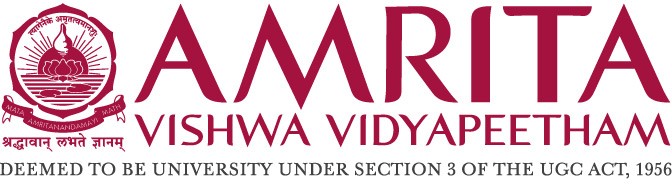
“Simplified Timetable Management and Substitution System for Amrita College using Java”

*As a part of the subject*

### OBJECT ORIENTIED PROGRAMMING



**Centre for Computational Engineering and Networking**

**AMRITA SCHOOL OF ENGINEERING**

## AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112 (INDIA)

**July- 2023**

**GROUP – 5**

**GROUP MEMBERS:**

|  |  |  |
| --- | --- | --- |
| S.NO | NAME | ROLL NO |
| 1 | M.NILAVARASAM | CB.EN.U4AIE22139 |
| 2 | M SANTHOSH | CB.EN.U4AIE22152 |
| 3 | KL AMRITHA NANDINI | CB.EN.U4AIE22126 |
| 4 | CH ARJUN | CB.EN.U4AIE22106 |

## Acknowledgement

We would like to express our special thanks of gratitude to our teacher (MR. VIPIN sir), who gave us the golden opportunity to do this wonderful project on the topic ( ), which also helped us in doing a lot of Research and we came to know about so many new things. We are thankful for the opportunity given.

We would also like to thank our group members, as without their cooperation, we would not have been able to complete the project within the prescribed time.

# 

# DECLARATION

We declare that the Submitted Report is our original work and no values and context of it has been copy pasted from anywhere else. We take full responsibility, that if in future, the report is found invalid or copied, the last decision will be of the Faculty concerned. Any form of plagiarism will lead to disqualification of the report.

# 1.1.Abstract

This project aim’s in shorting out one of the very common problem, in amrita university. Amrita College faces the common issue of managing timetables for its teachers when the teacher is absent or may not be availabe in a specific period, with each faculty member having unique schedules. To address this challenge, we propose the development of a Java-based application that automates the process of finding substitute teachers when a regular teacher is absent during a period. If a teacher is absent or will not conduct class for a specfic period , the

The application leverages the existing timetables of teachers to identify available periods, and sends notifications to those teachers indicating the free periods. Through this system, teachers have the opportunity to accept the free period and indicate their availability to conduct a class. If no teacher have approved the period then that it will be declared as a free-hour.

Once a teacher accepts the free period, the application sends a notification to all students in the respective class, informing them of the substitute teacher assigned to conduct the class. This ensures seamless communication and minimizes disruptions for students in case of teacher absences.

The current system's **weaknesses** include time-consuming manual processes, lack of real-time updates, difficulty finding substitutes, and inefficient communication. The proposed application aims to **overcome** these issues by providing a streamlined process. It sends notifications to available teachers and informs students of substitute teachers, minimizing disruptions.

## 1.1.1. Modules.

There are Total Of Four Module Present In this Project, and Each member are Are not Contributed based on the Module, Since each module is related to ther module. So we have not Done this Project based on The module, but on the flow of the Project.

### 1.1.1.1. Module-1: Class Schedule Generator.

* Automates finding substitute teachers by leveraging existing timetables
* Identifies available periods and notifies teachers about free periods

### 1.1.1.2. Module-2: Period Reservation.

* Manages reservations for tests, general meetings, public events, and holidays
* Ensures proper scheduling and avoids conflicts with regular classes

### 1.1.1.3. Module-3:Free Period Notification

* Allows teachers to indicate availability for substitute teaching during free periods
* Teachers receive notifications and can reserve periods for their classes

### 1.1.1.4. Module-4: Updated Class Timing.

* Manages changes and updates to the class schedule
* Allows modifications to class timings and room allocations

# 1.2.Introduction.

“Your team is required to build a JAVA application which would solve an issue in Amrita”, this project was to solve, any issue we are currently facing in the Amrita campus, using java.

And So we have decided to solve a very common issue that we are currenlty facing in Amrita, the most common issue among student is suppose, if a teacher is absent they may not be informed that the teacher is absent. And even the other teacher’s may not know that there is a free-period. So there is loss of a period for both students and teacher’s.

To solve this issue we are implementing, a system where the teacher who is going to be absent for a period or, a whole day should first get approved from the admin. And if he grantes permisson, then we will see through each teacher’s timetable’s and if any teacher has free-period, then the notification will be sent to that specific teacher. If any teacher accept’s the period, (then that period will be reserved for that teacher) then notification will be sent to that specific student’s of the class, And the student’s can even complain to the Admin, in case if they are not informed about the subsititue teacher’s, or an notification stating free Perod

This above is just a important-part in this project, and there is a lot more stuff’s in this project, which will be covered in the upcoming section of the report.

So this project was implemented using java, and it solve’s the issue that we are currently facing in Amrita. But this project has many pros and some cons also, below section we have illustrated the resources used in this java project.

## 1.2.1. Resources Used For This Project.

* IDE: intelijee.
* Package Type/ Project Type: Java Fx.
* Data Storage: txt-File.

## 1.2.2. Pro’s And Con’s Of This Project.

As told in the above section there are many Pro’s and some con’s in our project, the pros’s and the con’s I our project are listed below.

### 1.2.2.1.Project Pro’s.

1. Our project will solve the issue in amrita.
2. Everything is automated, so even the admin does’nt require to assign period in case if a teacher is not present, even currently this is a big issue in Amrita. Where suppose a period is free, the admin has to go-through all teacher’s timetable, and if he/she has free period. Then he have text/call and ask the teacher’s. So this can be overcomed using our project.
3. Efficiently solves timetable management challenges at Amrita College, minimizing disruptions and improving productivity.
4. Automation eliminates the need for manual period assignments, saving time and effort for administrators.
5. Real-time updates and notifications ensure timely communication to teachers and students, enhancing efficiency and reducing confusion.
6. And there are many pro’s in this project also.

