
Fall 2011
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Uday Kanth Ranganamyna:

Graduate Advisor and Faculty Appointment System - SDSU

Joseph Lewis, Chair
Department of Computer Science

Carl Eckberg
Department of Computer Science

Michael O'Sullivan
Department of Mathematics and Statistics

July 6, 2011
Approval Date
Copyright © 2011
by
Uday Kanth Ranganamyna
All Rights Reserved
DEDICATION

I would like to dedicate this thesis work to my parents and brothers. It would not have been possible without their boundless love, continuous motivation and support.
ABSTRACT OF THE THESIS

Graduate Advisor and Faculty Appointment System - SDSU
by
Uday Kanth Ranganamyna
Master of Science in Computer Science
San Diego State University, 2011

A large institution like San Diego State University (SDSU) has a huge numbers of students visiting the professors everyday for different reasons. Given that these professors are really busy teaching classes and doing research, the students might sometimes be forced to wait in long queues near the office of the advisors or professors thereby creating a busy pathway near the offices. This situation can therefore create disappointment among the students who may not get a chance to talk to the advisor or professor due to time constraints.

This thesis project provides an effective time management solution to the current problem using a web-based application which allows the professors and advisors to schedule their timings in the application using a web browser from either a mobile device or a computer. Also, this application will allow the students to make an appointment with a professor during his/her available schedule. Once a student makes an appointment, they will get a confirmation of their appointment using email. A professor can check all his/her appointments on a given date using this web application. This application guarantees that the students have a confirmed appointment time before meeting an advisor/professor.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT .......................................................... v</td>
</tr>
<tr>
<td>LIST OF FIGURES ......................................................... ix</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS ..................................................... xi</td>
</tr>
</tbody>
</table>

## CHAPTER

1 INTRODUCTION ................................................................. 1
2 TECHNOLOGY AND REQUIREMENTS ........................................ 3
   2.1 Software and Languages .............................................. 3
   2.2 Libraries ................................................................. 3
   2.3 JAVA ..................................................................... 4
   2.4 JavaScript ............................................................... 4
   2.5 JSP .......................................................................... 4
   2.6 HTML .................................................................... 5
   2.7 Ajax ....................................................................... 6
3 ARCHITECTURE AND DATABASE DESIGN ................................. 7
   3.1 Introduction ............................................................ 7
   3.2 Web Application (Client) .............................................. 8
   3.3 Database Design ..................................................... 8
       3.3.1 Register Table .................................................. 8
       3.3.2 Add Schedule Table .......................................... 9
       3.3.3 Appointment Type Table ..................................... 10
       3.3.4 Appointment Register Table ................................ 10
       3.3.5 Add CPT Table .................................................. 11
       3.3.6 Add Reduced Course Load Table ......................... 11
4 IMPLEMENTATION AND RESULTS .......................................... 13
   4.1 Authentication Module ............................................. 13
       4.1.1 Role Based Login Authorization .......................... 13
       4.1.2 Forgot Password ............................................. 16
4.2 Admin Module ...........................................................................................................18
  4.2.1 Add Professor ........................................................................................................20
  4.2.2 Edit Professor .........................................................................................................20
  4.2.3 View Professors .....................................................................................................22
  4.2.4 Delete Professor ....................................................................................................22
  4.2.5 Edit Admin .............................................................................................................22
4.3 Advisor and Professor Module ..................................................................................23
  4.3.1 Advisor Module ......................................................................................................24
    4.3.1.1 Add Appointment Type ..................................................................................24
    4.3.1.2 View Appointment Type ...............................................................................26
    4.3.1.3 Edit Appointment Type ..................................................................................26
    4.3.1.4 Delete Appointment Type ...............................................................................27
    4.3.1.5 Add Schedule ..................................................................................................27
    4.3.1.6 Edit Schedule ..................................................................................................30
    4.3.1.7 Delete Schedule ..............................................................................................30
    4.3.1.8 View Schedule ................................................................................................30
    4.3.1.9 View Appointments .......................................................................................30
    4.3.1.10 Edit Profile ....................................................................................................30
    4.3.1.11 View CPT Applications ...............................................................................32
    4.3.1.12 View Reduced Course Load Applications ...................................................33
  4.3.2 Professor Module ....................................................................................................34
    4.3.2.1 Add Schedule ..................................................................................................34
    4.3.2.2 Edit Schedule ..................................................................................................36
    4.3.2.3 Delete Schedule ..............................................................................................36
    4.3.2.4 View Schedule ................................................................................................36
    4.3.2.5 View Appointments .......................................................................................37
  4.4 Student Module ..........................................................................................................38
    4.4.1 View Appointments .............................................................................................38
    4.4.2 View Schedule ......................................................................................................39
    4.4.3 Make Appointments ............................................................................................39
    4.4.4 Apply CPT ...........................................................................................................44
4.4.5 Apply Reduced Course Load ................................................................. 44

5  CONCLUSION ............................................................................................... 47
    5.1 Limitation ............................................................................................... 47
    5.2 Future Enhancements ............................................................................. 47

