

# Training Report on “Time Table Management”

At

**“GRAPESS SOLUTIONS”**

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology

In

Computer Engineering

By

“Vijay Sehgal”

Roll No.: “1812390”



Department of Computer Science and Engineering  
Haryana Engineering College, Jagadhri, Haryana, India  
Kurukshetra University, Kurukshetra, Haryana, India

2014-2015

## **ACKNOWLEDGEMENT**

Apart from the efforts of me, the success of my project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the completion of the project.

I would like to show my greatest appreciation to my project in-charge, Mr. Mukesh Jamwal. I can't say thank you enough for their tremendous support and help. I feel motivated and encouraged every time I attend her meeting. Without her encouragement and guideline this project work would not have been materialized.

I'm highly grateful to Mr. Mukesh Jamwal, I.T Education and Trainer at Grapess Solutions, for his thorough guidance right from one day till the end of training. He actually laid the ground for conceptual understanding of technologies and project.

Vijay Sehgal  
1812390

## **DECLARATION**

I, Vijay Sehgal hereby declare that the training report entitled (“Time Table Management”) has not presented as a part of any other academic work to get my degree or certificate except HEC Jagadhri for the fulfilment of the requirement for the degree of Bachelor Computer Science & Engineering.

Vijay Sehgal  
1812390

## LIST OF FIGURES

<b>S No.</b>	<b>Images</b>
1.	Login Screen
2.	Valid User Pop up
3.	Main Screen
4.	Add Subject Information Page
5.	View Subject Information Page
6.	Add Room Information Page
7.	View Room Information Page
8.	Update Room Information Page
9.	Delete Room Information Page
10.	Add Faculty Information Page
11.	View Faculty Information Page
12.	Add Faculty Subject Information Page
13.	View Faculty Subject Information Page
14.	Search Page Portal of Faculty Subject
15.	Add Room Allocation Portal Page
16.	View Room Allocation Portal Page
17.	Search Page Portal of Room Allocation

# TABLE OF CONTENTS

Chapter No.	Title
<b>1.</b>	<b>Company Profile</b>
1.1	About
<b>1.2</b>	Vision / Mission
<b>1.3</b>	Weekly Schedule
<b>2.</b>	<b>ProjectDescription</b>
2.1	Interface Requirement
2.2	Front End Used
<b>2.3</b>	Back End Used
2.3.1	Introduction To MYSQL
2.3.2	Features of MYSQL
2.3.3	E-R Diagram
<b>3.</b>	<b>Project Details</b>
3.1	System Requirement
3.1.1	Software Requirements
3.1.2	Hardware Requirements
<b>4.</b>	<b>Results/Output</b>
5.1	Initial Study of the Project (Snapshots)
<b>5.</b>	<b>Future Scope</b>
<b>6.</b>	<b>Conclusion</b>
<b>7.</b>	<b>Appendix</b>
<b>8.</b>	<b>References</b>
8.1	Bibliography
<b>8.2</b>	Institute Contact Details

## **1.1 About Grapess Solutions**

Grapess Solutions has started in March, 2010 with a vision to provide every solution of IT. We have started with IT education & training and software solution

In Web Design & Development:

Grapess Solutions provides professional website design, development and maintenance services. Our experienced web designers and developers carry out various website projects from brochure sites to multi-functional web portals. Grapess Solutions has a large team of resources and the required business and technical expertise to develop websites of any complexity. We provide complete front-end and back-end development based on the latest technologies and industry trends.

What you can expect with every website design project:

- Great, fast loading, modern design.
- Private, in progress viewing during design and development.
- Content management system to easily edit your website.
- Basic search engine optimization to help you rank better.
- Site map to help visitors find what they are looking for.
- User friendly/Search engine friendly navigation menu.
- Email/information request forms to allow visitor feedback and inquiries.
- Upload to your website host when going live.
- Manual submission to Google, Yahoo, and Bing.
- Free minor changes for two weeks following launch of new site
- Providing Industrial Training and Education Training.

## **1.2 Vision / Mission:-**

### **➤ Our Vision**

Our vision is to be the leader in Infrastructure Products, Services and creating value for our customers to earn their Lifetime loyalty.

### **➤ Our Mission**

Every team member at Grapess Solutions will strive to provide Cost effective, Best Quality Products and Services to our customers through adherence to and with continuous improvements in Standards, Systems and Processes.

Our Software Development Services include:

1. Windows Software Development
2. Web Application Development
3. Software Products
4. Software Rebranding

## **Website Designing**

Professional website designing and website development services, web application development, ecommerce website development, web design India, offshore website design, SEO services, offshore engineering services and web design services.

Our services are backed by a professional work portfolio and glowing reviews. We have been providing best-in-class web site design, web development, search engine optimization & website maintenance services to Indian & offshore businesses since 2002. Our website design services will get you the results that you always wanted. Our websites are clean, clear and customized to your needs! Your website is an online resume for your business. Why wouldn't you want to appear you're best? At Web shine, we provide professional website designing and create custom logos at affordable prices.

We can improve the appearance of your web page templates, creating a visually appealing layout that will encourage your visitors to linger and take the time to read the content on your website instead of surfing the internet for other similar sites. Websites have emerged as a fast, reliable and efficient media to display and share useful piece of information with the help & use of pictures, text and graphics. Within a short span of time we have designed & developed a number of websites that are successfully fulfilling their objectives.

### 1.3 WEEKLY SCHEDULE

Classes used to hold regularly i.e. six days a week .Every day a two hour class used to be there. For an hour, sir used to give us the theory classes and for one hour we used to practice the concepts taught to us by the sir. It was a great learning experience to work under the guidance of Mr. Sandeep Sharma who is our project guide.

#### Phases Involved in the development of the system

- Identify needs and benefits
  - ❖ Studied and identified the existing system.
  - ❖ Identify needs and project constraints
  - ❖ Established project statements

Prepared a detailed report of the existing system at work.

- ❖ Prepare the software requirement specifications.
- ❖ Actual coding started



## **2.1 INTERFACE REQUIREMENT**

### **1. User Interface**

The package must be user friendly and robust. It must prompt the user with proper message boxes to help them perform various actions and how to precede further the system must respond normally under any in out conditions and display proper message instead of turning up faults and errors.

### **Software and Hardware Requirements**

In the following, we describe the software and hardware requirements for system developers and system users. During our system development, we have to design both static and dynamic website interfaces, create website functions and a database system, edit photos and pictures, and print out reports, so it has a set of software and hardware requirements.

#### **Software Requirements**

Operating System	:	Windows XP, Windows 7, Windows 8.1
Front- End	:	J2SE (Java)
Back- End	:	MYSQL (RDBMS)

#### **Hardware Requirements**

Processor	:	Intel Core i3 or more
RAM	:	4GB or More
Hard Disk Drive	:	500GB or More
Video	:	1280X800, 256 colors
CD-ROM	:	Optional
Key Board	:	Standard 101/102 or Digi Sync Family
Monitor	:	Display Panel (1280 X 800)
Display Adapter	:	Trident Super VGA
Network Adapter	:	SMC Ethernet Card Elite 16 Ultra
Mouse	:	Logitech Serial Mouse/Inbuilt

## 2.2Front-end Used

### JAVA

Java Technology is both a programming language and a platform.

Java: The Programming Language

The Java programming language is a high level language that can be characterized by all of the following buzzwords:

- Simple
- Object oriented
- Distributed
- Interpreted
- Robust
- Secure
- Architecture neutral
- Portable
- High performance
- Multithreaded

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

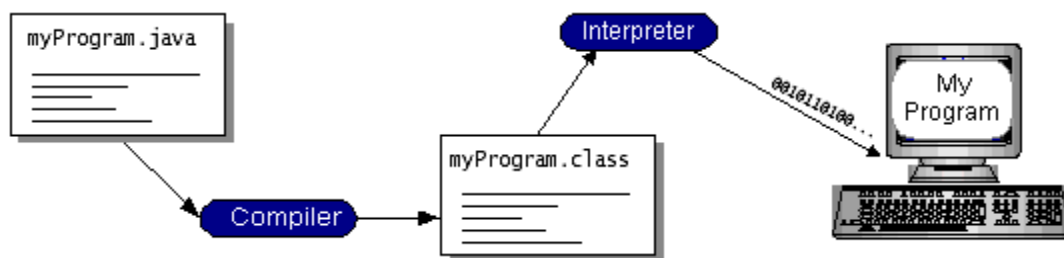


Fig 1. Working of Java Byte codes

You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM.

Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

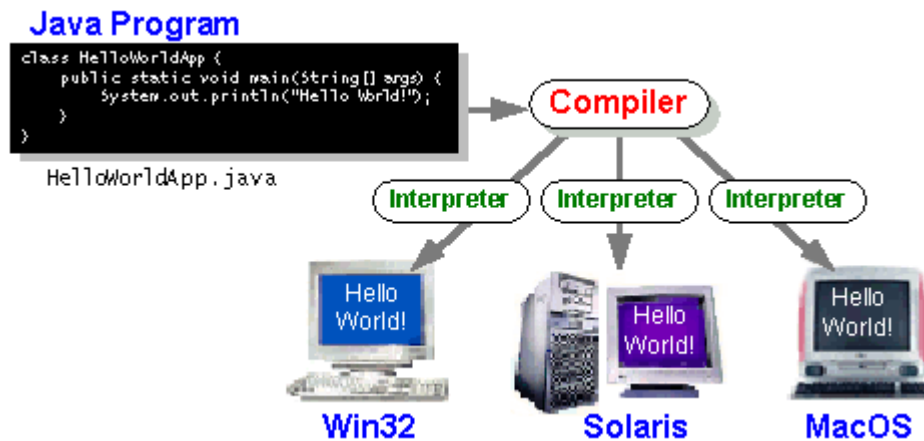


Fig 2. Implementation of the Java VM

## Java: The Platform

A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

Java Virtual Machine is standardized hypothetical computer, which is emulated inside our computer by a program. It is base of Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

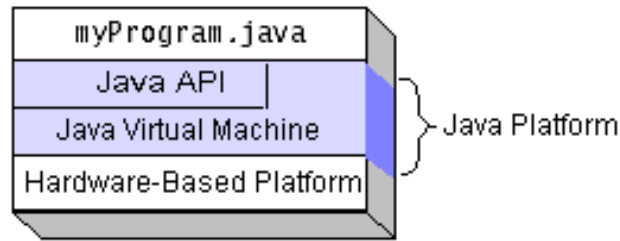


Fig 3. Java API and Virtual Machine

### What Can Java Technology Do...?

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've accessed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

### JFC and Swing

JFC and Swing is a part of java extension. JFC is short for Java Foundation Classes, which encompass a group of features to help people build graphical user interfaces (GUIs). The JFC was first announced at the 1997 Java One developer conference and is defined as containing the following features:

- **The Swing Components**

Include everything from buttons to split panes to tables.

- **Pluggable Look and Feel Support**

It gives any program that uses Swing components a choice of looks and feels. For example, the same program can use either the Java look and feel or the Windows look and feel. We expect many more look-

and-feel packages -- including some that use sound instead of a visual "look" -- to become available from various sources.

- **Accessibility API**

It enables assistive technologies such as screen readers and Braille displays to get information from the user interface.

- **Java 2D™ API (Java 2 Platform only)**

It enables developers to easily incorporate high-quality 2D graphics, text, and images in applications and in applets.

- **Drag and Drop Support (Java 2 Platform only)**

Provides the ability to drag and drop between a Java application and a native application.

The Swing API is available in two forms:

- As a core part of the Java 2 Platform, Standard Edition (including versions 1.2, 1.3, and 1.4)
- JFC 1.1 (for use with JDK 1.1)

Sun community recommends that to use the latest version of the Java 2 Platform. Not only will you be getting the latest bug fixes, but you'll get more features. (As we used JSDK1.5.0\_01 version, which is the latest release of java development kit up to the time of written of this report.)

## **2.3Back End Used**

### **2.3.1 Introduction to MYSQL**

The MySQL (R) software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software. MySQL is a trademark of MySQL AB.

The MySQL software is Dual Licensed. Users can choose to use the MySQL software as an Open Source/Free Software product under the terms of the GNU General Public License or can purchase a standard commercial license from MySQL AB. The MySQL web site(<http://www.mysql.com/>) provides the latest information about the MySQL software.

### **2.3.2Features of MySQL**

The following list describes some of the important characteristics of the MySQL Database Software

1. Internals and Portability

- Written in C and C++.
- Tested with a broad range of different compilers.
- Works on many different platforms. Uses GNU Auto make, Autoconf, and Libtool for portability.
- APIs for C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, and Tcl are available.
- Fully multi-threaded using kernel threads. This means it can easily use multiple CPUs if they are available.
- Provides transactional and non-transactional storage engines.
- A very fast thread-based memory allocation system.
- Very fast joins using an optimized one-sweep multi-join.
- In-memory hash tables which are used as temporary tables.
- SQL functions are implemented using a highly optimized class library and should be as fast as possible. Usually there is no memory allocation at all after query initialization.
- The MySQL code is tested with Purify (a commercial memory leakage detector) as well as with Valgrind, a GPL tool.
- The server is available as a separate program for use in a client/server networked environment. It is also available as a library that can be embedded (linked) into standalone applications. Such applications can be used in isolation or in environments where no network is available.

## 2. Column Types

- Many column types: signed/unsigned integers 1, 2, 3, 4, and 8 bytes long, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM, and Open GIS geometry types.
- Fixed-length and variable-length records.

## 3. Commands and Functions

- Full operator and function support in the SELECT and WHERE clauses of queries. **For example:**

```
mysql> SELECT CONCAT(first_name, ' ', last_name)
-> FROM tbl_name
-> WHERE income/dependents > 10000 AND age > 30;
```

- Full support for SQL GROUP BY and ORDER BY clauses. Support for group functions (COUNT(), COUNT(DISTINCT ...), AVG(), STD(), SUM(), MAX(), MIN(), and GROUP\_CONCAT()).
- Support for LEFT OUTER JOIN and RIGHT OUTER JOIN with both standard SQL and ODBC syntax.
- Support for aliases on tables and columns as required by SQL-92.