### 1.2.2.2 Project Con’s.

1. The project is one-side, everything happen in one-side. Suppose if a staff send’s request for pass, to the admin.
2. This project cannot be implemented in college directly, as there is no served connected in this project.
3. No privacy, for the teacher’s. As the admin has access over everyone’s account.
4. It may become Compilcated, when the data is Stored is more. (since we are using txt-file).

# 1.3. Libraries Imported In This Project.

Java libraries are pre-compiled code collections that provide additional functionality to programs. Importing libraries allows for code reuse and simplifies complex tasks. There were several libraries used in this Project and the below section illustartes the list of all the libraries that were imported in our Java Project and the usage of this libraries in our java project.

* import javafx.animation.PauseTransition;
* import javafx.application.Application;
* import javafx.geometry.Insets;
* import javafx.geometry.Pos;
* import javafx.scene.Node;
* import javafx.scene.Scene;
* import javafx.scene.control.\*;
* import javafx.scene.image.Image;
* import javafx.scene.image.ImageView;
* import javafx.scene.layout.\*;
* import javafx.scene.paint.Color;
* import javafx.scene.text.Font;
* import javafx.scene.text.FontWeight;
* import javafx.scene.text.Text;
* import javafx.stage.Stage;
* import javafx.util.Duration;
* import java.io.BufferedWriter;
* import java.io.File;
* import java.io.FileWriter;
* import java.io.IOException;
* import java.time.LocalDate;
* import java.util.ArrayList;
* import java.util.Arrays;
* import java.util.*List*;
* import java.util.Scanner;

This were the libraries That were imported in Our Java Project, the below section will Explain the Role of this Libraries in our Project.

## 1.3.1import javafx.animation.PauseTransition;

This library provides classes and methods for creating animations and transitions in JavaFX. The PauseTransition class allows you to pause the execution of your program for a specified duration.

## 1.3.2.import javafx.application.Application;

This library is used for creating JavaFX applications. It provides the necessary classes and methods to initialize and launch a JavaFX application, such as defining the application's entry point and managing the application lifecycle.

## 1.3.3.import javafx.geometry.Insets;

This library provides classes to define and manipulate insets, which represent the padding or margins around a graphical element in JavaFX.

## 1.3.4.import javafx.geometry.Pos;

This library provides classes to define and work with positions and alignments in JavaFX. The Pos class represents a position or alignment on the screen, such as top-left, center, or bottom-right. It is used for aligning nodes within layout containers.

## 1.3.5.import javafx.scene.Node;

This library provides classes and methods for working with graphical elements in JavaFX. The Node class is the base class for all graphical elements in JavaFX, such as buttons, labels, and layouts. It provides properties and methods for managing the visual representation and behavior of a node

## 1.3.6.import javafx.scene.Scene;

This library provides classes and methods for creating and managing scenes in JavaFX. A scene represents a container for the graphical elements of a JavaFX application. It provides methods for setting the root node of the scene and managing the scene's dimensions and properties.

## 1.3.7.import javafx.scene.control.\*;

This library provides various UI controls such as buttons, labels, text fields, and more, allowing you to interact with users and display information in your application.

## 1.3.8.import javafx.scene.image.Image;

It enables you to load and work with images in different formats. You can use it to display images within your application.

## 1.3.9.import javafx.scene.image.ImageView;

This library provides a way to display images within the UI. It allows you to resize, rotate, and apply other transformations to the images.

1.3.10.import javafx.scene.layout.\*;

These classes help you arrange and organize UI elements in a structured manner. They provide different layout containers like VBox, HBox, GridPane, etc., allowing you to control the positioning and alignment of UI components.

## 1.3.11.import javafx.scene.paint.Color;

It represents colors used in the UI. You can specify colors using predefined constants or custom RGB values. It allows you to set the color of UI elements, text, shapes, etc.

## 1.3.12.import javafx.scene.text.Font;

This library provides font-related functionalities. You can use it to set the font family, size, and style for text elements in your application.

## 1.3.13.import javafx.scene.text.FontWeight; It allows you to specify the weight or boldness of the text. You can make text appear bold or use different font weights for emphasis.

## 1.3.14.import javafx.scene.text.Text;

This class enables you to display text in the UI. You can customize the font, size, color, and alignment of the text.

## 1.3.15.

import javafx.stage.Stage;

It represents the main application window. You can manage its properties, such as size, title, and visibility. It also provides methods to control the lifecycle of the application.

## 1.3.16.import javafx.util.Duration;

This library represents a duration or time interval. It is commonly used for animations and transitions, allowing you to specify the duration of visual effects and control the timing of UI changes.

## 1.3.17.import java.io.BufferedWriter;

This class is used for writing text to a character-output stream with buffering capabilities. It provides methods to write data efficiently, improving performance by reducing the number of write operations to the underlying stream.

## 1.3.18. import java.io.File;

It represents a file or directory path in the file system. It provides methods to manipulate files, such as creating new files, deleting files, checking file properties, and more.

## 1.3.19. import java.io.FileWriter;

This class extends `Writer` and is used for writing characters to a file. It allows you to write text data to a file using a character stream.

## 1.3.20. import java.io.IOException;

It is an exception class that indicates an input/output error. It is thrown when there are issues with reading from or writing to a file, or when file operations fail.

## 1.3.21. import java.time.LocalDate;

It represents a date without a specific time zone. It provides methods to work with dates, perform calculations, and format dates.