REFERENCES .................................................................................................................. 49
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Home page</td>
<td>1</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>Advisor home page</td>
<td>2</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>JSP workflow</td>
<td>5</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Architecture</td>
<td>7</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Register table</td>
<td>8</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Add schedule table</td>
<td>9</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Add appointment type</td>
<td>10</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>Appointment register table</td>
<td>11</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Add CPT table</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>Add Reduced Course Load table</td>
<td>12</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Login page</td>
<td>14</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Login not successful</td>
<td>15</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Forgot password page</td>
<td>16</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>Forgot password mail</td>
<td>17</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>Invalid email</td>
<td>18</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>Admin use case diagram</td>
<td>19</td>
</tr>
<tr>
<td>Figure 4.7</td>
<td>Admin page</td>
<td>19</td>
</tr>
<tr>
<td>Figure 4.8</td>
<td>Add professor</td>
<td>21</td>
</tr>
<tr>
<td>Figure 4.9</td>
<td>Add user java script</td>
<td>21</td>
</tr>
<tr>
<td>Figure 4.10</td>
<td>Edit professor in admin page</td>
<td>22</td>
</tr>
<tr>
<td>Figure 4.11</td>
<td>Edit admin profile page</td>
<td>23</td>
</tr>
<tr>
<td>Figure 4.12</td>
<td>Advisor use case diagram</td>
<td>25</td>
</tr>
<tr>
<td>Figure 4.13</td>
<td>Advisor home page</td>
<td>25</td>
</tr>
<tr>
<td>Figure 4.14</td>
<td>Add appointment type</td>
<td>26</td>
</tr>
<tr>
<td>Figure 4.15</td>
<td>View appointment type</td>
<td>27</td>
</tr>
<tr>
<td>Figure 4.16</td>
<td>Add schedule calendar</td>
<td>28</td>
</tr>
<tr>
<td>Figure 4.17</td>
<td>Add schedule time picker</td>
<td>28</td>
</tr>
</tbody>
</table>
Figure 4.18. View advisor schedule........................................................................................31
Figure 4.19. View advisor appointments................................................................................31
Figure 4.20. Print CPT..........................................................................................................32
Figure 4.21. Print Reduced Course Load...............................................................32
Figure 4.22. Professor use case diagram...........................................................................34
Figure 4.23. Professor home page.......................................................................................35
Figure 4.24. Add professor schedule..................................................................................36
Figure 4.25. Edit professor profile.......................................................................................37
Figure 4.26. Student use case diagram..............................................................................38
Figure 4.27. Student page..................................................................................................39
Figure 4.28. View student appointment1............................................................................40
Figure 4.29. View student appointment2............................................................................40
Figure 4.30. Student view schedule....................................................................................41
Figure 4.31. Office appointment.........................................................................................42
Figure 4.32. Advising appointment.....................................................................................42
Figure 4.33. Student appointment mail...............................................................................43
Figure 4.34. CPT application form......................................................................................45
Figure 4.35. Reduced Course Load form............................................................................46
ACKNOWLEDGEMENTS

I take this opportunity to sincerely thank my thesis advisor, Dr. Joseph Lewis, for the guidance, encouragement, tremendous motivation and support during each and every step of my research work. Without his help, it would not have been possible for me to accomplish this huge complex project.

I would also like to thank Professor Carl Eckberg and Professor Michael O'Sullivan for providing valuable suggestions and taking their time to review my thesis as a part of the panel.
CHAPTER 1

INTRODUCTION

Since the invention of internet, web-applications have been in a great demand and use to people around the world in solving their various needs. Web applications allow users to access them using a web browser from Computer or Mobile phones. I have chosen web application development to help students and professors to utilize their time effectively and efficiently. The application home page can be seen in the Figure 1.1.

![Figure 1.1. Home page.](image)

The project constitutes of 4 Modules:

1. Authentication Module: It is role based authentication module to allow Administrator, Advisor and Professors to login into the application. Administrator will have permissions to add, edit and delete users. Professors will be able to add, edit and delete their schedule to allow students to make appointments. Administrator has the
ability to change the roles from professor to advisor or advisor to professor. If the professors forget their password he can use the forgot password facility.

2. Advisor Module: In this module the advisor has an ability to add, edit and delete schedule and he can schedule either office hour schedule or advising schedule. It allows the advisor to edit his profile information and also allows the advisor to add, edit and delete different appointment types which are viewed by the students when making appointment with Advisor. Appointment type is only visible when the scheduled time is Advising time. It also allows him to view appointments scheduled for the given date. Advisor module actions can be seen in the advisor home page Figure 1.2.

3. Professor Module: In this module. It allows the professor to edit his profile details. It allows professors to add, edit and delete his schedule. He will be allowed to have only office hour’s schedule. He can view the appointments made with him on a particular day.

4. Student Module: In this module. It will allow the students to make an appointment with the professors during office hours and advising hours. When it’s advising hours they can see what appointment type they are coming for and it will tell them the checklist items to bring along with them. A email notification of the appointment is send to the student. It contains start and end time scheduled with the professor.

Figure 1.2. Advisor home page.
CHAPTER 2

TECHNOLOGY AND REQUIREMENTS

2.1 SOFTWARE AND LANGUAGES

- Apache Tomcat Server – It is web server required for hosting the projects coded using advanced java (Servlets, Jsp).
- MySQL – It is required to create and maintain Administrator, Advisors, Professors and students information.
- Eclipse Java EE – It is IDE used to create JSP web projects.
- SMTP Server: It is used to send email appointment notifications and password recovery
- Java: The scriplets in Jsp allow us to write the java code in between the html code.
- JavaScript: It is used to validate the user input in the html forms.
- HTML: It is used to create the html forms for interacting with the application
- JSP: It is used to process the information on the server side
- AJAX: It is used for partial page refreshment in the application.
- JDBC: It is used to interact with the database inside the JSP pages.
- Web browser: It is required to interact with the web application by opening the required pages in it.

2.2 LIBRARIES

Commons-email-1.2.jar: This library is used to send emails after setting up the SMTP (Simple Mail Transfer Protocol) server.
Commons-fileupload-1.2.2.jar: This library is used to upload the file and set multi mime type in web page to upload the image into the database.
Commons-io-2.0.1.jar: This library is used to utilize the common utilities in the java code.
Mysql-connector-java-5.1.1.5-bin.jar: This library file is used to connect to the MySql database.
Standard.jar and jstl.jar: Both of these libraries are used by the JSP code inside the project.
2.3 JAVA

Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities.

Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is currently one of the most popular programming languages in use, and is widely used from application software to web applications [1].

2.4 JAVASCRIPT

Java script is a general purpose, prototype based, objects oriented scripting language developed jointly by Sun and Netscape. It is meant for the WWW. It is designed to be embedded in diverse applications and systems, without consuming much memory. Java script borrows most of its syntax from java but also inherits from awk and perl, with some indirect influence from self in its object prototype system.

Java scripts dynamically typed that are programs do not declare variable types, and the type of variable is unrestricted and can change at runtime. Source can be generated at runtime and evaluated against an arbitrary scope. Typical implementations compile by translating source into a specified byte code format, to check syntax and source consistency. Note that the availability to generate and interpret programs at runtime implies the presence of a compiler at runtime.

Java script is a high level scripting language that does not depend on or expose particular machine representations or operating system services. It provides automatic storage management, typically using a garbage collector [2].