- DELETE, INSERT, REPLACE, and UPDATE return the number of rows that were changed (affected). It is possible to return the number of rows matched instead by setting a flag when connecting to the server.
- The MySQL-specific SHOW command can be used to retrieve information about databases, tables, and indexes. The EXPLAIN command can be used to determine how the optimizer resolves a query.
- Function names do not clash with table or column names. For example, ABS is a valid column name. The only restriction is that for a function call, no spaces are allowed between the function name and the '(' that follows it.

#### 4. Security

- A privilege and password system that is very flexible and secure, and allows host-based verification. Passwords are secure because all password traffic is encrypted when you connect to a server.

#### 5. Scalability and Limits

- Handles large databases. We use MySQL Server with databases that contain 50 million records. We also know of users that use MySQL Server with 60,000 tables and about 5,000,000,000 rows.
- Up to 32 indexes per table are allowed. Each index may consist of 1 to 16 columns or parts of columns. The maximum index width is 500 bytes (this may be changed when compiling MySQL Server). An index may use a prefix of a CHAR or VARCHAR column.

#### 6. Connectivity

- Clients may connect to the MySQL server using TCP/IP sockets on any platform. On Windows systems in the NT family (NT, 2000, or XP), clients may connect using named pipes. On Unix systems, clients may connect using Unix domain socket files.
- The Connector/ODBC interface provides MySQL support for client programs that use ODBC (Open-Database-Connectivity) connections. For example, you can use MS Access to connect to your MySQL server. Clients may be run on Windows or Unix. Connector/ODBC source is available. All ODBC 2.5 functions are supported, as are many others.
- The Connector/JDBC interface provides MySQL support for Java client programs that use JDBC connections. Clients may be run on Windows or Unix. Connector/JDBC source is available.
- The server can provide error messages to clients in many languages.
- Full support for several different character sets, including ISO-8859-1 (Latin1), German, big5, ujis, and more. For example, the Scandinavian characters '@^a', '@"a' and '@"o' are allowed in table and column names. Unicode support is available as of MySQL

- All data is saved in the chosen character set. All comparisons for normal string columns are case-insensitive.
- Sorting is done according to the chosen character set. It is possible to change this when the MySQL server is started. To see an example of very advanced sorting, look at the Czech sorting code. MySQL Server supports many different character sets that can be specified at compile and runtime.

## 7. Clients and Tool

The MySQL server has built-in support for SQL statements to check, optimize, and repair tables. These statements are available from the command line through the MySQL check client. MySQL also includes myisamchk, a very fast command-line utility for performing these operations on MyISAM tables.

- All MySQL programs can be invoked with the --help or -? Options to obtain online assistance.

## 2.4ER Diagram

The Entity Relationship Diagram (ERD) is the graphical notation of the relationship between data object and attributes. The ERD was originally proposed by Peter Chen for the design of relational database systems and has been extended by others.

Primary components identified for ERD are:

- Data objects
- Attributes
- Relationship
- Various type indicators.

### Purpose of ERD

The primary purpose of the ERD is to represent data objects and their relationship.



## **3.1 System Required**

### **3.1.1 Software Requirements**

- Window 8.1 or any other
- Jdk 1.7.0\_25
- NetBeans 8.0

#### **a) NetBeans:**

NetBeans refers to both a platform framework for Java desktop applications, and an integrated development environment (IDE) for developing with Java, JavaScript, PHP, Python, C, C++ and others. The NetBeans IDE is written in Java and can run anywhere a compatible JVM is installed, including Windows, Mac OS, Linux, and Solaris. A JDK is required for Java development functionality, but is not required for development in other programming languages.

The NetBeans platform allows applications to be developed from a set of modular software components called modules.

The NetBeans IDE is an open-source integrated development environment. NetBeans IDE supports development of all Java application types. The NetBeans Platform is a reusable framework for simplifying the development of Java Swing desktop applications. The NetBeans IDE bundle for Java SE contains what is needed to start developing NetBeans plugins and NetBeans Platform based applications; no additional SDK is required.

Applications can install modules dynamically. Any application can include the Update Center module to allow users of the application to download digitally-signed upgrades and new features directly into the running application. Reinstalling an upgrade or a new release does not force users to download the entire application again.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are:

- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management (saving and loading any kind of data)
- Window management

- Wizard framework (supports step-by-step dialogs)
- NetBeans Visual Library
- Integrated Development Tools

### 3.1.2 Hardware Requirements:

- Processor: Pentium IV
- Primary Memory: Minimum 256 MB for server and 128 MB for clients. The requirements of the primary memory will vary depending on the terminal i.e. whether it is a server or a client terminal. The server has to load the Microsoft Access database into the memory and also has to deal with the incoming connections from the users. Hence, it requires comparatively high amount of memory in comparison to the memory required by the client terminal.
- Secondary Memory: Minimum 32 GB for server and 15 GB for clients, since the server must have Microsoft Access database installed on it, the secondary memory required for the server has to be greater than the client's terminal. The secondary memory requirement for the installation of Microsoft Access is 215 MB. The approximate memory required for this application will be 30 MB.
- JDBC
- Setting up a Database
  - ✓ We will assume that the database PHONEBOOKINFO already exists. (Creating a database is not at all difficult, but it requires special permissions and is normally done by a database administrator.) When you create the tables used as examples in this tutorial, they will be in the default database. We purposely kept the size and number of tables small to keep things manageable.
  - ✓ First we will show you how to open a connection with your DBMS, and then, since what JDBC does is to send your SQL code to your DBMS, we will demonstrate some SQL code. After that, we will show you how easy it is to use JDBC to pass these SQL statements to your DBMS and process the results that are returned.
  - ✓ This code has been tested on most of the major DBMS products. However, you may encounter some compatibility problems using it with older ODBC drivers with the JDBC-ODBC Bridge.
- Establishing a Connection
  - ✓ The first thing you need to do is establish a connection with the DBMS you want to use. This involves two steps: (1) loading the driver and (2) making the connection.

- ✓ Ospecified in the JDBC URL. The DriverManager class, true to its name, manages all of the details of establishing the connection for you behind the scenes. Unless you are writing a driver, you will probably never use any of the methods in the interface Driver, and the only DriverManager method you really need to know is DriverManager.getConnection.
- ✓ The connection returned by the method DriverManager.getConnection is an open connection you can use to create JDBC statements that pass your SQL statements to the DBMS. In the previous example, con is an open connection, and we will use it in the examples that follow.

- Creating JDBC Statements

- ✓ A Statement object is what sends your SQL statement to the DBMS. You simply create a Statement object and then execute it, supplying the appropriate execute method with the SQL statement you want to send. For a SELECT statement, the method to use is executeQuery . For statements that create or modify tables, the method to use is executeUpdate .
- ✓ It takes an instance of an active connection to create a Statement object. In the following example, we use our Connection object con to create the Statement object stmt :
- ✓ Statement stmt = con.createStatement();
- ✓ At this point stmt exists, but it does not have an SQL statement to pass on to the DBMS. We need to supply that to the method we use to execute stmt . For example, in the following code fragment, we supply executeUpdate with the SQL statement from the example above:
- ✓ stmt.executeUpdate("CREATE TABLE PINFO" +
- ✓ "(UNAME VARCHAR(32), PASSWORD INTEGER, PRICE FLOAT, " + "SALES INTEGER, TOTAL INTEGER)");
- ✓ Since we made a string out of the SQL statement and assigned it to the variable createTableCoffees , we could have written the code in this alternate form:
- ✓ stmt.executeUpdate(createTableCoffees);

- Executing Statements

- ✓ We used the method executeUpdate because the SQL statement contained in createTableCoffees is a DDL (data definition language) statement. Statements that create a table, alter a table, or drop a table are all examples of DDL statements and are executed with the method executeUpdate . As youmightexpect from its name, the method executeUpdate is

also used to execute SQL statements that update a table. In practice, `executeUpdate` is used far more often to update tables than it is to create them because a table is created once but may be updated many times.

- ✓ The method used most often for executing SQL statements is `executeQuery`. This method is used to execute `SELECT` statements, which comprise the vast majority of SQL statements. You will see how to use this method shortly.

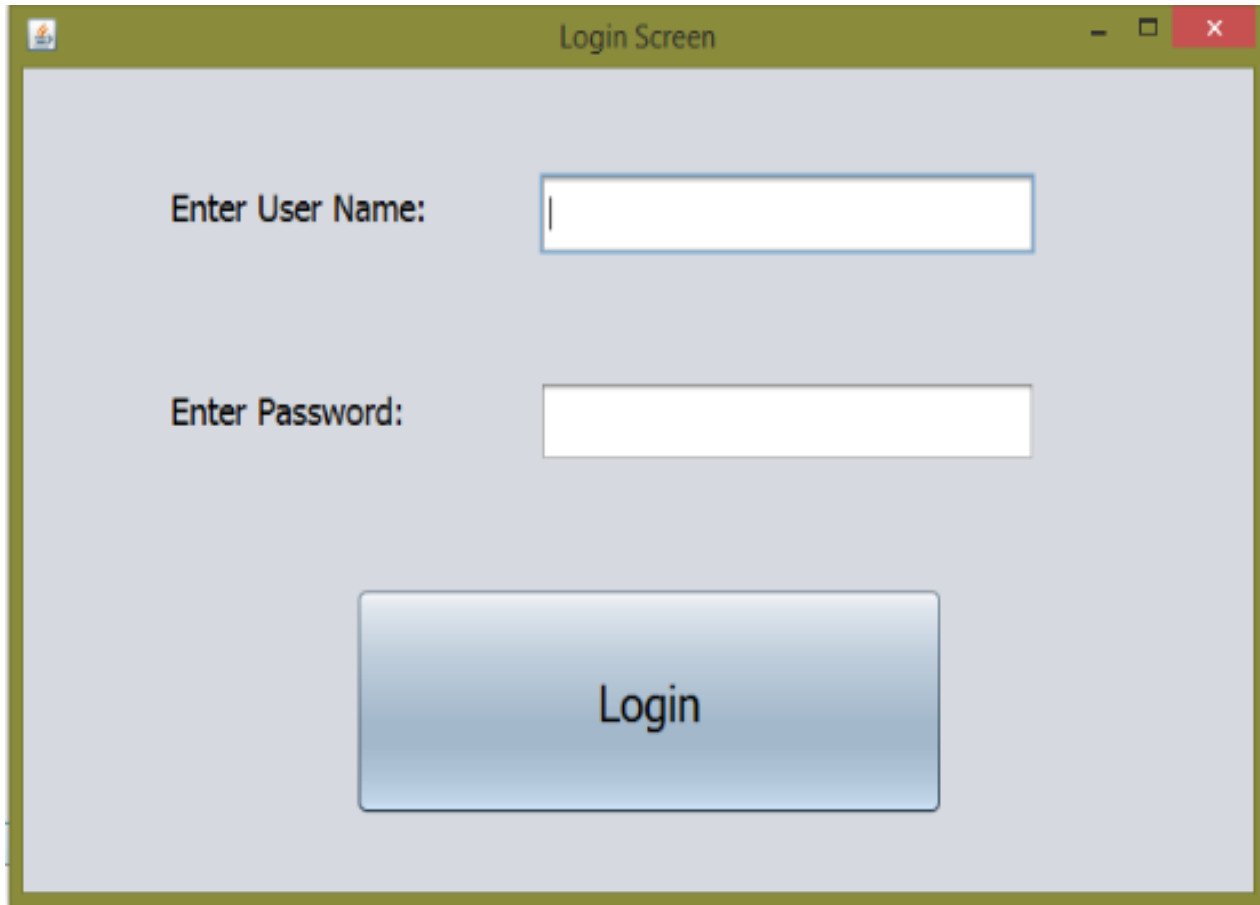
- Entering Data into a Table

- We have shown how to create the table `COFFEES` by specifying the names of the columns and the data types to be stored in those columns, but this only sets up the structure of the table. The table does not yet contain any data. We will enter our data into the table one row at a time, supplying the information to be stored in each column of that row. Note that the values to be inserted into the columns are listed in the same order that the columns were declared when the table was created, which is the default order.
- The following code inserts one row of data, with `Colombian` in the column `COF_NAME`, `101` in `SUP_ID`, `7.99` in `PRICE`, `0` in `SALES`, and `0` in `TOTAL`. (Since The Coffee Break has just started out, the amount sold during the week and the total to date are zero for all the coffees to start with.) Just as we did in the code that created the table `COFFEES`, we will create a `Statement` object and then execute it using the method `executeUpdate`.
- Since the SQL statement will not quite fit on one line on the page, we have split it into two strings concatenated by a plus sign (+) so that it will compile. Pay special attention to the need for a space between `COFFEES` and `VALUES`. This space must be within the quotation marks and may be after `COFFEES` or before `VALUES`; without a space, the SQL statement will erroneously be read as `"INSERT INTO COFFEESVALUES . . ."` and the DBMS will look for the table `COFFEESVALUES`. Also note that we use single quotation marks around the coffee name because it is nested within double quotation marks. For most DBMSs, the general rule is to alternate double quotation marks and single quotation marks to indicate nesting.
- `Statement stmt = con.createStatement();`
- `stmt.executeUpdate( "INSERT INTO COFFEES " + "VALUES ('Colombian', 101, 7.99, 0, 0)");`
- The code that follows inserts a second row into the table `COFFEES`. Note that we can just reuse the `Statement` object `stmt` rather than having to create a new one for each execution.
- `stmt.executeUpdate("INSERT INTO COFFEES " + "VALUES ('French_Roast', 49, 8.99, 0, 0)");`

- Values for the remaining rows can be inserted as follows:
- `stmt.executeUpdate("INSERT INTO COFFEES " + "VALUES ('Espresso', 150, 9.99, 0, 0)");`
- `stmt.executeUpdate("INSERT INTO COFFEES " + "VALUES ('Colombian_Decaf', 101, 8.99, 0, 0)");`
- `stmt.executeUpdate("INSERT INTO COFFEES " + "VALUES ('French_Roast_Decaf', 49, 9.99, 0, 0)");`