## 1.3.22. import java.time.LocalTime;

It represents a time without a specific time zone. It provides methods to work with times, perform calculations, and format times.1.3.23. import java.time.format.DateTimeFormatter;

This class provides methods to format and parse dates and times. It allows you to define custom patterns for date and time representations.  
1.3.24. import java.util.ArrayList;

It is a class that implements the `List` interface and provides an ordered collection of elements. It allows dynamic resizing and provides methods to add, remove, and access elements in the list.

## 1.3.25. import java.util.Arrays;

It is a utility class that provides various methods for working with arrays. It includes methods to sort arrays, fill arrays with values, search for elements, and more.

## 1.3.26. import java.util.*List*;

It is an interface that defines the behavior of an ordered collection. It provides methods to add, remove, and access elements in the collection.

## 1.3.27 import java.util.Scanner;

This class is used for reading input from different sources, such as files or user input. It provides methods to parse and extract data from various types of input.

# 1.4. Class’s Create For This Project.

There were several class’s that were created by us for our Project, each individulal class’s represents a GUI-page, and each class contains both Front-End and Back-End. All The Class’s are connected to each other, by some means. Below is the List of all the class’s that we have Created for this project.

* public class Step1 extends Application
* public class Step2 extends Application
* public class Step3 extends Application
* public class Step4 extends Application
* public class Step5 extends Application
* public class Step6 extends Application
* public class Step7 extends Application
* public class Step1About extends Application
* public class Step1Contact extends Application
* public class TeacherAdmin extends Application
* public class TeacherNotification extends Application
* public class SignInInfoPage extends Application
* public class LeaveReq extends Application
* public class AdminAllTeachersInfo extends Application
* public class AdminHomePage extends Application
* public class AdminNotification extends Application
* public class AdminRecruitTeachers extends Application
* public class Storage

1. class StorageAdmin

These were the Class’s that were create by us for Our Java Project, in the upcoming section we will explain briefly each class, their role, and the output of each class’s. We have split the class’s into three part,

1. admin side class’s.
2. Teacher side class’.
3. Storage Class.

These were the Three Segments we have seperated our, each section has the output image, UML image of the class and the method’s present in the class.

## 1.4.1 Storage Class.

### 1.4.1.1 class StorageAdmin

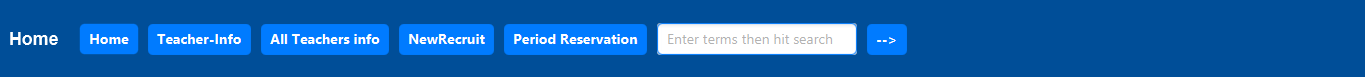
There Are Two Function in this class, and these class’s are called Storage since this contain’s a common function. And this function is called in all class’s below, so instead of writing these function’s in the main-code each time, we have created this storage class. And since these function is static, and default , these function’s can be called anywhere of the same package. And The role of the class’s is Only GUI part, and contains the header and footer of GUI, since it will be common to all the class. And this class is for the **section 1.5.2** GUI part only. The role of this class’s they have the function which serves as the header and footer in the GUI page.

1.)public static StackPane createFooterAdmin()

2.)public static HBox createHeaderAdmin(String heading)

These Are the function’s present in this class.

#### 1.4.1.1.1. Output Of Header Function, When Called In Other Class’s (Admin).



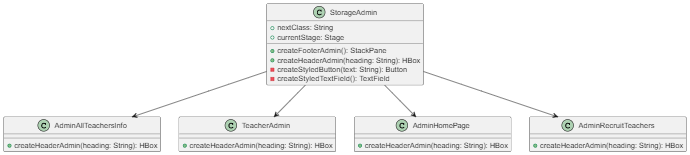
**Figure 1** Output of Header When called in other Admin-Side Class’s.

#### 1.4.1.1.2. Output Of Footer Function, When Called In Other Class’s.



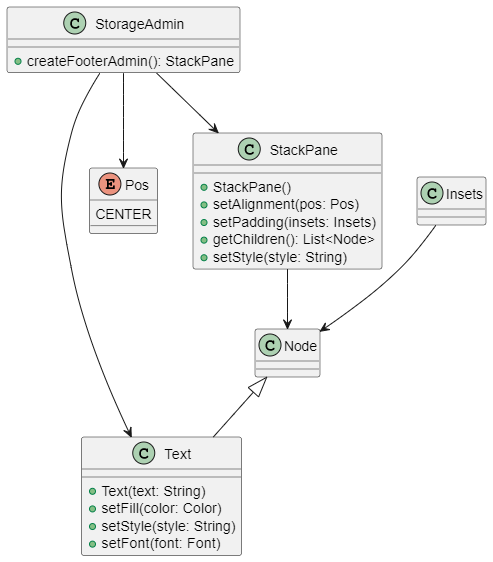
**Figure 2** Output of Header When called in other Admin-Side Class’s.

#### 1.4.1.1.3.UML Diagrams of The StorageAdmin class



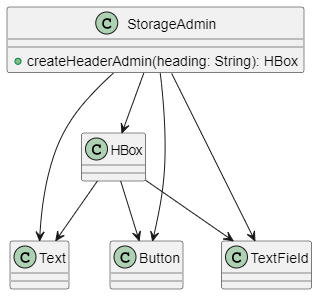
**Figure 3** Output of Header When called in other Admin-Side Class’s

#### 1.4.1.1.4. UML Of the Footer Function Of Admin Storage class.

.