2.5 JSP

JSP’s are built on top of Sun’s servlet technology. JSP’s are essential an HTML page with special JSP tags embedded. These JSP tags can contain Java code. The JSP file
extension is .jsp rather than .htm or .html. The JSP engine parses the .jsp and creates a Java servlet source file. It then compiles the source file into a class file, this is done the first time and this why the JSP is probably slower the first time it is accessed. Any time after this the special compiled servlet is executed and is therefore returns faster. JSP work-flow can be seen in the Figure 2.1.

**Figure 2.1. JSP workflow.**

### 2.6 HTML

HTML (hyper text markup language) is a language used to create hyper text documents that have hyper links embedded in them. It consists of tags embedded in the text of a document with HTML. We can build web pages or web documents. It is basically a formatting language and not a programming language. The browser reading the document interprets mark up tags to help format the document for subsequent display to a reader. HTML is a language for describing structured documents. HTML is a platform independent. WWW (World Wide Web) pages are written using HTML. HTML tags control in part the representation of the WWW page when view with web browser. The browser interprets HTML tags in the web document and displays it. Different browsers show data differently. Examples of browsers used to view websites are Mozilla and Firefox [3, 4].
2.7 Ajax

Ajax (asynchronous JavaScript and XML) is a group of interrelated web development methods used on the client-side to create interactive web applications. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the Background) without interfering with the display and behavior of the existing page. Data is usually retrieved using the XMLHttpRequest object. Despite the name, the use of XML is not needed (JSON is often used instead), and the requests need not be asynchronous.

Like DHTML and LAMP, Ajax is not one technology, but a group of technologies. Ajax uses a combination of HTML and CSS to mark up and style information. The DOM is accessed with JavaScript to dynamically display, and to allow the user to interact with the information presented. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads [5, 6].
CHAPTER 3
ARCHITECTURE AND DATABASE DESIGN

3.1 INTRODUCTION

Figure 3.1 represents the abstract architecture for this project. The web browser in laptop or workstation send a request to the web server and the web server sends a response to browser. It will display the html page requested by the user (students, professor) in web browser. If the request sent to web server need to get information from table it opens a connection with mysql database and displays the user appropriate information. The SMTP server receives request from web server when the user requests his password or user makes an appointment with the professor. The only requirement for this application is a web browser from the client side. So that, this application can also be accessed from Desktop or smart phones which has a web browser.

Figure 3.1. Architecture.
3.2 WEB APPLICATION (CLIENT)

This is a web application designed to run on any operating system. The only requirement for this application to be used by the users (student, professor) is to have a web browser that has the capability to run html and JavaScript content.

In the current application I have tested on the following web browsers Mozilla, Firefox and internet explorer. All the Features of the web application worked as designed.

3.3 DATABASE DESIGN

I wanted to use a free database because it does not cost the school for its licenses. So, I have used the widely used free open source database MySQL. The current database consists of four tables in the database to satisfy the requirements of our project. The database name is gradappointment. The four tables it consists are:

- Register
- Add schedule table
- Appointment type table
- Appointment Register table
- Add CPT table
- Add Reduced Course Load table

3.3.1 Register Table

The register table contains the information of professors and admin. The login authentication code uses this table information to authenticate the professor or admin and display appropriate home page designed for them. The following is the schema of the register table in Figure 3.2.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Datatype</th>
<th>NOT NULL</th>
<th>AUTO_INCREMENT</th>
<th>Flags</th>
<th>Default Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>VARCHAR(30)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>VARCHAR(30)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>firstname</td>
<td>CHAR(30)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastname</td>
<td>CHAR(30)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>email</td>
<td>VARCHAR(30)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>role</td>
<td>CHAR(12)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.2. Register table.
All the columns names in the table are not null. It is to ensure that all the information is provided. The username is a primary key because when we add a new user we don’t want to allow the user with the same name. The username and password are varchar to allow the admin to create a user name of his wish and also to have a secure password. The firstname and lastname should only be characters so we defined them to be character. The email is a varchar as it contains special characters, numbers and alphabets. The role should only contain characters so I have used char.

Depending on the role of the user after successful authentication, The Application displays the appropriate page for the user for his activity.

### 3.3.2 Add Schedule Table

This table contains the information of professors along with their office hours. The add schedule code of the professor page uses this table to add the schedule information into the table. The following is the schema of the add schedule table in the Figure 3.3.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Datatype</th>
<th>Null</th>
<th>Key</th>
<th>Flags</th>
<th>Default Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>INT(12)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>type_appointment</td>
<td>CHAR(12)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>date</td>
<td>DATE</td>
<td>✓</td>
<td></td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>p_starttime</td>
<td>TIME</td>
<td>✓</td>
<td></td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>s_starttime</td>
<td>TIME</td>
<td>✓</td>
<td></td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>s_endtime</td>
<td>TIME</td>
<td>✓</td>
<td></td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>username</td>
<td>VARCHAR(20)</td>
<td>✓</td>
<td>✓</td>
<td>BINARY</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>CHAR(40)</td>
<td>✓</td>
<td></td>
<td>BINARY</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>first_name</td>
<td>CHAR(20)</td>
<td>✓</td>
<td></td>
<td>BINARY</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>last_name</td>
<td>CHAR(20)</td>
<td>✓</td>
<td></td>
<td>BINARY</td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.3. Add schedule table.**

All the columns in the table are not null. It is to ensure that all the information is provided by the professor. We need a unique identifier to be used to extract the information. So, we have selected id has a primary key to extract the schedule information during making appointments. Date column uses date. p_starttime is the start time of the professor which is used to show the professor when his schedule starts. s_starttime is initially same as p_starttime. s_starttime is changed when a new appointment is made. s_endtime also remains the same once the schedule is entered and not edited under edit schedule. Column username
is a varchar because username contains characters, numbers and special characters. First name and last name contains characters so I have selected them as characters.

3.3.3 Appointment Type Table

This table contains the information about the appointment types. The appointment type table information is shown to the student when he makes an appointment during the advisor advising hours. The Figure 3.4 is the appointment type table schema.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Datatype</th>
<th>NOT NULL</th>
<th>AUTO INCREMENT</th>
<th>Flags</th>
<th>Default Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>VARCHAR(30)</td>
<td>✓</td>
<td></td>
<td>BINARY</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>message</td>
<td>TEXT</td>
<td>✓</td>
<td></td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>suggestedtime</td>
<td>INT(10)</td>
<td>✓</td>
<td>✓</td>
<td>UNSIGNED</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>LONGBLOB</td>
<td>✓</td>
<td></td>
<td>ZEROFILL</td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.4. Add appointment type.