## 4.1 INITIAL STUDY OF THE PROJECT

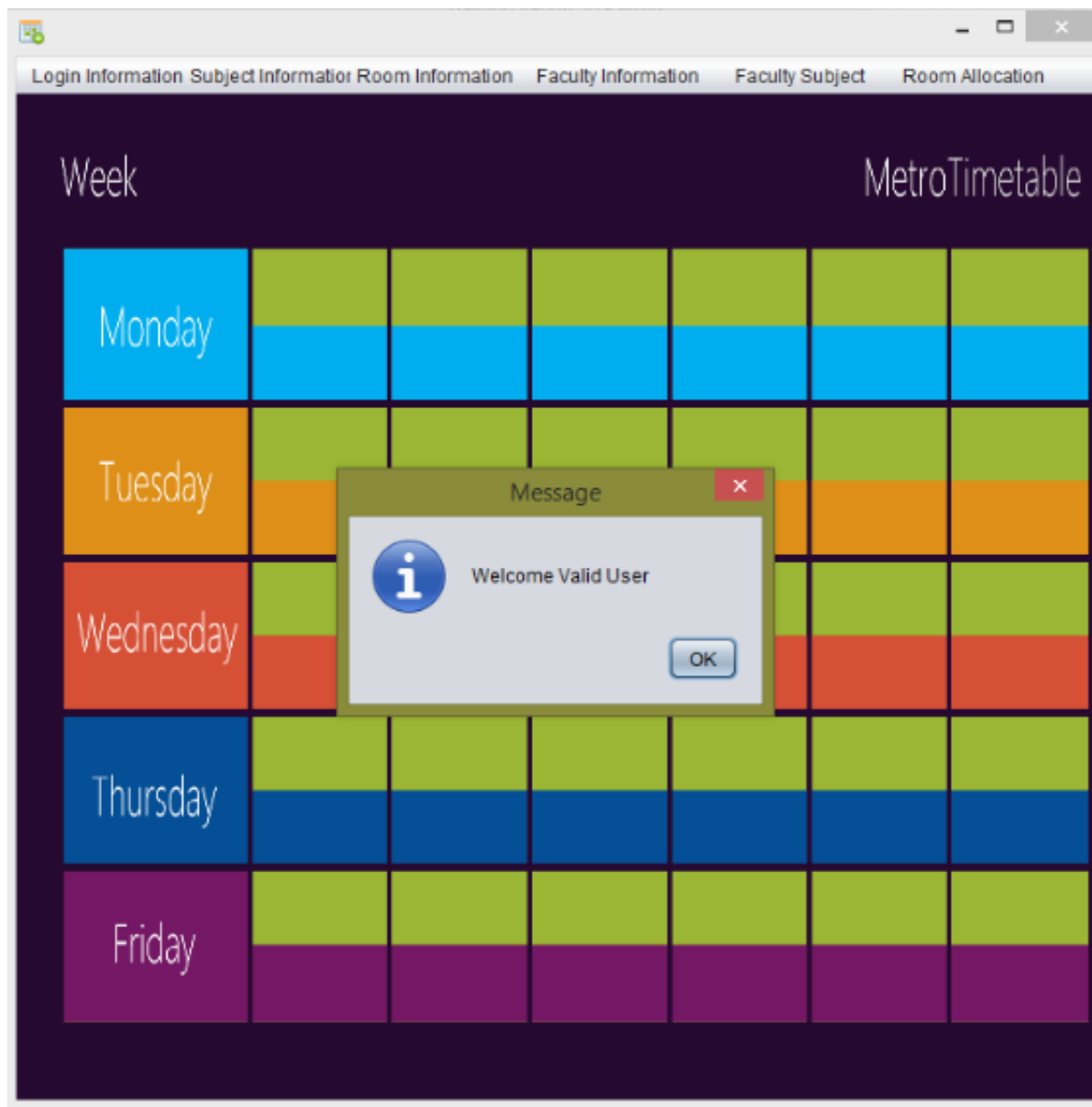
### Screenshot 1: Login Screen



*Fig 1. Login Screen*

This is login page of the Time Table Management Project. When you enter the correct password then you will login in this project. And After three false tries, the project automatically terminates.

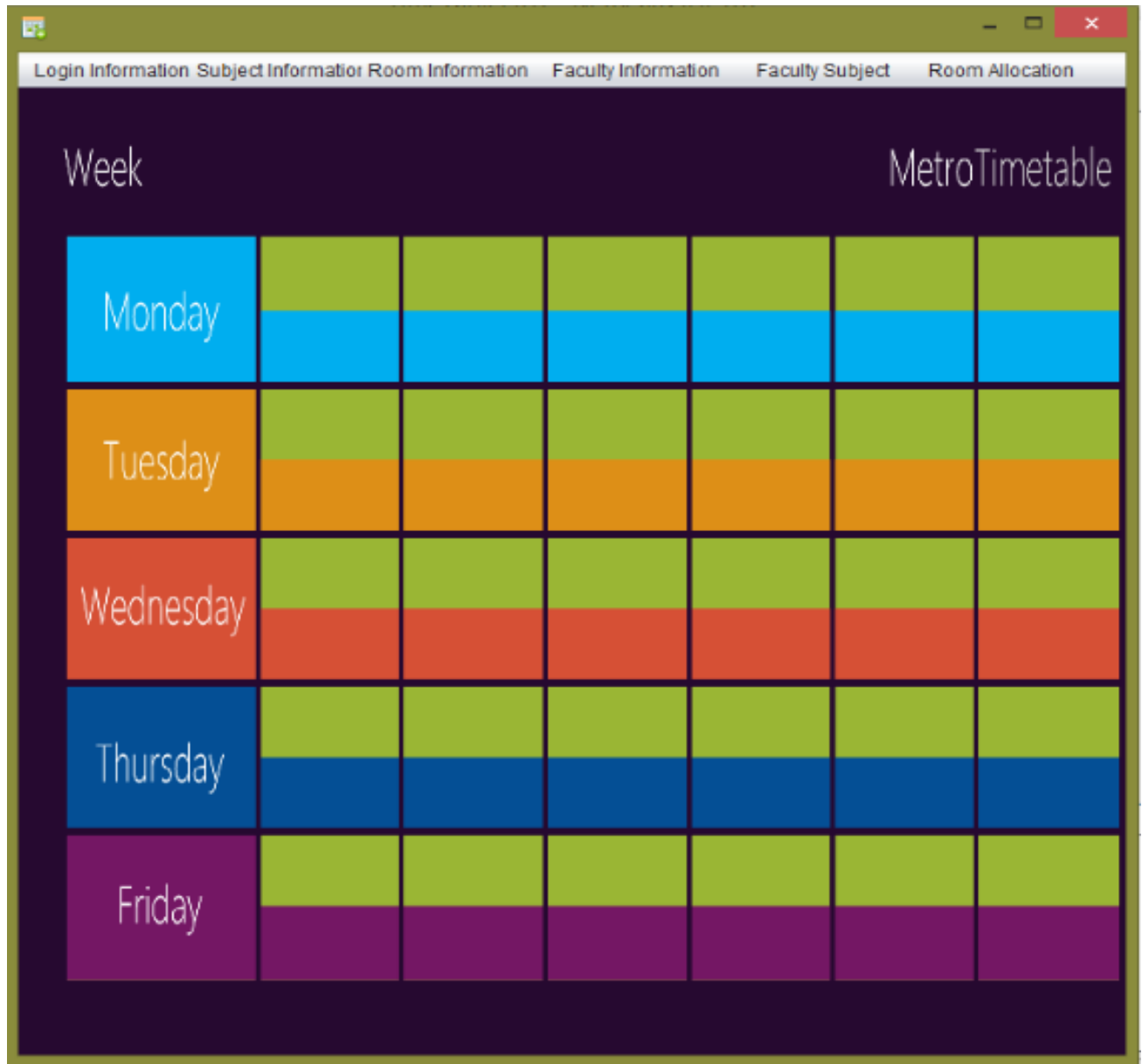
## Screenshot 2: Welcome Valid User Pop up



*Fig 2. Valid User Pop Up*

This is the front page that appears only when a valid user has logged in to the project. Otherwise the page will not open. After that on clicking on “OK”, User can enjoy the project.

### Screenshot 3: Main Screen Page

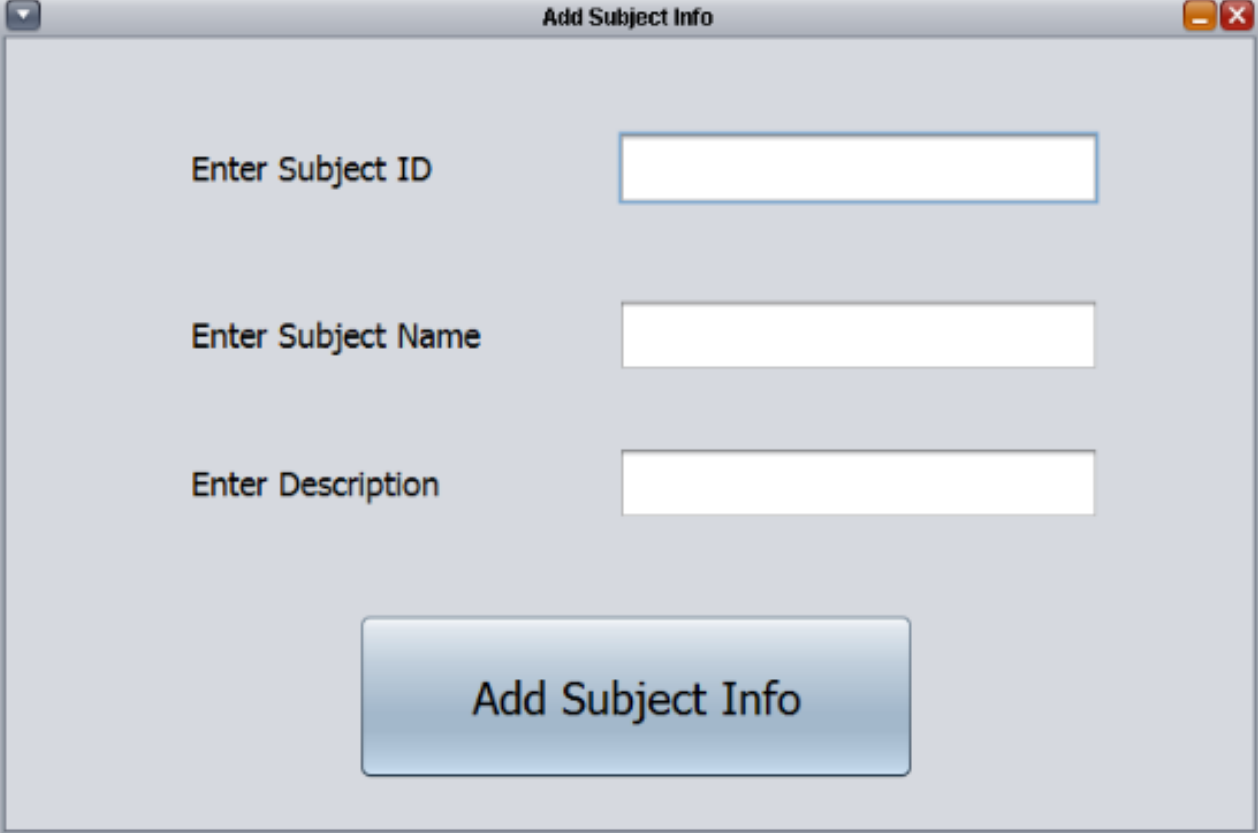


*Fig 3.Main Screen*

This is the main front page after login. Now user can make use of the various menus and can explore the project in whatever way he wants to.



#### Screenshot 4: Add Subject Information Page

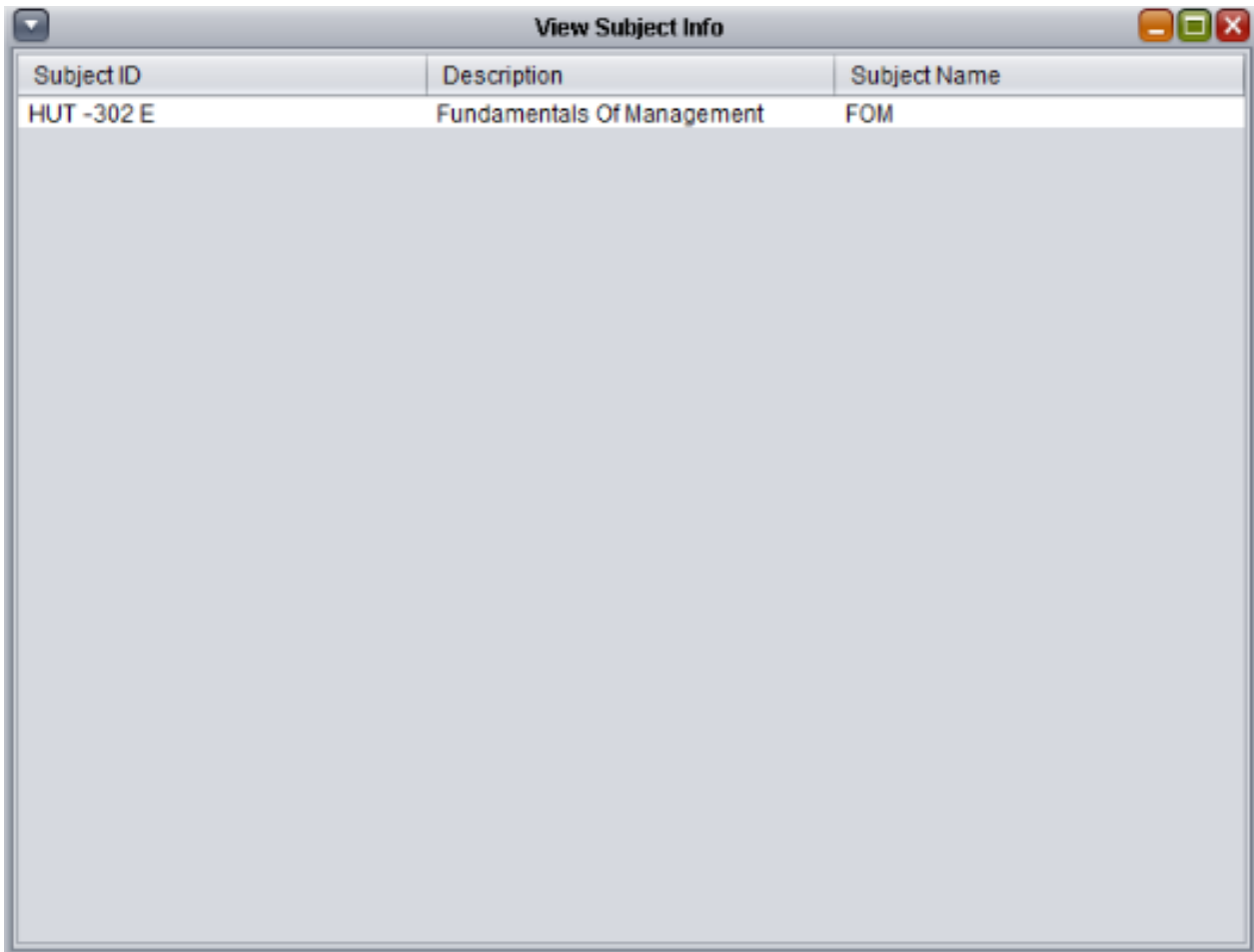


The screenshot shows a window titled "Add Subject Info" with a standard Windows-style title bar (minimize, maximize, close buttons). The window has a light gray background. It contains three text input fields arranged vertically. The first field is labeled "Enter Subject ID", the second "Enter Subject Name", and the third "Enter Description". Below these fields is a large, rounded rectangular button with a blue gradient and the text "Add Subject Info".