#### *Figure 4 Footer Function Of Admin Storage class.*

#### 1.4.1.1.5. UML Of the Header Function Of Admin Storage class.

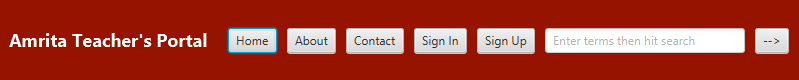


**Figure 5** UML Of the Header Function Of Admin Storage class.

### 1.4.1.2 public class Storage

Similar to above, these these function is static, and default , these function’s can be called anywhere of the same package. And The role of the class’s is Only GUI part, and contains the header and footer of GUI, since it will be common to all the class. And this class is for the **section 1.5.3** GUI part only. This also contain’s only header and footer, but this header and footer is for the teach-side class’s, these are red in color, and the above header and footer is blue in color.

#### 1.4.1.2.1. Output Of Header Function, When Called In Other Class’s (Teach)



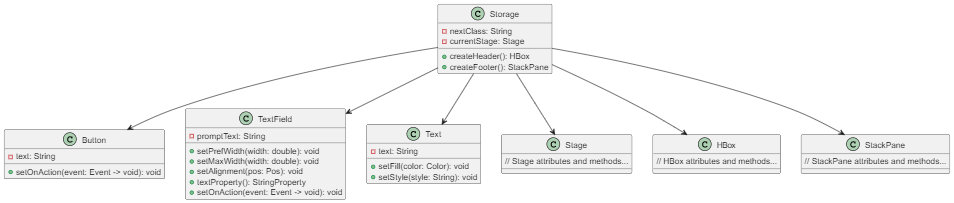
**Figure 6** Output Of Header Function, When Called In Other Class’s (Teach).

#### 1.4.1.2.2. Output Of Footer Function, When Called In Other Class’s (Teach)



**Figure 7.** Output Of Footer Function, When Called In Other Class’s (Teach).

#### 1.4.1.2.3. UML of the Storage class.



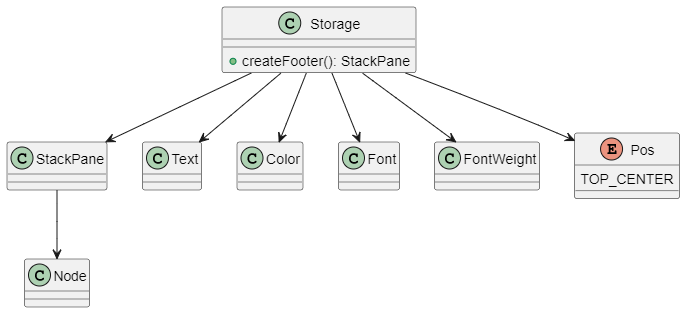
**Figure 8** UML of the Storage class**.**

#### 1.4.1.2.4. UML Of the Header Function Of Storage class.



#### *Figure 9 UML Of the Header Function Of Storage class.*

#### 1.4.1.2.5. UML Of the Footer Function Of Storage class.



**Figure 10** UML Of the Footer Function Of Storage class

## 1.4.2 Admin-Side Class’s.

These were the class’s that were created for the sake of the admin side, so basically in our project the admin corressponds to similar to administarative’s who recruit teachers, give permission for the teacher’s to take leave, in short a person who is head for all the teacher’s. This admin has several class’s. The admin has several Functionallity in our Java-Project. The Functionalties the admin that we correspond in our project are listed below:

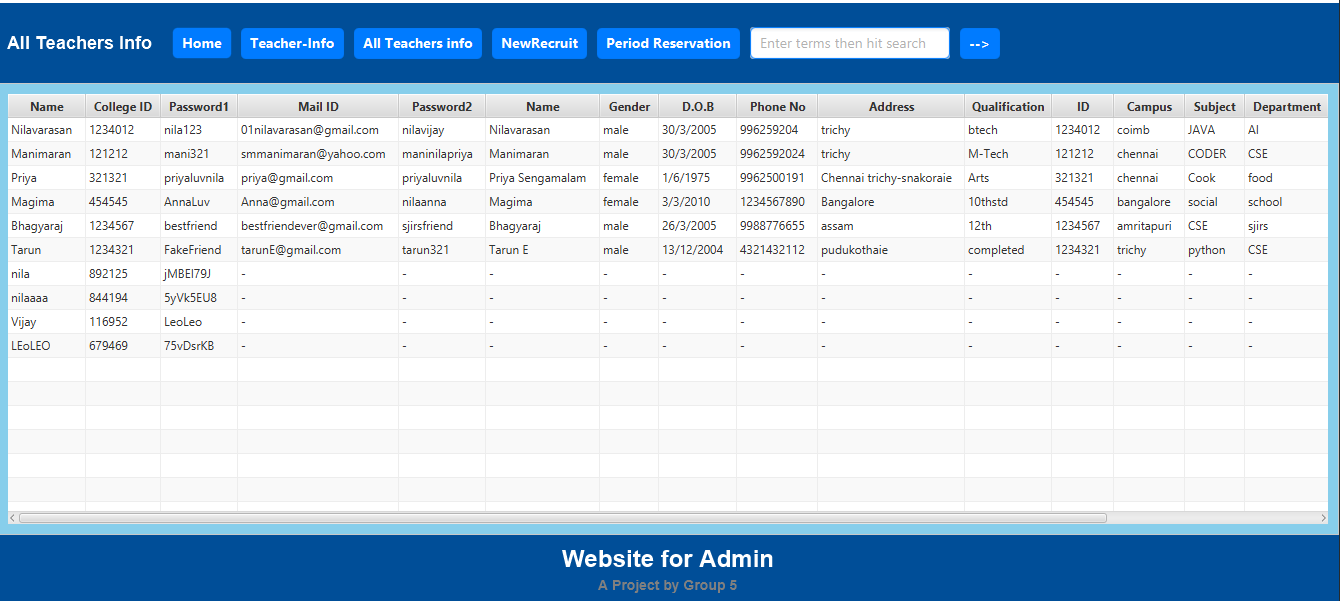
1. Can view all Teacher’s password, time-table, personal-info (when they sign-up).
2. Can message indivuial teacher’s.
3. Send email to a specific teacher.
4. Pass will be approved by the admin, he can “cancel” or “accept”.
5. He can recruit new Teacher’s, he will generate new password, college-ID. Any teacher can sign-up with the help of this College-ID and password given by the admin only.
6. Can reserve a period for any special occasion, conducting exam etc.
7. He can receive notification from all teacher’s.
8. Student’s can complain about a staff to the Admin, and so on…
9. Staff’s can leave only when the admin accept’s, and he can also sent notification to the teacher’s.