In the above table the name is a primary key. It allows us to uniquely identify the row information. The name in the above table is varchar allowing the professor to enter the desired name he wants and thus allow students to identify it. Message is a text data type because it contains many words. Suggested time is selected as integer data type as time is in minutes. Image is selected as longblob as it is used to store the image file.

3.3.4 Appointment Register Table

This table contains the information of the appointments. Whenever a student’s wants to make an appointment or check the appointments of the professor. The Information is accessed and inserted into this table. The Figure 3.5 is the schema of the appointment register table.

We need complete information for each row in the table to perform operations in different pages accessed by professor and student. All the columns in the table are not null because of the previous reason mentioned. Beginning with student name, name only contains characters so its data type is selected as char. Red id is a number so we have selected redid data type as Integer in the table. email contains both number numbers and characters so its data type is selected as varchar. app_start and app_end both hold the start and end time of the appointment when an appointment is made by the student. So, both the app times are selected
Table 3.5. Appointment register table.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Datatype</th>
<th>Not Null</th>
<th>Auto Inc</th>
<th>Flags</th>
<th>Default Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>student_name</td>
<td>CHAR(30)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>redid</td>
<td>INT(15)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>email</td>
<td>VARCHAR(30)</td>
<td>✓</td>
<td></td>
<td>BINARY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>app_start</td>
<td>TIME</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>app_end</td>
<td>TIME</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prof_name</td>
<td>CHAR(30)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>date</td>
<td>DATE</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reason</td>
<td>VARCHAR(40)</td>
<td>✓</td>
<td></td>
<td>BINARY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>CHAR(10)</td>
<td>✓</td>
<td></td>
<td>BINARY</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

as time data type. Date column is used to hold the date of the appointment so we have selected date as data type date. Reason can contain both characters and numbers and some special characters so we have selected reason data type as varchar. Type contains the type of appointment student made it contains only character so we have selected data type char for type. This data type shows the advisor how many students have made appointment for office hours and advising hours.

3.3.5 Add CPT Table

The table contains the CPT information filled by the students in the Web Application. This is the table which the professor uses to print the CPT of the student. Whenever the professor deletes or edits the CPT information of the students, changes can be seen in this table. Figure 3.6 shows the schema of the Add CPT table. The Figure 3.6 Add CPT table, shows what are the data types we are using for each column. The id is the primary key, which helps us to extract the complete information for each row whenever needed.

3.3.6 Add Reduced Course Load Table

The table contains the Reduced Course Load information filled by the students and advisor in the Web Application. This is the table which the professor uses to print the Reduced Course Load of the student. Whenever the professor deletes or edits the Reduced Course Load information of the students, changes can be seen in this table. Figure 3.7 shows the schema of the Add Reduced Course Load table. The Figure 3.7 Add Reduced Course
Figure 3.6. Add CPT table.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Datatype</th>
<th>PK</th>
<th>FK</th>
<th>Flags</th>
<th>Default Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>INT(32)</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>WIKI</td>
</tr>
<tr>
<td>sname</td>
<td>VARCHAR(32)</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>WIKI</td>
</tr>
<tr>
<td>aname</td>
<td>VARCHAR(32)</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>WIKI</td>
</tr>
<tr>
<td>role</td>
<td>VARCHAR(45)</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>WIKI</td>
</tr>
<tr>
<td>cname</td>
<td>VARCHAR(45)</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>WIKI</td>
</tr>
<tr>
<td>address</td>
<td>TEXT</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>WIKI</td>
</tr>
<tr>
<td>hours</td>
<td>INT(2)</td>
<td>✔</td>
<td>✔</td>
<td>UNSIGNED</td>
<td></td>
<td>WIKI</td>
</tr>
<tr>
<td>type</td>
<td>VARCHAR(15)</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>WIKI</td>
</tr>
<tr>
<td>date</td>
<td>VARCHAR(10)</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>WIKI</td>
</tr>
</tbody>
</table>

Figure 3.7. Add Reduced Course Load table.

Load table shows what are the data types we are using for each column. The id is the primary key, which helps us to extract the complete information for each row whenever needed.
CHAPTER 4

IMPLEMENTATION AND RESULTS

There are four different modules which I have implemented in the application. The first module in the project is the authentication module.

4.1 AUTHENTICATION MODULE

The authentication module mainly contains two modules:

- 4.1.1 Role based login authorization
- 4.1.2 Forgot password

4.1.1 Role Based Login Authorization

When the web application is initially launched it shows the home page with a admin link for the application on the left side. When the user clicks on the link a page opens with asking the user to authenticate his login credentials in the page. When the user enters the username and password it will check with user details in the database. The login page Figure 4.1 can be found below. The jdbc code inside the jsp file uses a properties file to get the connection details. After using the connection it checks for the role associated with the user and redirects him to the appropriate page [7-10].

Code snippet for properties file:

```
url=jdbc:mysql://localhost:3306/
dbName=gradappointment
driver=com.mysql.jdbc.Driver
user=testname
password=testpassword
```

Code Snippet for login authorization file:

```
<%@ page language="java" import="java.sql.*, java.util.*, java.text.*, java.text.SimpleDateFormat, java.io.InputStream, java.util.Properties" %>

<
String userName = request.getParameter("userName");
String password = request.getParameter("password");

InputStream stream = application.getResourceAsStream("db.properties");
Properties props = new Properties();
```
Figure 4.1. Login page.

```java
props.load(stream);
String url = props.getProperty("url");
String dbName=props.getProperty("dbName");
String driver =props.getProperty("driver");
String username=props.getProperty("user");
String userPassword=props.getProperty("password");
Connection conn = null;

try {
Class.forName(driver).newInstance();
conn = DriverManager.getConnection(url+dbName,username,userPassword);
Statement st = conn.createStatement();

String strQuery = "select * from register where username='"+userName+'" and password='"+password+'"";

ResultSet rs = st.executeQuery(strQuery);

if(rs.next())
{
    String user=rs.getString(1);
    String firstName=rs.getString(3);
    String lastName=rs.getString(4);
    String role=rs.getString(6);
    session.setAttribute("username",user);
    session.setAttribute("firstname",firstName);
    session.setAttribute("lastname",lastName);
}
```
session.setAttribute("role",role);
if(role.contains("Admin"))
{
    response.sendRedirect("adminPage.jsp");
}
else{
    response.sendRedirect("professorPage.jsp");
}
else
{
    %>
    <html>
    <body>
    <h1>Please check your username and password</h1>
    </body>
    </html>
    <%
    conn.close();
    %>
} catch (Exception e) {
    e.printStackTrace();
}
%

When the user details are authenticated successfully. Depending on the role of the user in the database he is redirected to the appropriate page. If the login credentials are not valid, than the user is shown a message login not successful. It can be seen in the Figure 4.2.