*Fig 4. Add Subject Information Page*

This page is used to add the subject information of the student. After adding the information the pop up appears and shows the message of Record Inserted.

## Screenshot 5: View Subject Information Page



The screenshot shows a window titled "View Subject Info" with standard window controls (minimize, maximize, close) in the top right corner. Inside the window is a table with three columns: "Subject ID", "Description", and "Subject Name". The table contains one data row with the following values: "HUT -302 E" for Subject ID, "Fundamentals Of Management" for Description, and "FOM" for Subject Name. The rest of the window area is a light gray background.

Subject ID	Description	Subject Name
HUT -302 E	Fundamentals Of Management	FOM

*Fig 5. View Subject Information Page*

This page is used to view the Subject Information entered by the user. In this user can edit the table content by deleting it as well as by updating it.

## Screenshot 6: Add Room Information Page

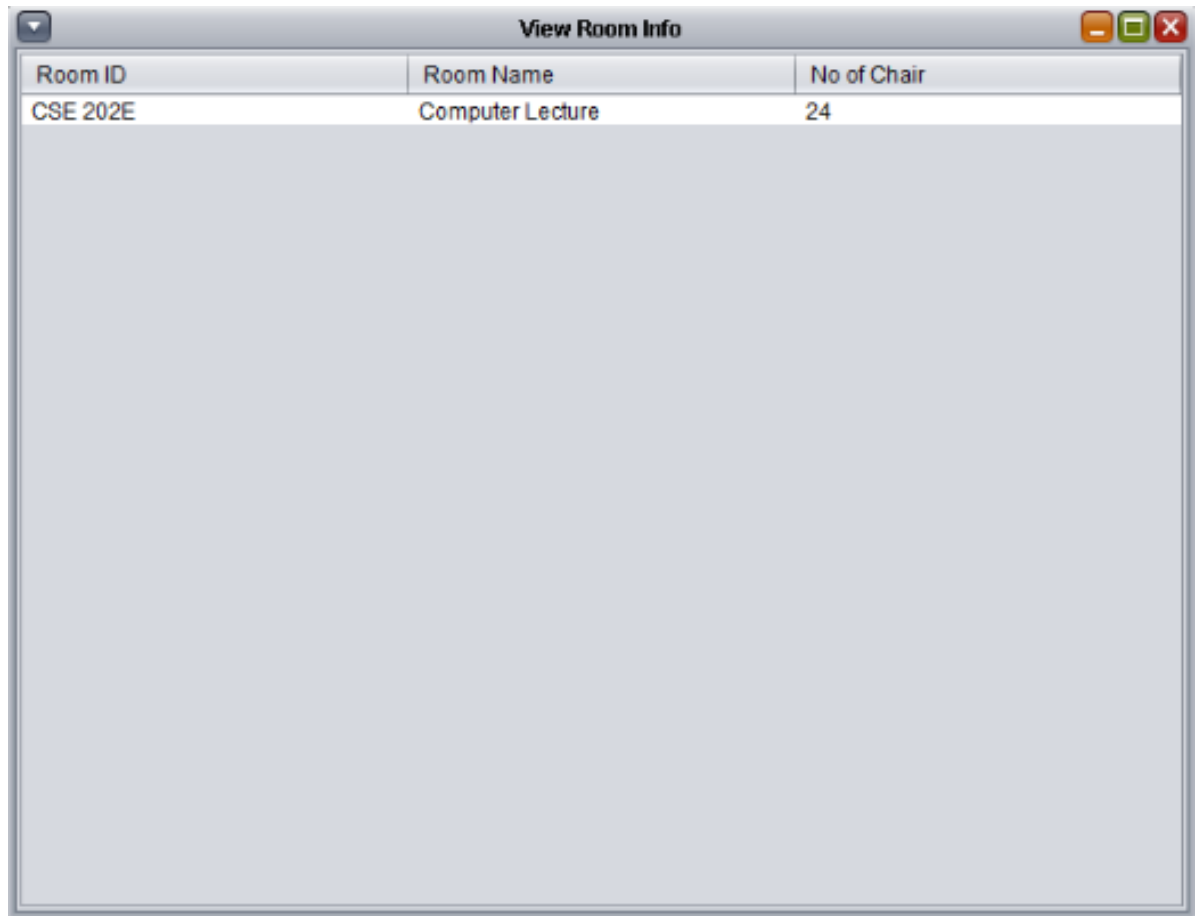


The screenshot displays a window titled "Add Room Info" with a standard operating system window frame (minimize, maximize, close buttons). Inside the window, there are three input fields arranged vertically. The first field is labeled "Enter Room ID:" and contains a single vertical line cursor. The second field is labeled "Enter Room Name:" and is empty. The third field is labeled "Enter No of Chairs:" and is empty. Below these fields is a large, light green button with the text "Add Room Info" in black.

*Fig 6. Add Room Information Page*

This page shows the portal where user can add room information where the faculty can add as a lecture or tutorial room according to their schedule. After adding the message pops up that the record inserted. In this the No of chairs shows the number of strength in the class.

## Screenshot 7: View Room Information Page



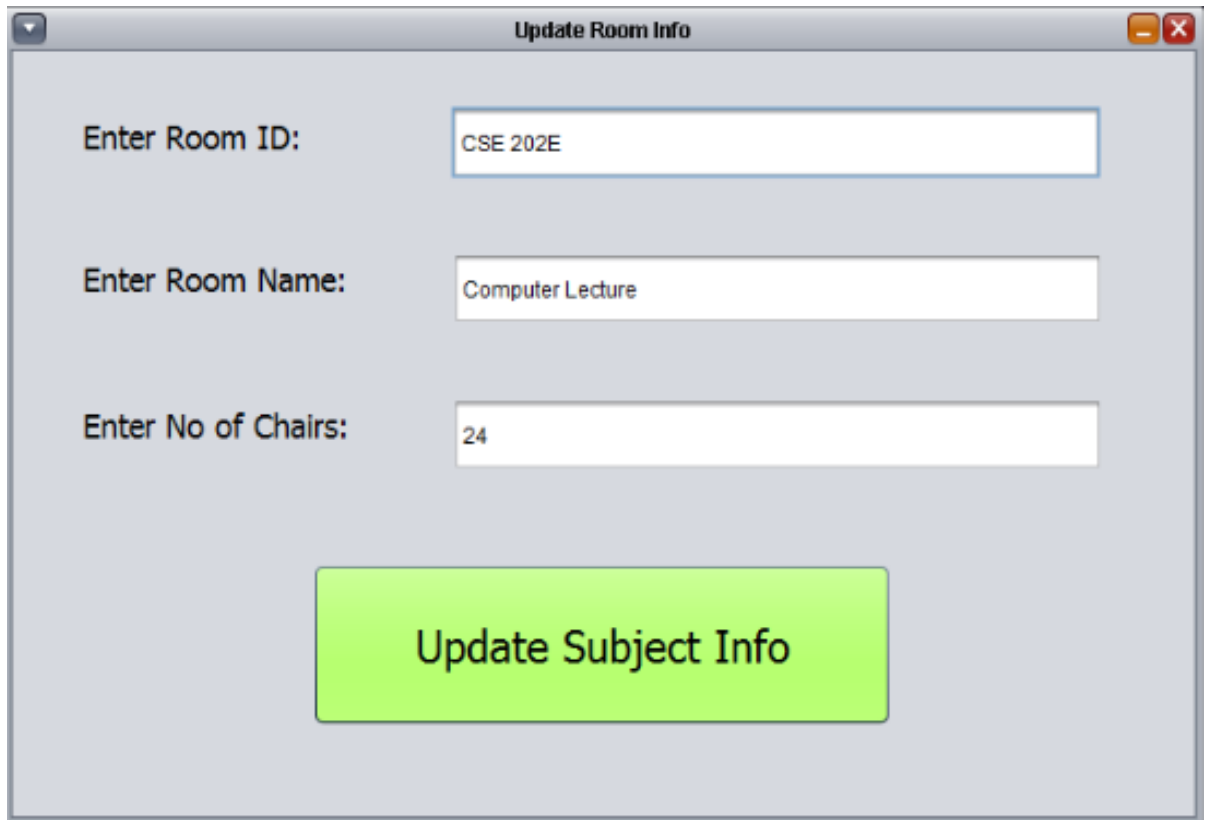
The screenshot shows a window titled "View Room Info" with a standard macOS-style title bar (minimize, maximize, close buttons). Inside the window is a table with three columns: "Room ID", "Room Name", and "No of Chair". The table contains one data row with the values "CSE 202E", "Computer Lecture", and "24". The rest of the window area is a light gray background.

Room ID	Room Name	No of Chair
CSE 202E	Computer Lecture	24

*Fig 7. View Room Information Page*

This page appears when the user want to see the table of information one entered after inserting the value. In this the user has an authority to either delete the data as well as to update the data.

## Screenshot 8: Update Room Information Page



The screenshot shows a window titled "Update Room Info" with a standard macOS-style title bar (minimize, maximize, close buttons). The window has a light gray background. It contains three input fields, each with a label to its left:

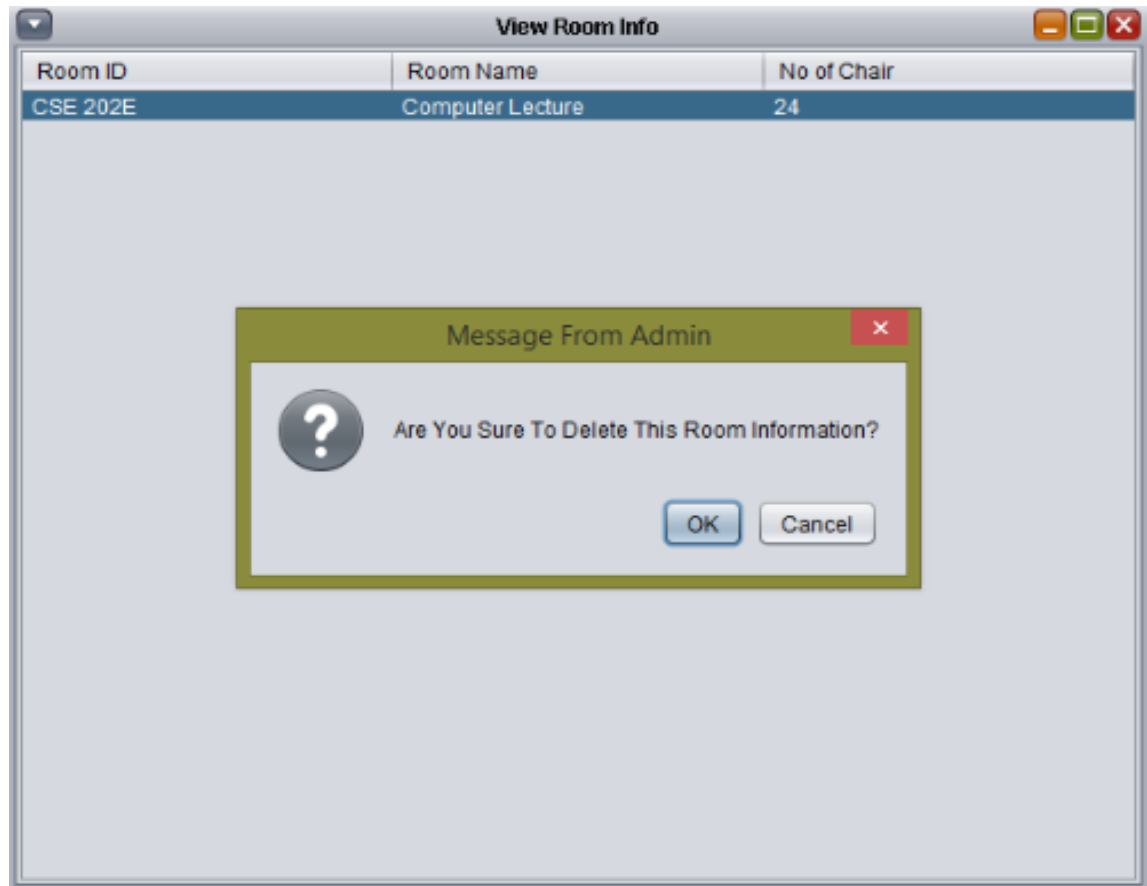
- Enter Room ID:** The input field contains the text "CSE 202E".
- Enter Room Name:** The input field contains the text "Computer Lecture".
- Enter No of Chairs:** The input field contains the text "24".