So basically, these admin-side class’s correspond’s to the admin and other type of staff’s etc.

### 1.4.2.1public class AdminAllTeachersInfo

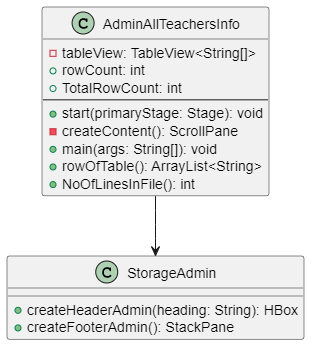
This is the class which was created for displaying all the teacher’s info personal-info(filled in the teacher-section), password, email-id. On a Single page. This is additional info for the admin, where he can dynamically see the teacher’s info. Even the new Teacher’s he recruit, that teacher’s info will also get updated dynamically here. In short this class is for the ease of the Admin, to see all the teacher’s detail in a single page, inbuilt-table which is available in javafx is used here to display this table.

#### 1.4.2.1.1. Output Image Of This Class



**Figure 11** Output Image Of This Class.

#### 1.4.2.1.2. Uml Image Of This Class

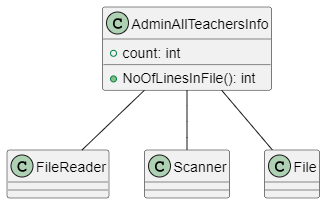


**Figure 12** Uml Image Of This Class

There are several Method’s That are created for the working of this class, below section will list the method’s and their UML-diagram.

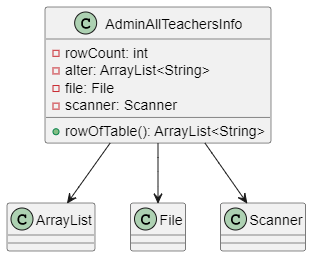
* public static int NoOfLinesInFile()
* public static ArrayList<String> rowOfTable()
* private ScrollPane createContent()
* public void start(Stage primaryStage)

#### 1.4.2.1.3. Uml Image int NoOfLinesInFile()



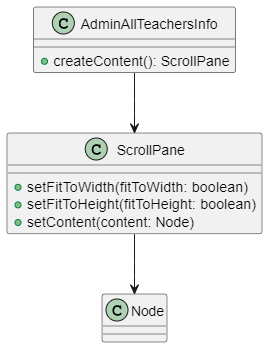
**Figure 13** Uml Image int NoOfLinesInFile().