Figure 4.2. Login not successful.
4.1.2 Forgot Password

The login page contains a link for forgot password. When the user clicks on the forgot password link in the page, he is redirected to the forgot password page. This can be seen in the Figure 4.3. When the user enters the password in the page and clicks on submit, the JSP code checks the email in the register table and if the email is valid, the login details from the register table are sent to the email address. If the password entered in the page is not a valid one, an appropriate message is displayed [11-13].

![Figure 4.3. Forgot password page.](image)

**Code snippet for email validation:**

```javascript
<script type="text/javascript">
function formValidator()
{
    var email = document.getElementById('email');
    if(notEmpty(email, "email cannot be empty")){
        if(emailValidator(email, "Please check email")){
            return true;
        }
    }
    return false;
}
</script>
```
function emailValidator(elem, helperMsg){
    var emailExp = /^\w\-\+@\[a-zA-Z0-9\-\+]+\.[a-zA-Z0-9]{2,4}$/;
    if(elem.value.match(emailExp)){
        return true;
    }else{
        alert(helperMsg);
        elem.focus();
        return false;
    }
}

When the user enters his email address in the forgot password page. It is also checked for weather it is a valid email address or not. The above code snippet is a code for email validation. When the user details are valid an email containing his username and password are sent to him [14].

In Figure 4.4 we can that the username and password are sent to him. Validation of email in the forgot password page can be seen in the Figure 4.5.

![Figure 4.4. Forgot password mail.](image-url)
4.2 ADMIN MODULE

When the user enters the login information with the admin privileges he will be displayed a admin page home page. The admin module contains sub modules. The following are the modules:

- 4.2.1 Add Professor
- 4.2.2 Edit Professor
- 4.2.3 View Professors
- 4.2.4 Delete Professor
- 4.2.5 Edit Admin

The Figure 4.6 admin use case diagram gives a clear over view of the operations that can be performed by the admin user in the appointment system application.

The home page of the admin page can be seen in the Figure 4.7 admin page. It shows different operations that can be performed by the admin user. If the admin of the application
Figure 4.6. Admin use case diagram.

Figure 4.7. Admin page.
wants to logout of the page. He can logout of the application by clicking on the logout link of the application.

**4.2.1 Add Professor**

The add professor link in the admin page allows the admin to add the new professor. He has the ability to add professor with two roles either as an advisor or professor. In the add professor page each and every input field is validated with java script to ensure proper data is provided when entering the information into the table of database. The input fields in the add professor page are username, password, first name, last name, email, role. The username can contains either characters or numbers. The password cannot be empty. The first name and last name must only contain characters when entering into the input text field. The role of the user must be selected either as a advisor or professor. When all the information is entered correctly and click on submit the information of the professor is entered into the database. As soon as the add professor is successful it will redirect him to his home page .There he can view the professor added using add professor.

The Figure 4.8 add professor shows us the view of the add professor page. If the user clicks on cancel button than he is redirected to admin home page. The add user page java script for validating the input fields in the page can be seen in Figure 4.9 add user java script page.

**4.2.2 Edit Professor**

In the admin home page when the admin selects a user he can either change the information of the professor or can also change the role of the professor. After editing the user details if the admin click on submit, changes will be saved. If he clicks on cancel no changes will be made and he is redirected to the admin main page where he can see how many professors are there. Edit professor page also contains java script to validate when the user has changed the fields properly. The username is readonly in the edit professor page. It can be seen in the Figure 4.10 edit professor in admin page.

The admin has also has the ability to change the role of the professor. He can change the role of the professor from either advisor to a professor or professor to advisor.
Figure 4.8. Add professor.

Figure 4.9. Add user java script.
4.2.3 View Professors

The Admin can see all the professors using the appointment system in his admin home page. They all are displayed in the form of a table showing their complete details. After viewing the professors in the home page he can either edit or delete them.

4.2.4 Delete Professor

In the admin page after viewing the different professors using the system. When the admin click on the delete link at the end of the row than that user will be deleted from the system.

4.2.5 Edit Admin

In the admin page there is a link called profile. When the admin clicks on this link he can edit his profile details. He can change all the fields except the username as it readonly and should not be allowed the by the user to change to it. Admin cannot edit his role. If we want to make any one admin than we need to do that only using the database. A sample edit admin profile page can be seen in Figure 4.11 edit admin profile page.
If the user clicks on the submit button after making changes, the changes will be saved. If the user clicks on the cancel button, the changes are not saved, and the admin is redirected to his home page. If the admin clicks on the logout link in the page, he is logged out from the application. All the input fields of the page in the edit admin page are validated using the java script.

**Code snippet of java script to see that the fields are not empty:**

```javascript
function notEmpty(elem, helperMsg) {
    if (elem.value.length == 0) {
        alert(helperMsg);
        elem.focus();
        return false;
    }
    return true;
}
```

### 4.3 ADVISOR AND PROFESSOR MODULE

This Module consists of two sub modules advisor module and professor module.
4.3.1 Advisor Module

When the user enters the login information with the advisor privileges he will be displayed an advisor home page. The advisor module contains sub modules. The following are the modules:

- 4.3.1.1 Add Appointment type
- 4.3.1.2 View Appointment type
- 4.3.1.3 Edit Appointment type
- 4.3.1.4 Delete Appointment type
- 4.3.1.5 Add Schedule
- 4.3.1.6 Edit Schedule
- 4.3.1.7 Delete Schedule
- 4.3.1.8 View Schedule
- 4.3.1.9 View Appointments
- 4.3.1.10 Edit Profile
- 4.3.1.11 View CPT Applications
- 4.3.1.12 View Reduced Course Load Applications

The Advisor use case diagram in Figure 4.12 shows the different operations that can be performed by the advisor. In Figure 4.13 Advisor home page shows all the links that can be used by the advisor.