Below the input fields is a large, rectangular green button with the text "Update Subject Info" in black, centered on the button.

*Fig 8. Update Room Information Page*

This page appears when the user who viewed the information he entered does not seem satisfied with the data. So he can update the data and while updating the data he will be displayed with the data he entered so that it becomes easier for the user to update the sheet.

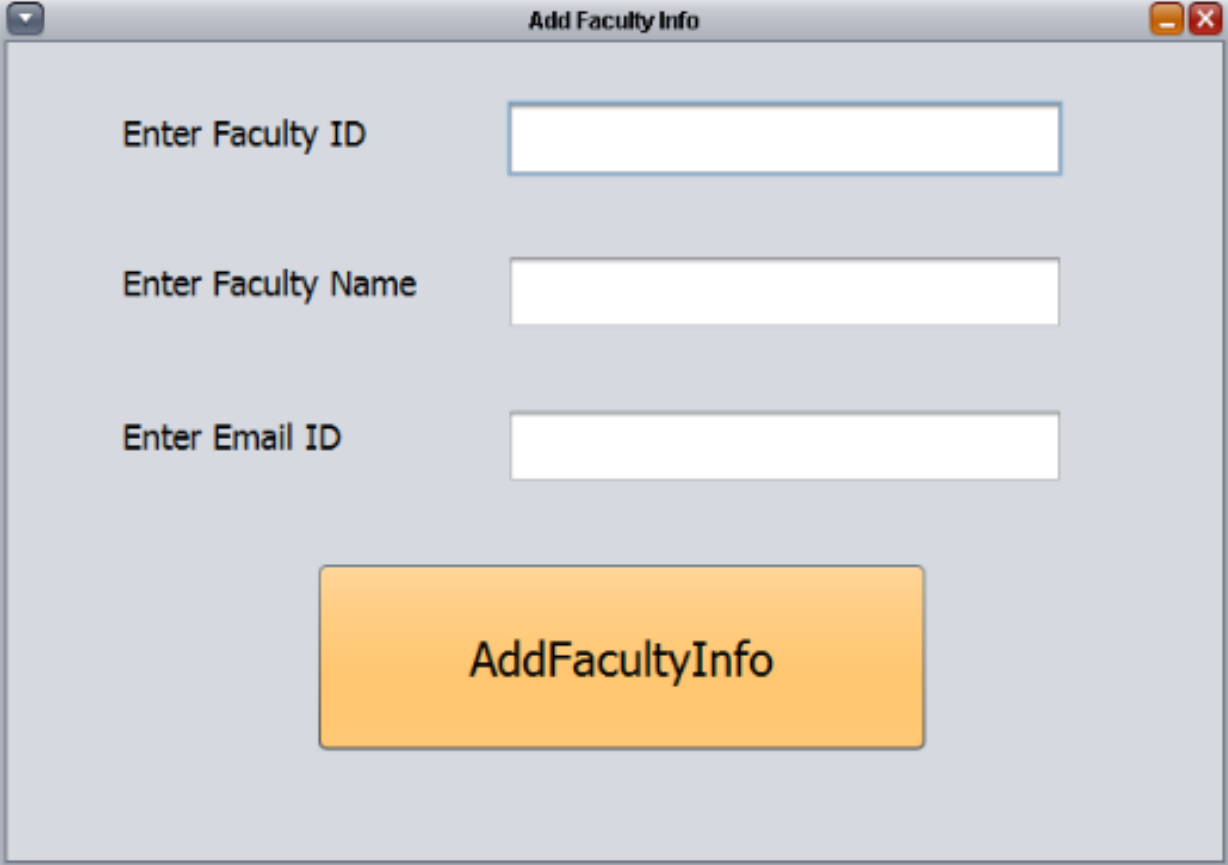
## Screenshot 9: Delete Room Information Pop Up



*Fig 9. Delete Room Information Pop Up*

This page appears when the user who viewed his information he entered does not seem satisfied with the information he entered and now he wants to delete it so he can directly delete it from the view table.

## Screenshot 10: Add Faculty Information Page

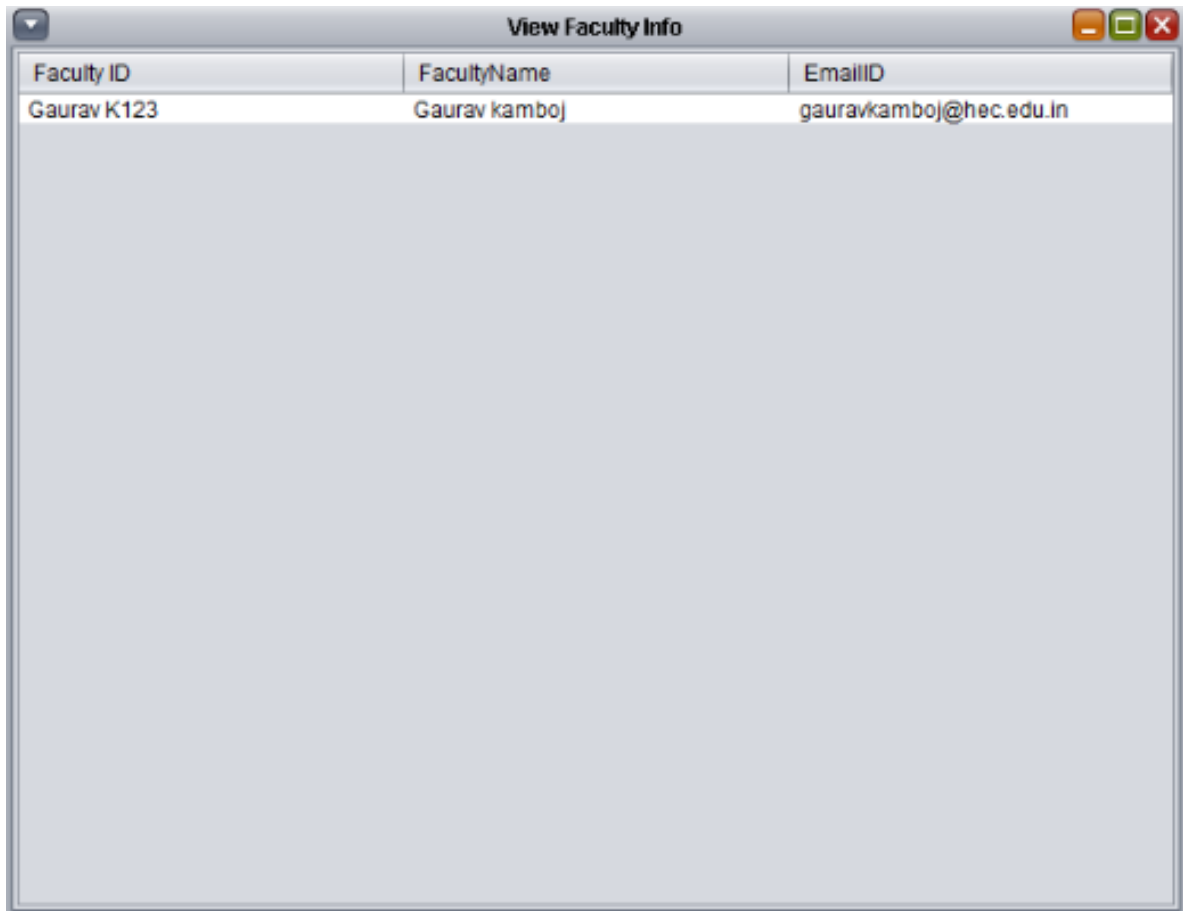


The screenshot shows a window titled "Add Faculty Info" with a light gray background. It contains three input fields stacked vertically, each with a label to its left: "Enter Faculty ID", "Enter Faculty Name", and "Enter Email ID". Below these fields is a large orange button with the text "AddFacultyInfo". The window has standard OS window controls (minimize, maximize, close) in the top right corner.

*Fig 10. Add Faculty Information Page*

This page appears when the user wants to add faculty information along with their email id.

## Screenshot 11: View Faculty Information Page



The screenshot shows a window titled "View Faculty Info" with standard Windows-style window controls (minimize, maximize, close) in the top right corner. Inside the window is a table with three columns: "Faculty ID", "FacultyName", and "EmailID". The table contains a single data row for a faculty member named Gaurav Kamboj.

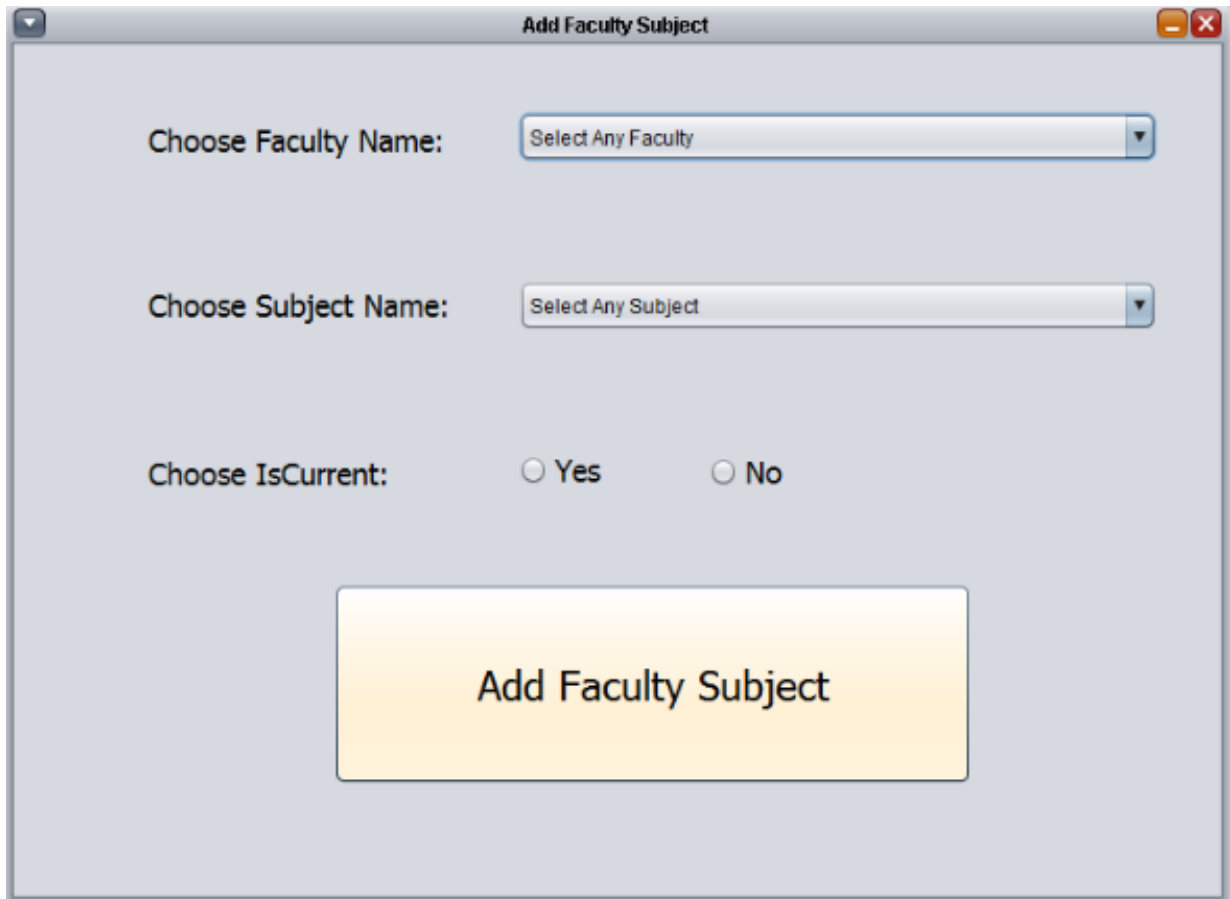
Faculty ID	FacultyName	EmailID
Gaurav K123	Gaurav kamboj	gauravkamboj@hec.edu.in

*Fig 11. View Faculty Information Page*

This page appears when the user who entered the information in the add sheet wants to see the information he entered. So he can view the information here and in this he has authority to edit data entered by updating one's information as well as he can delete it.