#### 1.4.2.1.3. Uml Image ArrayList<String> rowOfTable()



**Figure 14** Uml Image ArrayList<String> rowOfTable().

#### 1.4.2.1.3. Uml Image ScrollPane createContent()



#### *Figure 15 Uml Image ScrollPane createContent()*

#### 1.4.2.1.3. Uml Image start(Stage primaryStage)

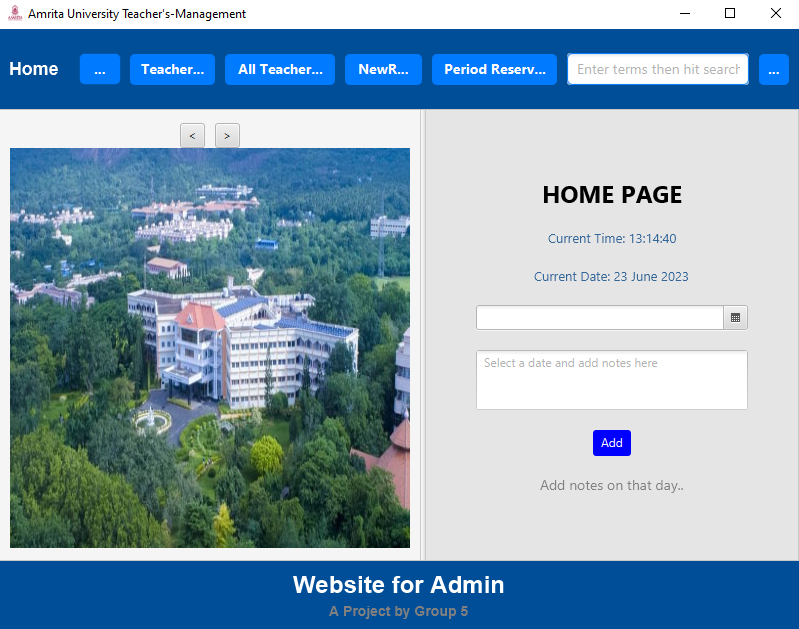


**Figure 16** Uml Image start(Stage primaryStage)

### 1.4.2.2.public class AdminHomePage

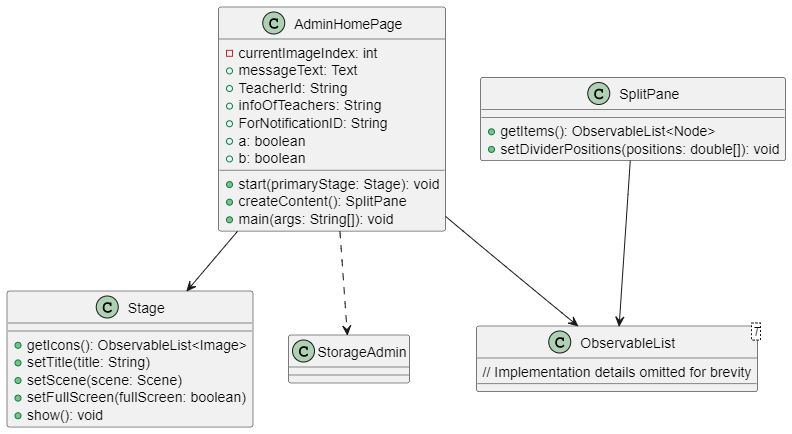
This class is the Front-page of the admin-side, this just a home-page just like any website. It has real-dynnamic-IST time, and current date. And the admin can add notes, in case if he want to remember anything, on a specific date. This is just to make the home-page look Attractive. And just a normal home-page. Also, has button’s on the left-pane where the user can see different photo’s.

#### 1.4.2.2.1. Output Image Of This Class



**Figure 17** Output Image Of This Class.

#### 1.4.2.2.2. Uml Image Of This Class

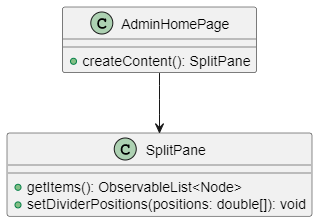


**Figure 18** Uml Image Of This Class.

There are several Method’s That are created for the working of this class, below section will list the method’s and their UML-diagram.

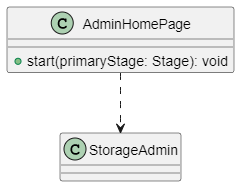
* public void start(Stage primaryStage)
* private SplitPane createContent()

#### 1.4.2.2.3. Uml Image SplitPane createContent()



**Figure 19** Uml Image SplitPane createContent()

#### 1.4.2.2.4. Uml Image void start(Stage primaryStage)

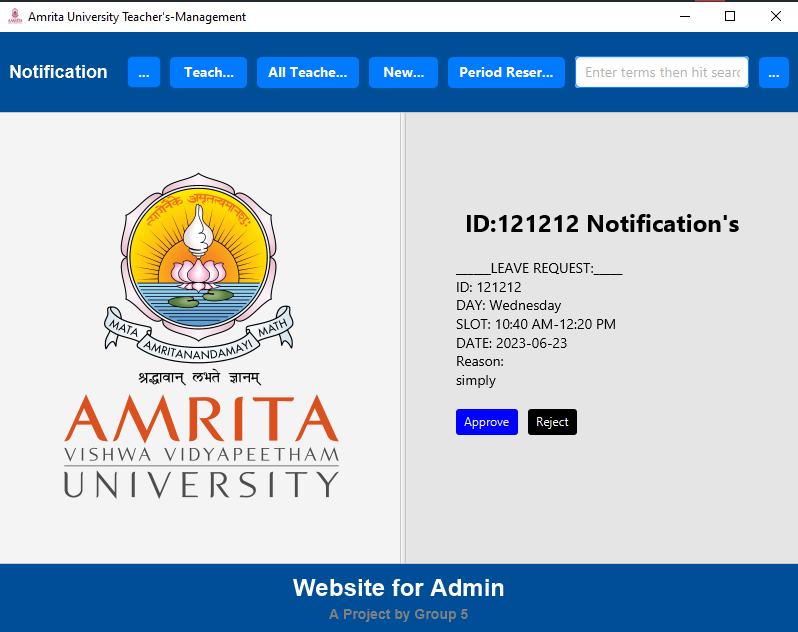


**Figure 20** Uml Image void start(Stage primaryStage)

### 1.4.2.3public class AdminNotification

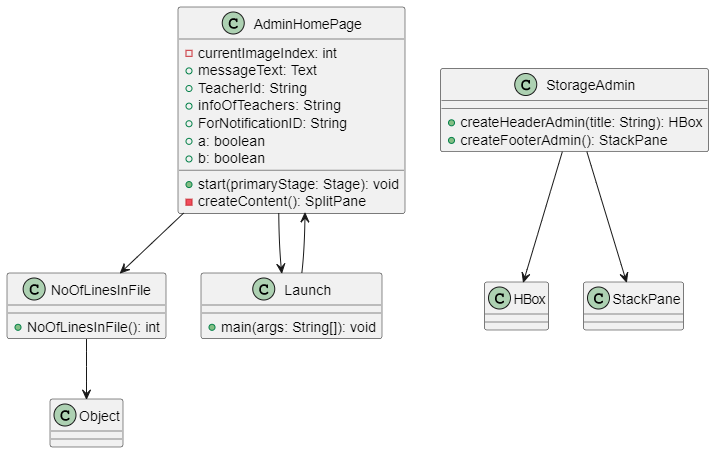
This page is where the admin and the teacher interact which each other, the admin recievies, any message or the request of pass to this page, if there is no message or anything then, there will be a message indicating “no notification”. Basically this is a page for receving notification from a specifcic teacher.

#### 1.4.2.3.1. Output Image Of This Class.



**Figure 21** Output Image Of This Class

#### 1.4.2.3.2. UML Image Of This Class

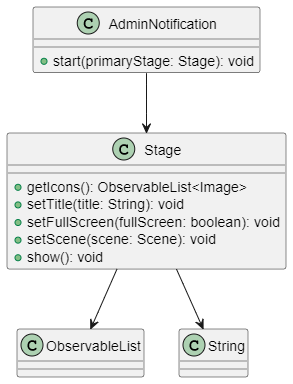


**Figure 22** UML Image Of This Class.

There are several Method’s That are created for the working of this class, below section will list the method’s and their UML-diagram.