4.3.1.1 ADD APPOINTMENT TYPE

The Advisor has the ability to add appointment type. When a new appointment type has been added by the advisor. This can be seen during the advising hour appointment made by the student. There is a drop down box of field type appointment name. Whenever the user selects one appointment type, the page is refreshed partially with the information of the appointment type. This helps students to have a checklist of items to bring with them when coming to the advisor [15-17].

The Figure 4.14 add appointment type shows the details that need to be filled by the advisor in the appointment type form. If he clicks on the submit button all the appointment
Figure 4.12. Advisor use case diagram.

Figure 4.13. Advisor home page.
type details will be saved. If he clicks on the cancel button the information will not be saved and redirected to his home page.

4.3.1.2 VIEW APPOINTMENT TYPE

The advisor can view all the appointment types inserted by him. He can view the different appointment types inserted by him by clicking the link view appointment types. He can go back to the advisor home page by clicking on the home link in the view appointment type’s page [18].

In the Figure 4.15 view Appointment type beside every row there is a link to either or delete the appointment type.

4.3.1.3 EDIT APPOINTMENT TYPE

The advisor can edit his appointment types made by him by click on the edit link in the view appointment types. The advisor can edit the appropriate information of the appointment type. After making necessary changes in the edit page if he clicks on the submit button. The necessary changes will be made to the appointment type. If he clicks on cancel
button the changes will not be saved and after that he will be redirected to the view appointments page and when he clicks on the home link he will be redirected to advisor home page.

4.3.1.4 DELETE APPOINTMENT TYPE

When the advisor clicks on the delete link beside each appointment type in the view appointment type page the appointment type will be deleted from the database. Once the appointment type is deleted from the table he is redirected again to the view appointments page to show the advisor that appointment type has been deleted.

4.3.1.5 ADD SCHEDULE

After the advisor logins into the application. In his home page there is a link for add schedule. When he clicks on the link add schedule a page containing the fields date, start time and end time and type of advising hours is asked by the advisor to fill them. The date field uses a java script to load a calendar in the page when clicks on the small image beside the date field. Using JavaScript, the calendar is written in such a way
that, it allows only to select the current date and after that date. It can be seen in the Figure 4.16 Add schedule calendar [19, 20].

![Schedule Details](image)

**Figure 4.16. Add schedule calendar.**

Java script code is written in the following file datetimepicker_css.js. It contains more than 1k lines of code.

The time fields use a java script which allows the user to select the time in 24 hours format. As soon as the user places the cursor in the field a time picker pops up to allow the advisor to select the time. It can be seen in the Figure 4.17 add schedule calendar. It is written in several JavaScript files [21, 22].

![Schedule Details](image)