## Screenshot 12: Add Faculty Subject Information



Choose Faculty Name:

Choose Subject Name:

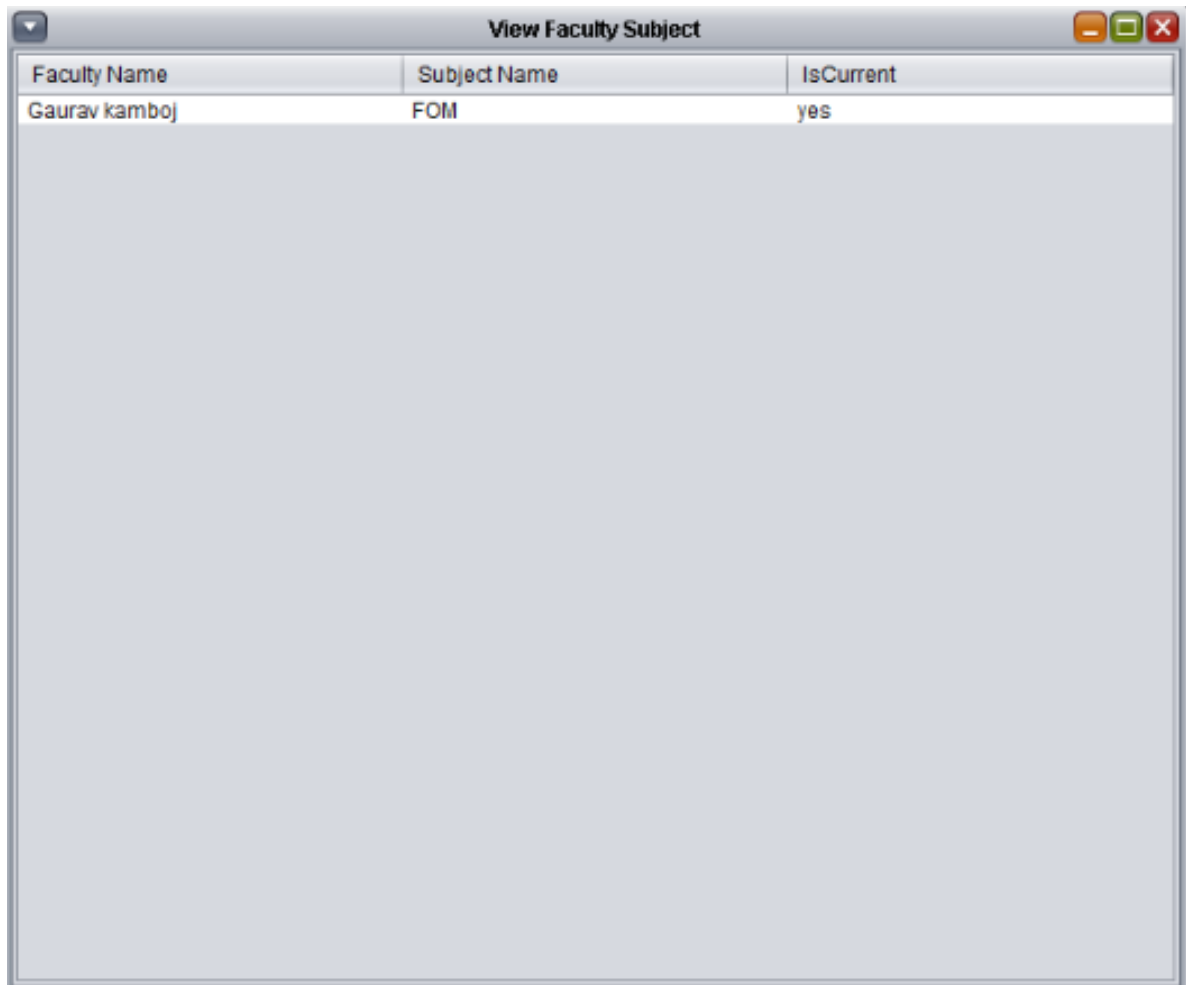
Choose IsCurrent: ☐ Yes ☐ No

**Add Faculty Subject**

*Fig 12. Add Faculty Subject Information Page*

This page shows the portal where user can add faculty subject he teaches currently so that it can be set into the time table.

### Screenshot 13: View Faculty Subject Information Page



Faculty Name	Subject Name	IsCurrent
Gaurav kamboj	FOM	yes

*Fig 13. View Faculty Subject Information Page*

This page shows the portal where the user who entered his information can view here as well as edit it by updating it and also he/she can delete that information entered and add with new one also.

## Screenshot 14: Search Faculty Subject Page

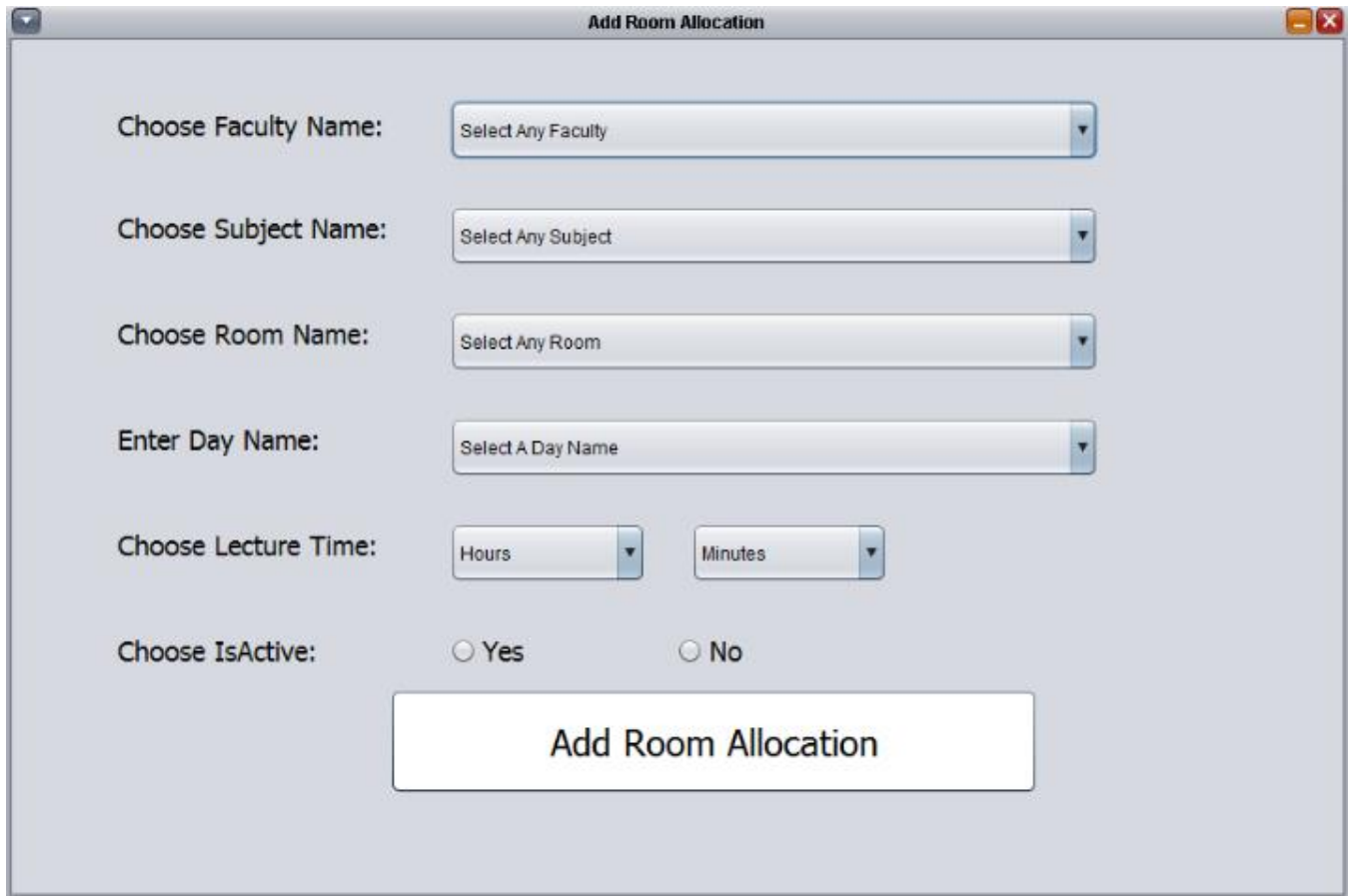
The screenshot shows a web application window titled "Search Faculty Subject". It contains three dropdown menus on the left for filtering search results: "Gaurav kamboj ( Gaurav K123 )", "FOM ( HUT -302 E )", and "YES". To the right of each dropdown is a button labeled "Show Faculty Info", "Show Subject Info", and "Show Yes/No Info" respectively. Below these controls is a table with three columns: "Faculty Name", "Subject Name", and "IsCurrent". The table contains one row of data: "Gaurav kamboj", "FOM", and "yes".

Faculty Name	Subject Name	IsCurrent
Gaurav kamboj	FOM	yes

*Fig 14. Search Page Portal of Faculty Subject*

This page shows the portal where the user can view his/her details by more options he has. For example if the user entered faculty and subject along with yes then only the information with yes will be displayed otherwise if he chooses no then information with no will be displayed.

## Screenshot 15: Add Room Allocation Page



The screenshot shows a web browser window titled "Add Room Allocation". The form contains the following fields and controls:

- Choose Faculty Name:** A dropdown menu with the text "Select Any Faculty".
- Choose Subject Name:** A dropdown menu with the text "Select Any Subject".
- Choose Room Name:** A dropdown menu with the text "Select Any Room".
- Enter Day Name:** A dropdown menu with the text "Select A Day Name".
- Choose Lecture Time:** Two dropdown menus, one labeled "Hours" and one labeled "Minutes".
- Choose IsActive:** Two radio buttons labeled "Yes" and "No".
- Add Room Allocation:** A large rectangular button at the bottom center of the form.

*Fig 15. Add Room Allocation Page*

This page shows the portal where user can add information of which room is allotted to a faculty with the information being filled along with the time of the particular lecture followed by the next faculty assuming it being in the same room with some other subject.

### Screenshot 16: View Room Allocation Information

View Room Allocation					
Faculty Name	Subject Name	Room Name	Day Name	Lecture Time	IsActive
Gaurav kamboj	FOM	Computer Lectur...	Tuesday	07:10:00	yes

Fig 16. View Room Allocation Information Page

This page shows the portal where the user can view his/her entered information and also one can edit it by updating as well as deleting it and adding the new one.

## Screenshot 17: Search Room Allocation Page

Faculty Name	Subject Name	Room Name	IsActive
Gaurav kamboj	FOM	Computer Lecture [ CSE 202E ]	yes

*Fig 17. Search Page Portal of Room Allocation*

This page shows the portal where the user can view the particular information one wants to be viewed and if one wants to update or delete it then one can open the view room allocation table and delete or update accordingly there.

## **FUTURE SCOPE**

- This project can be used in the college according to their requirement. And Time table can be made with ease and can be distributed if requirement increases.
- This project has a lot of restrictions that the user cannot add invalid data. So this can be beneficial for the faculty members.
- In this project the provision of printing can be also be accomplished successfully.
- In this project the user either faculty or student can edit the data by updating or deleting the record from the table.

## CONCLUSION

- ▶ Hence, Time table management plays an important role in every organisation's and individual's life. Through this project, we can easily manage the time table of various regular classes in an organisation.
- ▶ The primary mission of the Time table management is to ensure that all the people in an organisation whether school or college can make the best use of it by creating the time table of various classes with an ease without any strong efforts.



## APPENDIX

### **Program code:**

#### **Login Screen -**

```
package model.to;

public class LoginInfoTO {

    private String UserName , Password , RoleName;

    public String getUserName() {

        return UserName;

    }

    public void setUserName(String UserName) {

        this.UserName = UserName;

    }

    public String getPassword() {

        return Password;

    }

    public void setPassword(String Password) {

        this.Password = Password;

    }

    public String getRoleName() {

        return RoleName;

    }

    public void setRoleName(String RoleName) {

        this.RoleName = RoleName;

    }

    public String toString() {
```

```
        return UserName;
    }
}
```

---

```
package model.dao;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import model.to.LoginInfoTO;
```

```
public class LoginInfoDAO {

    private String errorMessage;

    public String getErrorMessage() {

        return errorMessage;

    }
}
```

```
    public LoginInfoTO checkLogin(String username , String password){ /* In this code Check Login
function is applied.*/
```

```
        try{

            String query = "select username ,password,rolename from logininfo where username=? and
password=?";

            PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

            stmt.setString(1,username);

            stmt.setString(2,password);

            LoginInfoTO data = null;

            ResultSet rs = stmt.executeQuery();
```

```

        if(rs.next()){

            data = new LoginInfoTO();

            data.setUserName(rs.getString(1));

            data.setPassword(rs.getString(2));

            data.setRoleName(rs.getString(3));

        }

        rs.close();

        stmt.close();

        return data;

    }catch(Exception ex){

        errorMessage = ex.getMessage();

        System.out.println(ex.getMessage());

        return null;

    }

}

public boolean insertRecord (LoginInfoTO data){ /* For Inserting the record. */

    try {

        String query ="insert into logininfo(Username,Password,RoleName) values(?,?,?)";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1, data.getUserName());

        stmt.setString(2, data.getPassword());

        stmt.setString(3, data.getRoleName());

        boolean ans = stmt.executeUpdate() > 0;

        stmt.close();

        return ans;

    }catch(Exception ex){

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return false;

    }

}

```

```

public boolean updateRecord(LoginInfoTO data) { /* For Updating the record.*/

    try {

        String query = "update logininfo set password=?,rolename=? where username=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1, data.getPassword());

        stmt.setString(2, data.getUserName());

        stmt.setString(3, data.getRoleName());

        boolean ans = stmt.executeUpdate() > 0;

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return false;

    }

}

```

```

public boolean deleteRecord(String username) { /* For Deleting the record.*/

    try {

        String query = "delete from logininfo where username=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1, username);

        boolean ans = stmt.executeUpdate() > 0;

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

    }
}

```

```

        errorMessage = ex.getMessage();

        return false;

    }

}

public LoginInfoTO getRecord(String username) { /* For Getting the record through Username.*/

    try {

        String query = "select username , password , rolename from logininfo where username=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1, username);

        LoginInfoTO ans = null;

        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            ans = new LoginInfoTO();

            ans.setUserName(rs.getString(1));

            ans.setPassword(rs.getString(2));

        }

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return null;

    }

}

public ArrayList<LoginInfoTO> getAllRecord() { /* For Getting All Records.*/

    try {

```

```

String query = "select username , password , rolename from logininfo";

PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

ArrayList<LoginInfoTO> data = null;

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

    rs.beforeFirst();

    data = new ArrayList<LoginInfoTO>();

    while (rs.next()) {

        LoginInfoTO ans = new LoginInfoTO();

        ans.setUserName(rs.getString(1));

        ans.setPassword(rs.getString(2));

        ans.setRoleName(rs.getString(3));

        data.add(ans);

    }

}

stmt.close();

return data;

} catch (Exception ex) {

    System.out.println(ex.getMessage());

    errorMessage = ex.getMessage();

    return null;

}

}

}

package model.to;

public class SubjectInfoTO {