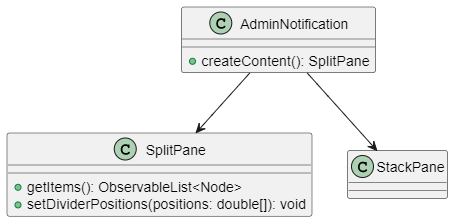
* public void start(Stage primaryStage)
* private SplitPane createContent()
* public static void DeclaringLeaveVariables()
* public static void LeaveApprovalRejection()
* public static void deleteLeaveMessage()
* public static void FileForTeachers(String temp)
* public static void print()
* public static void appendFileToFile()

#### 1.4.2.3.3. UML Image start(Stage primaryStage)



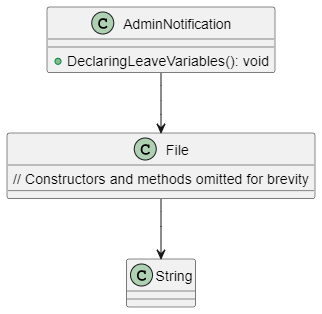
**Figure 23** UML Image start(Stage primaryStage)

#### 1.4.2.3.4. UML Image createContent()



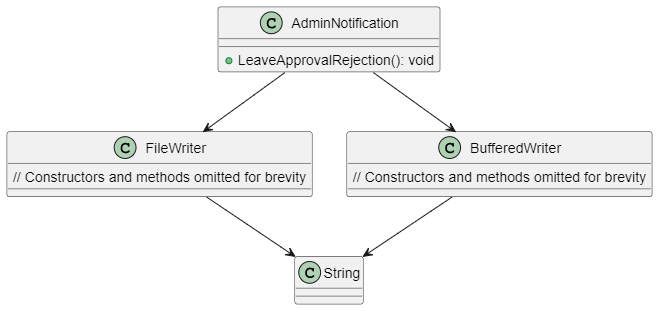
**Figure 24** UML Image createContent()

#### 1.4.2.3.5. UML Image DeclaringLeaveVariables()



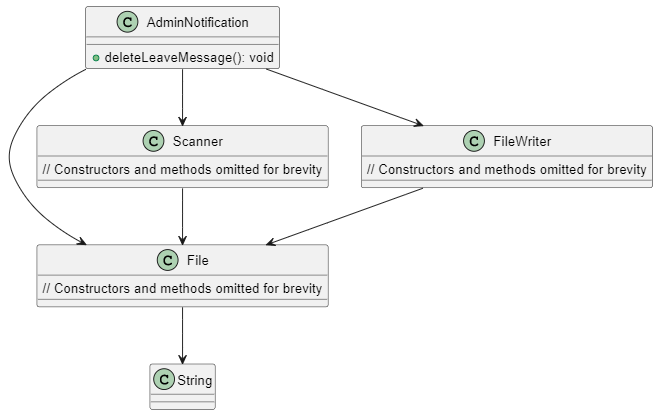
**Figure 25** UML Image DeclaringLeaveVariables()

#### 1.4.2.3.6. UML Image LeaveApprovalRejection()



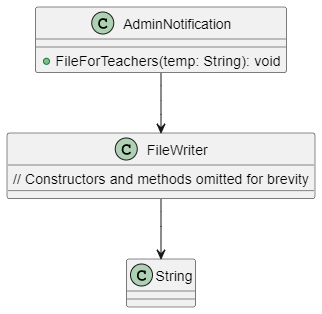
#### *Figure 26 UML Image LeaveApprovalRejection()*

#### 1.4.2.3.7. UML Image deleteLeaveMessage()



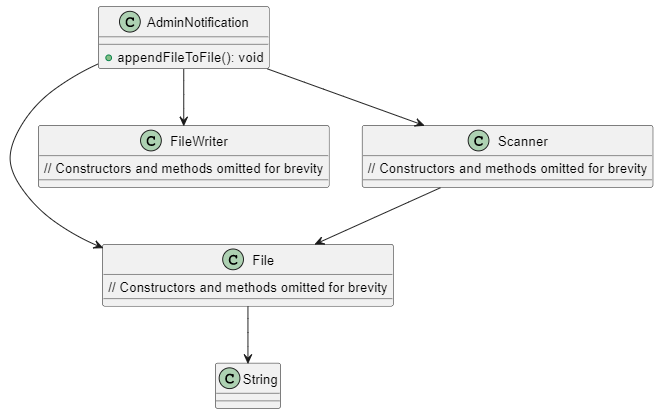
#### *Figure 27 UML Image deleteLeaveMessage()*