**Figure 4.17. Add schedule time picker.**

Whenever the user enters the start time and end time the java script validation code will check whether the end time is greater than start time or not and displays the error message if end time is less than start time.
Code snippet in JavaScript to validate time:

```javascript
function formValidator()
{
  var starttime=document.getElementById('stime');
  var endtime=document.getElementById('etime');
  var d1one=document.getElementById('day');
  if(checktime(starttime,endtime,"endtime less than start time")){
    if(dright(d1one,"not a valid date format"){return true;
  }
  return false;
}

function dright(ele,helperMsg)
{
  if(ele.value.match(/^[0-9]{4}\-(0[1-9]|1[012])\-(0[1-9]|1[0-9]|2[01])$/))
  {
    return true;
  }
  else
  {
    alert(helperMsg);
    return false;
  }
}

function checktime(ele,ele1,helperMsg)
{
  var time1=ele.value;
  var time2=ele1.value;
  var seconds1 = time1.split(":")[0] * 3600 + time1.split(":")[1] * 60+ time1.split(":")[2];
  var seconds2 = time2.split(":")[0] * 3600 + time2.split(":")[1] * 60+ time2.split(":")[2];
  var diff=seconds2-seconds1;

  if(diff<0)
  {
    alert(helperMsg);
    return false;
  }
  return true;
}
</script>

If the user clicks on the submit button the schedule details of the advisor are saved. If he clicks on the cancel button the schedule details are not saved and redirected to the advisor home page.
4.3.1.6 EDIT SCHEDULE

After login into the advisor home page there is a link for edit beside the each schedule made by him. The edit schedule page has a java script validation to check that all the information changed is valid. If he clicks on the submit the changes will be saved and if he clicks on the cancel the edited changes will not be saved and he is redirected to the advisor home page.

4.3.1.7 DELETE SCHEDULE

After login into the advisor home page there is a link for delete beside the each schedule made by him. If he clicks on the delete link the schedule will be deleted from the schedule of the professor and he is redirected to the advisor home page.

4.3.1.8 VIEW SCHEDULE

After login into the advisor home page different schedules made by the advisor can be seen in his home page. The schedule page is written in such a way that it only displays the schedule from the current date and after that. This page has the code which deletes the schedules and appointments which are made before the current date.

In the following Figure 4.18 view advisor schedule. It shows all the schedules made by the advisor. The advisor can either edit or delete his schedule which can be seen in this figure.

4.3.1.9 VIEW APPOINTMENTS

After login into the advisor home page there is a link for view appointments. When he clicks on the view appointments. He need to select the date in the date field and click on submit it will show the advisor the appointments made the students during his office hours or advising hours on the given date. In the Figure 4.19 View advisor appointments. It can been seen clearly.

4.3.1.10 EDIT PROFILE

After login into the advisor home page there is a link for edit profile. When the advisor clicks on the link he can edit his profile details. The edit profile page contains java
Figure 4.18. View advisor schedule.

Figure 4.19. View advisor appointments.
script which checks weather all the valid information is provided by the advisor when he edits his profile information. If he clicks on the submit button the profile changes are successful. If he clicks on the cancel button changes are not saved and he is redirected to the advisor home page.

### 4.3.1.11 VIEW CPT APPLICATIONS

After login into the advisor home page there is a link for View Applications. When he clicks on the View CPT Applications he can see the CPT applications applied by the students. He can either edit the CPT details or delete the CPT applied by the student by clicking the edit and delete link beside each row containing the CPT application details for the selected student. If he clicks on the print link beside each row the row which contains the student CPT information. The Student details will be automatically filled in the CPT approval letter usually given by the advisor. So, the advisor can print the document for the student. The Figure 4.20 print CPT shows the sample print page that will be print by the advisor for the student.

![Figure 4.20. Print CPT.](image)
4.3.1.12 **VIEW REDUCED COURSE LOAD APPLICATIONS**

After login into the advisor home page there is a link for View Applications. When he clicks on the View Reduced Course Load Applications he can see the Reduced Course Load applications applied by the students. He can either edit the Reduced Course Load details or delete the Reduced Course Load applied by the student by clicking the edit and delete link beside each row containing the Reduced Course Load application details for the selected student. If he clicks on the print link beside each row the row which contains the student Reduced Course Load information. The advisor should click on edit link and enter the advisor details and then click on submit prior to printing the application to be completely filled by the automated system.

The Student details will be automatically filled in the Reduced Course Load approval letter usually given by the advisor. So, the advisor can print the document for the student. The Figure 4.21 Print Reduced Course Load shows the sample print page that will be print by the advisor for the student.

![APPLICATION FOR REDUCED COURSE LOAD](image)

**Figure 4.21. Print Reduced Course Load.**
4.3.2 Professor Module

When the user enters the login information with the professor privileges he will be displayed a professor home page. The professor module contains sub modules. The following are the modules:

- 4.3.2.1 Add schedule
- 4.3.2.2 Edit schedule
- 4.3.2.3 Delete schedule
- 4.3.2.4 View schedule
- 4.3.2.5 View appointments
- 4.3.2.6 Edit profile

The professor use case diagram in Figure 4.22 shows the different operations that can be performed by the professor in the web application. It can be seen in the Figure 4.23 professor home page.

![Professor use case diagram](image)

**Figure 4.22. Professor use case diagram.**

### 4.3.2.1 ADD SCHEDULE

After the professor logins into the application. In his home page there is a link for add schedule. The difference between the add schedule of the advisor and professor is that
advisor has the ability to have two kinds of office hours i.e. advising hours and office hours where has professor cannot. When he clicks on the link add schedule a page containing the fields date, start time and end time are needed to be filled by him. This can be seen in the Figure 4.24 Add professor schedule.

The date field uses a java script to load a calendar in the page when clicks on the small image beside the date field. The calendar is a java script is written in such a way that it allows only to select the current date and after that date.

The time fields use a java script which allows the user to select the time in 24 hours format. As soon as the user places the cursor in the field a time picker pops up to allow the advisor to select the time.

Whenever the user enters the start time and end time the java script validation code will check whether the end time is greater than start time or not and displays the error message if end time is less than start time.

If the user clicks on the submit button the schedule details of the professor are saved. If he clicks on the cancel button the schedule details are not saved and redirected to the professor home page.
4.3.2.2 EDIT SCHEDULE

After logging, in the professor home page there is a link for edit beside the each schedule made by him. The difference between the edit schedule of advisor and professor is he cannot change his type of office hours it is always office hours and hidden in the user interface. The edit schedule page has a java script validation to check that all the information changed is valid. If he clicks on the submit the changes will be saved and if he clicks on the cancel the edited changes will not be saved and he is redirected to the professor home page.

4.3.2.3 DELETE SCHEDULE

After login into the professor home page there is a link for delete beside the each schedule made by him. If he clicks on the delete link the schedule will be deleted from the schedule of the professor and he is redirected to the professor home page

4.3.2.4 VIEW SCHEDULE

After logging, in the professor home page different schedules made by the professor can be seen in his home page. The schedule page is written in such a way that it only displays
the schedule from the current date and after that. This page has the code which deletes the schedules and appointments which are made before the current date.

4.3.2.5 View Appointments

After logging, in the professor home page there is a link for view appointments. When he clicks on the view appointments, he needs to select the date in the date field and click on submit; it will show the professor the appointments made by the students on that day.

4.3.2.6 Edit Profile

After logging, in the professor home page there is a link for edit profile. When the professor clicks on the link, he can edit his profile details. It can be seen in the Figure 4.25 edit professor profile. The edit profile page contains JavaScript which checks whether all the valid information is provided by the professor when he edits his profile information.

![Edit Professor Profile](image)

Figure 4.25. Edit professor profile.

If he clicks on the submit button, the profile changes are successful. If he clicks on the cancel button, changes are not saved, and he is redirected to the professor home page.
4.4 Student Module

When the student opens the website of the appointment systems. There is a link called students when he clicks on that it gives the student various options to interact with the application. It contains three modules:

- 4.4.1 View appointments
- 4.4.2 View schedule
- 4.4.3 Make appointments
- 4.4.4 Apply CPT
- 4.4.5 Apply Reduced Course Load

The student use case diagram in Figure 4.26 gives the quick overview of the operations that can be performed by the student in the graduate appointment application. It can be seen in the Figure 4.27 student page.

Figure 4.26. Student use case diagram.

4.4.1 View Appointments

When the student clicks on the view appointments link in the student page. A page containing the drop down of the professors is displayed and we need to select one from them. The student also need to select a date and once the date is selected and clicks on submit. All the appointments made by different students are displayed on the page. Only
professors who have made an appointment schedule will only be displayed in the drop down of the professors.

The above mentioned for view appointments can be seen in the Figures 4.28 and 4.29 view student appointment1 and view student appointment 2.

4.4.2 View Schedule

When the student clicks on the available schedule for appointment. Only professors who have made an appointment schedule will only be displayed in the drop down of the professors. A page containing the drop down of the professors is displayed and we need to select one from them. The student also needs to select a date and once the date is selected and clicks on submit. All the schedules made by the professor on the given date will be displayed. It can be seen in the Figure 4.30 student view schedule.

4.4.3 Make Appointments

After viewing the schedules of the professors there is a link called make appointment beside each schedule. When the student clicks on that a form containing the information needed to filled by the student is displayed. If the student selects to make an appointment
Figure 4.28. View student appointment1.

Figure 4.29. View student appointment2.
Figure 4.30. Student view schedule.

during office hours appointment type is not displayed to the student. If the appointment is made during advising hours the appointment form will contain a appointment type the student wants to make and shows the information student need to bring by the partially updating that page according to his selected appointment type. The minimum and maximum time given to students is different according to weather it is an advising hour appointment or office hour appointment type.

It can be seen in the Figures 4.31 and 4.32 office appointment and advising appointment.

**Code snippet of AJAX for partial page update from drop down:**