```

```

    private String SubjectID , Description , SubjectName;

    public String getSubjectID() {

        return SubjectID;

    }

    public void setSubjectID(String SubjectID) {

        this.SubjectID = SubjectID;

    }

    public String getDescription() {

        return Description;

    }

    public void setDescription(String Description) {

        this.Description = Description;

    }

    public String getSubjectName() {

        return SubjectName;

    }

    public void setSubjectName(String SubjectName) {

        this.SubjectName = SubjectName;

    }

    public String toString(){

        return SubjectName + " ( " + SubjectID + " ) ";

    }

}

```

---

```

package model.dao;

```

```

import java.sql.PreparedStatement;

```

```

import java.sql.ResultSet;

```

```

import java.util.ArrayList;

import model.to.SubjectInfoTO;


public class SubjectInfoDAO {

private String errorMessage;

public String geterrorMessage() {

    return errorMessage;

}

public boolean insertRecord (SubjectInfoTO data){

    try {

        String query ="insert into subjectinfo(SubjectID,Description,SubjectName) values(?,?,?)";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1, data.getSubjectID());

        stmt.setString(2, data.getDescription());

        stmt.setString(3, data.getSubjectName());

        boolean ans = stmt.executeUpdate() > 0;

        stmt.close();

        return ans;

    }catch(Exception ex){

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return false;

    }

}

public boolean updateRecord(SubjectInfoTO data) {

    try {

```



```

String query = "update subjectinfo set description=? subjectname=? where subjectid=?";

PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

stmt.setString(1, data.getDescription());

stmt.setString(2, data.getSubjectName());

stmt.setString(3, data.getSubjectID());

boolean ans = stmt.executeUpdate() > 0;

stmt.close();

return ans;

} catch (Exception ex) {

    System.out.println(ex.getMessage());

    errorMessage = ex.getMessage();

    return false;

}

}

public boolean deleteRecord(String subjectid) {

    try {

        String query = "delete from subjectinfo where subjectid=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1, subjectid);

        boolean ans = stmt.executeUpdate() > 0;

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return false;
    }
}

```

```

    }
}

public SubjectInfoTO getRecord(String subjectid) {

    try {

        String query = "select subjectid , description , subjectname from subjectinfo where subjectid=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1, subjectid);

        SubjectInfoTO ans = null;

        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            ans = new SubjectInfoTO();

            ans.setSubjectID(rs.getString(1));

            ans.setDescription(rs.getString(2));

            ans.setSubjectName(rs.getString(3));

        }

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return null;

    }

}

public ArrayList<SubjectInfoTO> getAllRecord() {

    try {

        String query = "select subjectid , description , subjectname from subjectinfo";

```

```

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        ArrayList<SubjectInfoTO> data = null;

        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            rs.beforeFirst();

            data = new ArrayList<SubjectInfoTO>();

            while (rs.next()) {

                SubjectInfoTO ans = new SubjectInfoTO();

                ans.setSubjectID(rs.getString(1));

                ans.setDescription(rs.getString(2));

                ans.setSubjectName(rs.getString(3));

                data.add(ans);

            }

        }

        stmt.close();

        return data;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return null;

    }

}

public ArrayList<SubjectInfoTO> getAllRecordFacultyWise(String facultyid) {

    try {

        String query = "select subjectid , description , subjectname from subjectinfo where subjectid in (
select subjectid from facultysubject where facultyid=? and isCurrent='yes' )";

```

```

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1,facultyid);

        ArrayList<SubjectInfoTO> data = null;

        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            rs.beforeFirst();

            data = new ArrayList<SubjectInfoTO>();

            while (rs.next()) {

                SubjectInfoTO ans = new SubjectInfoTO();

                ans.setSubjectID(rs.getString(1));

                ans.setDescription(rs.getString(2));

                ans.setSubjectName(rs.getString(3));

                data.add(ans);

            }

        }

        stmt.close();

        return data;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return null;

    }

}
}
}

```

```

package model.to;

```

```

public class RoomInfoTO {

    private String RoomID , RoomName;

    public String getRoomID() {

        return RoomID;

    }

    public void setRoomID(String RoomID) {

        this.RoomID = RoomID;

    }

    public String getRoomName() {

        return RoomName;

    }

    public void setRoomName(String RoomName) {

        this.RoomName = RoomName;

    }

    public Integer getNoofChair() {

        return NoofChair;

    }

    public void setNoofChair(Integer NoofChair) {

        this.NoofChair = NoofChair;

    }

    private Integer NoofChair;

    public String toString(){

        return RoomName + " ( " + RoomID + " ) ";

    }

}}

package model.dao;

import java.sql.PreparedStatement;

```

```
import java.sql.ResultSet;
```

```
import java.util.ArrayList;
```

```
import model.to.RoomInfoTO;
```

```
public class RoomInfoDAO {
```

```
    private String errorMessage;\
```

```
    public String getErrorMessage() {
```

```
        return errorMessage;
```

```
    }
```

```
    public boolean insertRecord(RoomInfoTO data) {
```

```
        try {
```

```
            String query = "insert into roominfo(roomid,roomname,noofchair) values(?,?,?)";
```

```
            PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);
```

```
            stmt.setString(1, data.getRoomID());
```

```
            stmt.setString(2, data.getRoomName());
```

```
            stmt.setInt(3, data.getNoofChair());
```

```
            boolean ans = stmt.executeUpdate() > 0;
```

```
            stmt.close();
```

```
            return ans;
```

```
        } catch (Exception ex) {
```

```
            System.out.println(ex.getMessage());
```

```
            errorMessage = ex.getMessage();
```

```
            return false;
```

```
        }
```

```
    }
```

```
    public boolean updateRecord(RoomInfoTO data) {
```

```

try {

    String query = "update roominfo set roomname=? , noofchair=? where roomid=?";

    PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

    stmt.setString(1, data.getRoomName());

    stmt.setString(3, data.getRoomID());

    stmt.setInt(2, data.getNoofChair());

    boolean ans = stmt.executeUpdate() > 0;

    stmt.close();

    return ans;

} catch (Exception ex) {

    System.out.println(ex.getMessage());

    errorMessage = ex.getMessage();

    return false;

}

}

public boolean deleteRecord(String roomid) {

    try {

        String query = "delete from roominfo where roomid=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1, roomid);

        boolean ans = stmt.executeUpdate() > 0;

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();
    }
}

```

```

        return false;
    }
}

public RoomInfoTO getRecord(String roomid) {

    try {

        String query = "select roomid , roomname from roominfo where roomid=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1, roomid);

        RoomInfoTO ans = null;

        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            ans = new RoomInfoTO();

            ans.setRoomID(rs.getString(1));

            ans.setRoomName(rs.getString(2));

            ans.setNoofChair(rs.getInt(3));

        }

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return null;

    }

}

public ArrayList<RoomInfoTO> getAllRecord() {

    try {

```



```

String query = "select roomid , roomname , noofchair from roominfo";

PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

ArrayList<RoomInfoTO> data = null;

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

    rs.beforeFirst();

    data = new ArrayList<RoomInfoTO>();

    while (rs.next()) {

        RoomInfoTO ans = new RoomInfoTO();

        ans.setRoomID(rs.getString(1));

        ans.setRoomName(rs.getString(2));

        ans.setNoofChair(rs.getInt(3));

        data.add(ans);

    }

}

stmt.close();

return data;

} catch (Exception ex) {

    System.out.println(ex.getMessage());

    errorMessage = ex.getMessage();

    return null;

}

}}

```

package model.to;

public class FacultyInfoTO {

```
private String FacultyID , FacultyName , EmailID;

public String getFacultyID() {

    return FacultyID;

}

public void setFacultyID(String FacultyID) {

    this.FacultyID = FacultyID;

}

public String getFacultyName() {

    return FacultyName;

}

public void setFacultyName(String FacultyName) {

    this.FacultyName = FacultyName;

}

public String getEmailID() {

    return EmailID;

}

public void setEmailID(String EmailID) {

    this.EmailID = EmailID;

}

public String toString() {

    return FacultyName + " ( " + FacultyID + " ) ";

}

}
```

```
package model.dao;
```

```
import java.sql.PreparedStatement;
```

```

import java.sql.ResultSet;

import java.util.ArrayList;

import model.to.FacultyInfoTO;

public class FacultyInfoDAO {

    private String errorMessage;

    public String geterrorMessage() {

        return errorMessage;

    }

    public boolean insertRecord (FacultyInfoTO data){

        try {

            String query ="insert into facultyinfo(facultyid,facultyname,emailid) values(?,?,?)";

            PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

            stmt.setString(1, data.getFacultyID());

            stmt.setString(2, data.getFacultyName());

            stmt.setString(3, data.getEmailID());

            boolean ans = stmt.executeUpdate() > 0;

            stmt.close();

            return ans;

        }catch(Exception ex){

            System.out.println(ex.getMessage());

            errorMessage = ex.getMessage();

            return false;

        }

    }

    public boolean updateRecord(FacultyInfoTO data) {

        try {

```

```

String query = "update facultyinfo set facultyname=? , emailid=? where facultyid=?";

PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

stmt.setString(1, data.getFacultyName());

stmt.setString(2, data.getEmailID());

stmt.setString(3, data.getFacultyID());

boolean ans = stmt.executeUpdate() > 0;

stmt.close();

return ans;

} catch (Exception ex) {

    System.out.println(ex.getMessage());

    errorMessage = ex.getMessage();

    return false;

}

}

public boolean deleteRecord(String facultyid) {

    try {

        String query = "delete from facultyinfo where facultyid=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1, facultyid);

        boolean ans = stmt.executeUpdate() > 0;

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return false;

    }

}

```

```

    }
}

public FacultyInfoTO getRecord(String facultyid) {

    try {

        String query = "select facultyid , facultyname, emailid from facultyinfo where facultyid=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1, facultyid);

        FacultyInfoTO ans = null;

        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            ans = new FacultyInfoTO();

            ans.setFacultyID(rs.getString(1));

            ans.setFacultyName(rs.getString(2));

            ans.setEmailID(rs.getString(3));

        }

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return null;

    }

}

public ArrayList<FacultyInfoTO> getAllRecord() {

    try {

        String query = "select facultyid , facultyname , emailid from facultyinfo";

```

```

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        ArrayList<FacultyInfoTO> data = null;

        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            rs.beforeFirst();

            data = new ArrayList<FacultyInfoTO>();

            while (rs.next()) {

                FacultyInfoTO ans = new FacultyInfoTO();

                ans.setFacultyID(rs.getString(1));

                ans.setFacultyName(rs.getString(2));

                ans.setEmailID(rs.getString(3));

                data.add(ans);

            }

        }

        stmt.close();

        return data;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return null;

    }

}

```

package model.to;

public class FacultySubjectTO {

```
private String FacultyID , SubjectID;

private String FacultyName;

public String getFacultyName() {

    return FacultyName;

}

public void setFacultyName(String FacultyName) {

    this.FacultyName = FacultyName;

}

public String getSubjectName() {

    return SubjectName;

}

public void setSubjectName(String SubjectName) {

    this.SubjectName = SubjectName;

}

private String SubjectName;

public String getFacultyID() {

    return FacultyID;

}

public void setFacultyID(String FacultyID) {

    this.FacultyID = FacultyID;

}

public String getSubjectID() {

    return SubjectID;

}

public void setSubjectID(String SubjectID) {

    this.SubjectID = SubjectID;
```

```
}

public String getIsCurrent() {

    return IsCurrent;

}

public void setIsCurrent(String IsCurrent) {

    this.IsCurrent = IsCurrent;

}

public Integer getSrNo() {

    return SrNo;

}

public void setSrNo(Integer SrNo) {

    this.SrNo = SrNo;

}

private String IsCurrent;

private Integer SrNo;

}
```

-----

```
package model.dao;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import model.to.FacultySubjectTO;

public class FacultySubjectDAO {

    private String errorMessage;
```



```

public String getErrorMessage() {

    return errorMessage;

}

public boolean insertRecord(FacultySubjectTO data) {

    try {

        String query = "insert into facultysubject(facultyid,subjectid,iscurrent) values(?,?,?)";
        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);
        stmt.setString(1, data.getFacultyID());
        stmt.setString(2, data.getSubjectID());
        stmt.setString(3, data.getIsCurrent());

        boolean ans = stmt.executeUpdate() > 0;
        stmt.close();
        return ans;
    } catch (Exception ex) {

        System.out.println(ex.getMessage());
        errorMessage = ex.getMessage();
        return false;
    }
}

public boolean updateRecord(FacultySubjectTO data) {

    try {

        String query = "update facultysubject set facultyid=? , subjectid=? , iscurrent=? where SrNo=?";
        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);
        stmt.setString(1, data.getFacultyID());
        stmt.setString(2, data.getSubjectID());
        stmt.setString(3, data.getIsCurrent());
        stmt.setInt(4, data.getSrNo());

        boolean ans = stmt.executeUpdate() > 0;
        stmt.close();
    }
}

```

```

        return ans;
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
        errorMessage = ex.getMessage();

        return false;
    }
}

public boolean deleteRecord(int srno) {
    try {
        String query = "delete from facultysubject where srno=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setInt(1, srno);

        boolean ans = stmt.executeUpdate() > 0;

        stmt.close();

        return ans;
    } catch (Exception ex) {
        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return false;
    }
}

public boolean isTeach(String facid , String subid) {
    try {
        String query = "select * from facultysubject where facultyid=? and subjectid=? and
iscurrent='Yes'";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

```

```

        stmt.setString(1,facid);

        stmt.setString(2,subid);

        ResultSet rs = stmt.executeQuery();

        boolean ans = false;

        if(rs.next()){

            ans= true;

        }

        rs.close();

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return false;

    }

}

public FacultySubjectTO getRecord(int srno) {

    try {

        String query = "select srno , facultyid , subjectid , iscurrent from facultysubject where srno=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setInt(1, srno);

        FacultySubjectTO ans = null;

        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            ans = new FacultySubjectTO();

            ans.setSrNo(rs.getInt(1));