```javascript
function showUser(str)
{
  if(str=="")
  {
    document.getElementById("txtHint").innerHTML="";
    return;
  }

  if (window.XMLHttpRequest)
```
Figure 4.31. Office appointment.

Figure 4.32. Advising appointment.
How this works can be seen in the Figure 4.32 advising appointment. If the appointment is made successfully an email containing the appointment details is sent to the student who made the appointment with the professor. It can be seen in the Figure 4.33 student appointment email.
Code snippet of java that sends email:

```java
HtmlEmail email = new HtmlEmail();
email.setHostName("smtp.gmail.com");
email.setSmtpPort(587);
email.setAuthenticator(new DefaultAuthenticator("sdsugradappointment", "sdsu_grad");
email.setTLS(true);
email.setFrom("sdsugradappointment@gmail.com");
email.setSubject("Appointment Mail");
StringBuffer msg = new StringBuffer();
msg.append("<html><body>");
msg.append("<h3>"+"Appointment Details"+"</h3>");
msg.append("<table border="1">");
msg.append("<tr><th>PROFESSOR
"+"</th><td>"+"name"+"</td></tr>");
msg.append("<tr><th>DATE(YYYY-MM-DD)
"+"</th><td>"+"appdate"+"</td></tr>");
msg.append("<tr><th>START TIME
(HH:MM)
"+"</th><td>"+"mailStartTime"+"</td></tr>");
msg.append("<tr><th>END TIME
(HH:MM)
"+"</th><td>"+"mailEndTime"+"</td></tr>");
msg.append("</table>");
email.setHtmlMsg(msg.toString());
email.addTo(student_email);
eemail.send();
```

4.4.4 Apply CPT

When the student clicks on Apply CPT link in the student page. The page displays the CPT information need to be filled by the student. The page contains the JavaScript to validate weather all the required information is filled by the student before submission. If the information is successfully submitted to the application a confirmation page will be displayed to the student that the information was uploaded successfully else an error message will be displayed to the student.

The Figure 4.34 CPT application form shows the view of the CPT page needed to be filled by the student for applying CPT. This information will be used in the printing the CPT of the student by the advisor.

4.4.5 Apply Reduced Course Load

When the student clicks on Apply Reduced Course Load link in the student page. The page displays the Reduced Course Load information need to be filled by the student. The
page contains the JavaScript to validate weather all the required information is filled by the student before submission. If the information is successfully submitted to the application a confirmation page will be displayed to the student that the information was uploaded successfully else an error message will be displayed to the student.

The Figure 4.35 Reduced Course Load form shows the view of the Reduced Course Load page needed to be filled by the student for applying Reduced Course Load. This information will be used in the printing the Reduced Course Load of the student by the advisor.

After the details entered by the student, the professor need to click on the edit link beside row containing the Reduced course load in order to fill the advisor details into the application form for printing the Reduced Course Load Form.
Figure 4.35. Reduced Course Load form.
CHAPTER 5

CONCLUSION

In my thesis project, I have implemented a web application called Graduate Advisor and Faculty Appointment System which provides an effective time management and scheduling solution for both the students and professors of the university. This application helps the advisor(s) know ahead of time, how many students are coming during the scheduled appointment period and for what specific reason/purpose. It also helps the student know what documents they need to bring during the scheduled appointment time. Apart from this core functionality, it provides the advisors, professors and the admins with a mechanism to manage the application effectively. It does so using a strong authentication mechanism built into the system. Another great thing about this application is that it could be re-used by any department in the university with minimal changes.

5.1 LIMITATION

The current application does not allow the advisors and professors to register themselves with the application. The advisors and professors need to request the administrator of the application to provide the login privileges to schedule their available times and notify students to make appointments.

5.2 FUTURE ENHANCEMENTS

The system developed so far will act as a framework over which many other useful features can be built. Some of the enhancements that could be done to the existing appointment system are:

- Provide the advisor with the functionality to approve or decline the applications for CPT, Reduced Course Load when a student has applied for it. Notify this to ISC advisor and the student about his decision.

- Provide the ability to the advisor to add new types of application forms and automate them in order to receive the input from the students. Automatically update links in the student page for these new forms when created by the advisor.
• Allow the advisor to delete the appointment made by any student who unnecessarily selects the maximum time and send a warning notification to the student, that he needs to check the messages to read before he comes to the appointment.

• Allow the advisor to change the timing of the student’s appointments when he deletes them and notify the student about the new appointment times when he modifies there appointment time.

• If the professor decides to delete the schedule, send a notification message to all the students who made appointment with him that the schedule has been canceled with the reason by the professor and tell them to make appointments in his next available schedule.

• Add a new role called ‘ISC (International Student Center) advisor’ and allow the users with this role to perform the following tasks:
  1. Login and see the CPT details approved by the advisor.
  2. Create a new CPT I-20 and notify the students.

• Allow the students to fill out a reduced course load application online and notify the professor once this form/application is submitted. The professor can either approve it or decline it. If he approves it, send a notification approval to the ISC advisor.

• The ISC advisor will see the approval from the advisor and she will approve the Reduced Course Load and notify the student to come and collect the form once she has completed the processing.

• Allow the students to fill out CPT application online and notify the professor once this application is submitted.

• Later, he can notify this to the ISC advisor about his decision and also, the tracking Id could be sent to the students for tracking purposes.

• The ISC advisor will start the process and notify the student that ISC has got approval from the advisor and his i-20 process has began and send a notification to the student to use his tracking id and check status for completion of his process.

• The ISC advisor would be provided with the ability to store all the approval letters from the advisor for 10 years. This will save the university lots of money on the paper.
REFERENCES