```

```

        ans.setFacultyID(rs.getString(2));

        ans.setSubjectID(rs.getString(3));

        ans.setIsCurrent(rs.getString(4));

    }

    stmt.close();

    return ans;

} catch (Exception ex) {

    System.out.println(ex.getMessage());

    errorMessage = ex.getMessage();

    return null;

}

}

public ArrayList<FacultySubjectTO> getAllRecord() {

    try {

        String query = "select srno , f1.facultyid , f1.subjectid , iscurrent , facultyname, s.subjectname
from facultysubject f1\n" +

        "join facultyinfo f on f1.facultyid = f.facultyid join subjectinfo s on f1.subjectid = s.subjectid;";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        ArrayList<FacultySubjectTO> data = null;

        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            rs.beforeFirst();

            data = new ArrayList<FacultySubjectTO>();

            while (rs.next()) {

                FacultySubjectTO ans = new FacultySubjectTO();

                ans.setSrNo(rs.getInt(1));

```

```

        ans.setFacultyID(rs.getString(2));

        ans.setSubjectID(rs.getString(3));

        ans.setIsCurrent(rs.getString(4));

        ans.setFacultyName(rs.getString(5));

        ans.setSubjectName(rs.getString(6));

        data.add(ans);

    }

}

stmt.close();

return data;

} catch (Exception ex) {

    System.out.println(ex.getMessage());

    errorMessage = ex.getMessage();

    return null;

}

}

public ArrayList<FacultySubjectTO> getAllRecord(String facultyid) {

    try {

        String query = "select srno , f1.facultyid , f1.subjectid , iscurrent , facultyname, s.subjectname
from facultysubject f1\n" +

        "join facultyinfo f on f1.facultyid = f.facultyid join subjectinfo s on f1.subjectid = s.subjectid where
f1.facultyid=?;";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        ArrayList<FacultySubjectTO> data = null;

        stmt.setString(1, facultyid);

        ResultSet rs = stmt.executeQuery();

```

```

        if (rs.next()) {

            rs.beforeFirst();

            data = new ArrayList<FacultySubjectTO>();

            while (rs.next()) {

                FacultySubjectTO ans = new FacultySubjectTO();

                ans.setSrNo(rs.getInt(1));

                ans.setFacultyID(rs.getString(2));

                ans.setSubjectID(rs.getString(3));

                ans.setIsCurrent(rs.getString(4));

                ans.setFacultyName(rs.getString(5));

                ans.setSubjectName(rs.getString(6));

                data.add(ans);

            }

        }

        stmt.close();

        return data;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return null;

    }

}

public ArrayList<FacultySubjectTO> getAllRecord1(String subjectid) {

    try {

        String query = "select srno , f1.facultyid , f1.subjectid , iscurrent , facultyname, s.subjectname
from facultysubject f1\n" +

```

"join facultyinfo f on f1.facultyid = f.facultyid join subjectinfo s on f1.subjectid = s.subjectid where s.subjectid=?;"

```
PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

ArrayList<FacultySubjectTO> data = null;

stmt.setString(1, subjectid);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

    rs.beforeFirst();

    data = new ArrayList<FacultySubjectTO>();

    while (rs.next()) {

        FacultySubjectTO ans = new FacultySubjectTO();

        ans.setSrNo(rs.getInt(1));

        ans.setFacultyID(rs.getString(2));

        ans.setSubjectID(rs.getString(3));

        ans.setIsCurrent(rs.getString(4));

        ans.setFacultyName(rs.getString(5));

        ans.setSubjectName(rs.getString(6));

        data.add(ans);

    }

}

stmt.close();

return data;

} catch (Exception ex) {

    System.out.println(ex.getMessage());

    errorMessage = ex.getMessage();

    return null;
```

```

    }
}

public ArrayList<FacultySubjectTO> getAllRecord2(String Iscurrent) {

    try {

        String query = "select srno , f1.facultyid , f1.subjectid , iscurrent , facultyname, s.subjectname
from facultysubject f1\n" +

        "join facultyinfo f on f1.facultyid = f.facultyid join subjectinfo s on f1.subjectid = s.subjectid where
iscurrent=?;";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);
        ArrayList<FacultySubjectTO> data = null;
        stmt.setString(1, Iscurrent);
        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
            rs.beforeFirst();
            data = new ArrayList<FacultySubjectTO>();
            while (rs.next()) {
                FacultySubjectTO ans = new FacultySubjectTO();
                ans.setSrNo(rs.getInt(1));
                ans.setFacultyID(rs.getString(2));
                ans.setSubjectID(rs.getString(3));
                ans.setIsCurrent(rs.getString(4));
                ans.setFacultyName(rs.getString(5));
                ans.setSubjectName(rs.getString(6));
                data.add(ans);
            }
        }
        stmt.close();
        return data;
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
        errorMessage = ex.getMessage();
        return null;
    }
}

```



```
    }  
}  
}
```

---

```
package model.to;  
  
import java.sql.Time;  
  
public class RoomAllocationTO {  
  
    private String FacultyID , SubjectID , RoomID , DayName;  
  
    private String FacultyName;  
  
    private String IsActive;  
  
  
    public String getIsActive() {  
  
        return IsActive;  
  
    }  
  
    public void setIsActive(String IsActive) {  
  
        this.IsActive = IsActive;  
  
    }  
  
    public String getFacultyName() {  
  
        return FacultyName;  
  
    }  
  
    public void setFacultyName(String FacultyName) {  
  
        this.FacultyName = FacultyName;  
  
    }  
  
    public String getSubjectName() {  
  
        return SubjectName;  
  
    }  
}
```

```
public void setSubjectName(String SubjectName) {

    this.SubjectName = SubjectName;

}

public String getRoomName() {

    return RoomName;

}

public void setRoomName(String RoomName) {

    this.RoomName = RoomName;

}

private String SubjectName;

private String RoomName;

public String getFacultyID() {

    return FacultyID;

}

public void setFacultyID(String FacultyID) {

    this.FacultyID = FacultyID;

}

public String getSubjectID() {

    return SubjectID;

}

public void setSubjectID(String SubjectID) {

    this.SubjectID = SubjectID;

}

public String getRoomID() {

    return RoomID;

}
```

```
    public void setRoomID(String RoomID) {  
  
        this.RoomID = RoomID;  
  
    }  
  
    public String getDayName() {  
  
        return DayName;  
  
    }  
  
    public void setDayName(String DayName) {  
  
        this.DayName = DayName;  
  
    }  
  
    public Integer getSrNO() {  
  
        return SrNO;  
  
    }  
  
    public void setSrNO(Integer SrNO) {  
  
        this.SrNO = SrNO;  
  
    }  
  
    private Integer SrNO;  
  
    private Time LectureTime;  
  
  
    public Time getLectureTime() {  
  
        return LectureTime;  
  
    }  
  
    public void setLectureTime(Time LectureTime) {  
  
        this.LectureTime = LectureTime;  
  
    }  
  
}
```

---

```
package model.dao;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import model.to.RoomAllocationTO;

public class RoomAllocationDAO {

    private String errorMessage;

    public String getErrorMessage() {

        return errorMessage;

    }

    public boolean insertRecord(RoomAllocationTO data) {

        try {

            String query = "insert into
roomallocation(facultyid,subjectid,roomid,dayname,isactive,lecturetime) values(?,?,?,?,?,?)";

            PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

            stmt.setString(1, data.getFacultyID());

            stmt.setString(2, data.getSubjectID());

            stmt.setString(3, data.getRoomID());

            stmt.setString(4, data.getDayName());

            stmt.setString(5, data.getIsActive());

            stmt.setTime(6, data.getLectureTime());

            boolean ans = stmt.executeUpdate() > 0;

            stmt.close();
```

```

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return false;

    }

}

public boolean updateRecord(RoomAllocationTO data) {

    try {

        String query = "update roomallocation set facultyid=? , subjectid=? , roomid=? , dayname=? ,
isactive=? , lecturetime=? where SrNo=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1, data.getFacultyID());

        stmt.setString(2, data.getSubjectID());

        stmt.setString(3, data.getRoomID());

        stmt.setString(4, data.getDayName());

        stmt.setString(5, data.getIsActive());

        stmt.setTime(6, data.getLectureTime());

        stmt.setInt(7, data.getSrNO());

        boolean ans = stmt.executeUpdate() > 0;

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return false;
    }
}

```

```

    }
}

public boolean deleteRecord(int srno) {

    try {

        String query = "delete from roomallocation where srno=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setInt(1, srno);

        boolean ans = stmt.executeUpdate() > 0;

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return false;

    }

}

```

```

public boolean isAvailable(String facid , String subid , String roomid) {

    try {

        String query = "select * from roomallocation where facultyid=? and subjectid=? and roomid=? and isactive='Yes'";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setString(1,facid);

        stmt.setString(2,subid);

        stmt.setString(3, roomid);

        ResultSet rs = stmt.executeQuery();

        boolean ans = false;
    }
}

```

```

        if(rs.next()){

            ans= true;

        }

        rs.close();

        stmt.close();

        return ans;

    } catch (Exception ex) {

        System.out.println(ex.getMessage());

        errorMessage = ex.getMessage();

        return false;

    }

}

public RoomAllocationTO getRecord(int srno) {

    try {

        String query = "select srno , facultyid , subjectid , roomid , dayname , isactive from
roomallocation where srno=?";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        stmt.setInt(1, srno);

        RoomAllocationTO ans = null;

        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            ans = new RoomAllocationTO();

            ans.setSrNO(rs.getInt(1));

            ans.setFacultyID(rs.getString(2));

            ans.setSubjectID(rs.getString(3));

            ans.setRoomID(rs.getString(4));

```

```

        ans.setDayName(rs.getString(5));

        ans.setIsActive(rs.getString(6));

        ans.setLectureTime(rs.getTime(7));

    }

    stmt.close();

    return ans;

} catch (Exception ex) {

    System.out.println(ex.getMessage());

    errorMessage = ex.getMessage();

    return null;

}

}

public ArrayList<RoomAllocationTO> getAllRecord() {

    try {

        String query = "select srno , r1.facultyid , r1.subjectid , r1.roomid , dayname ,isactive ,
r1.lecturetime , facultyname , subjectname , roomname\n" +

        "from roomallocation r1\n" +

        "join facultyinfo f on r1.facultyid = f.facultyid\n" +

        "join subjectinfo s on r1.subjectid = s.subjectid join roominfo r on r1.roomid = r.roomid;";

        PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);

        ArrayList<RoomAllocationTO> data = null;

        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            rs.beforeFirst();

            data = new ArrayList<RoomAllocationTO>();

            while (rs.next()) {

```



```

        RoomAllocationTO ans = new RoomAllocationTO();

        ans.setSrNO(rs.getInt(1));

        ans.setFacultyID(rs.getString(2));

        ans.setSubjectID(rs.getString(3));

        ans.setRoomID(rs.getString(4));

        ans.setDayName(rs.getString(5));

        ans.setIsActive(rs.getString(6));

        ans.setLectureTime(rs.getTime(7));

        ans.setFacultyName(rs.getString(8));

        ans.setSubjectName(rs.getString(9));

        ans.setRoomName(rs.getString(10) + " [ " + rs.getString(4) + " ]");

        data.add(ans);

    }

}

stmt.close();

return data;

} catch (Exception ex) {

    System.out.println(ex.getMessage());

    errorMessage = ex.getMessage();

    return null;

}

}

public ArrayList<RoomAllocationTO> getAllRecord(String Roomid,String IsActive) {

    try {

```

```
String query = "select srno , r1.facultyid , r1.subjectid , r1.roomid , dayname ,isactive ,  
r1.lecturetime , facultyname , subjectname , roomname\n" +  
"from roomallocation r1\n" +  
"join facultyinfo f on r1.facultyid = f.facultyid\n" +  
"join subjectinfo s on r1.subjectid = s.subjectid join roominfo r on r1.roomid = r.roomid where  
r1.roomid=? and r1.isactive=?";
```

```
PreparedStatement stmt = DataConnection.getConnection().prepareStatement(query);
```

```
ArrayList<RoomAllocationTO> data = null;
```

```
stmt.setString(1, Roomid);
```

```
stmt.setString(2, IsActive);
```

```
ResultSet rs = stmt.executeQuery();
```

```
if (rs.next()) {
```

```
    rs.beforeFirst();
```

```
    data = new ArrayList<RoomAllocationTO>();
```

```
    while (rs.next()) {
```

```
        RoomAllocationTO ans = new RoomAllocationTO();
```

```
        ans.setSrNO(rs.getInt(1));
```

```
        ans.setFacultyID(rs.getString(2));
```

```
        ans.setSubjectID(rs.getString(3));
```

```
        ans.setRoomID(rs.getString(4));
```

```
        ans.setDayName(rs.getString(5));
```

```
        ans.setIsActive(rs.getString(6));
```

```
        ans.setLectureTime(rs.getTime(7));
```

```
        ans.setFacultyName(rs.getString(8));
```

```
        ans.setSubjectName(rs.getString(9));
```

```
        ans.setRoomName(rs.getString(10) + " [ " + rs.getString(4) + " ]");
```

```
        data.add(ans);

    }

}

stmt.close();

return data;

} catch (Exception ex) {

    System.out.println(ex.getMessage());

    errorMessage = ex.getMessage();

    return null;

} }
```

## REFERENCES

### 9.1 Bibliography

- Java Programming Language by Sun Microsystems
- The Complete Reference, Fifth Edition by Herbert Schildt
- SQL for Dummies, 5<sup>th</sup> Edition by Allen G. Taylor
- A project of Address Book

### 9.2 Institute Contact Details:

TEACHER'S NAME: **Mr. Mukesh Jamwal**

PHONE NO: 9355584910, 9896633881

E-MAIL ID: [grapess@grapess.com](mailto:grapess@grapess.com)

### COMPLETE ADDRESS:

#3, Block No. 533, First Floor, Model Town,

Opp. Kumar Mall, Sarni Chowk, YAMUNA NAGAR-135003

WEBSITE: [www.grapess.com](http://www.grapess.com)