#### 1.4.2.3.8. UML Image FileForTeachers(String temp)



**Figure 28** UML Image FileForTeachers(String temp)

#### 1.4.2.3.9. UML Image of appendFileToFile()

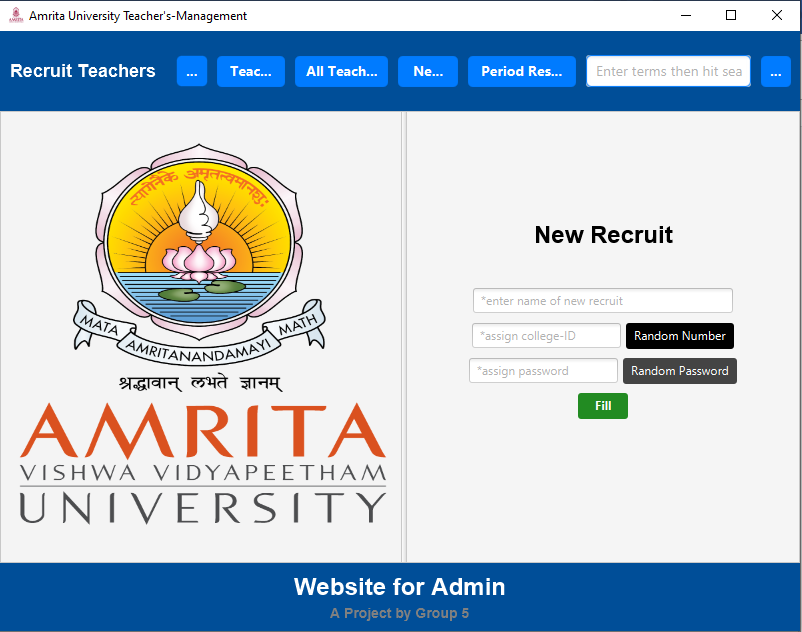


**Figure 29** UML Image of appendFileToFile()

### 1.4.2.4public class AdminRecruitTeachers

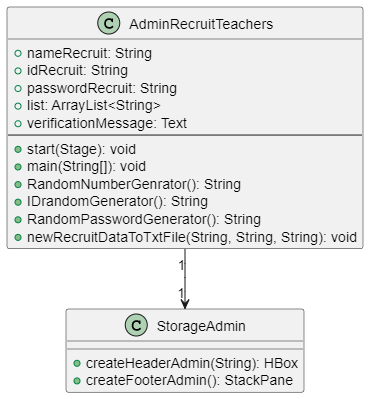
This page is where the admin will recruit any new teacher, he will give the teacher a specific password, and a unique college-ID. And any new teacher can sign-up with the help of this college-ID and password only. This password can eithier typed manullay, or a random Generator is there, if the admin feel’s lazy, can directly us this. So this where a teacher is provided detail’s to sign-up.

#### 1.4.2.4.1. Output Image Of This Class



#### *Figure 30 Output Image Of This Class*

#### 1.4.2.4.2. UML Image Of This Class

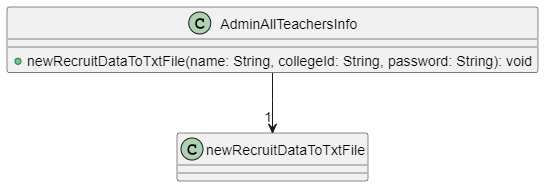


**Figure 31** UML Image Of This Class

There are several Method’s That are created for the working of this class, below section will list the method’s and their UML-diagram.

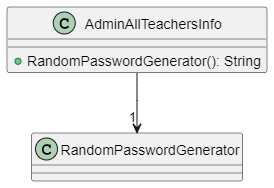
* public static void newRecruitDataToTxtFile(String name,String collegeId,String password)
* public static String RandomPasswordGenerator()
* public static String IDrandomGenerator()
* public static String RandomNumberGenrator()
* public void start(Stage primaryStage)

#### 1.4.2.4.3. UML Image newRecruitDataToTxtFile(String name,String collegeId,String password)



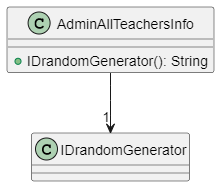
#### *Figure 32 UML Image newRecruitDataToTxtFile(String name,String collegeId,String password)*

#### 1.4.2.4.4. UML Image String RandomPasswordGenerator()



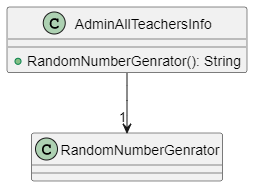
**Figure 33** UML Image String RandomPasswordGenerator()

#### 1.4.2.4.5. UML Image String IDrandomGenerator()



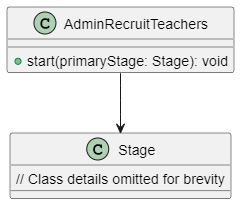
#### *Figure 34 UML Image String IDrandomGenerator()*

#### 1.4.2.4.6. UML Image String RandomNumberGenrator()



#### *Figure 35 UML Image String RandomNumberGenrator()*

#### 1.4.2.4.7. UML Image void start(Stage primaryStage)



#### *Figure 36 UML Image void start(Stage primaryStage)*

### 1.4.2.5.public class TeacherAdmin

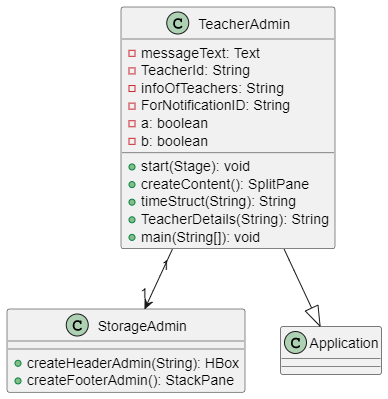
This class is where the admin, enter the collegeID of the respective teacher, to fetch his/her info,time-table,email them. And also he recvies notification (message,pass request) from the teacher in this portal.

#### 1.4.2.5.1. Output Image Of This Class

****

#### *Figure 37 Output Image Of This Class*

#### 1.4.2.5.2. UML Image Of This Class

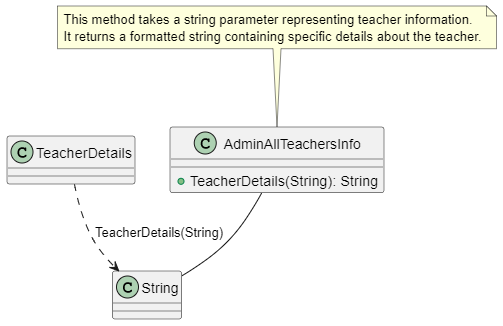


#### *Figure 38 UML Image Of This Class*

There are several Method’s That are created for the working of this class, below section will list the method’s and their UML-diagram.

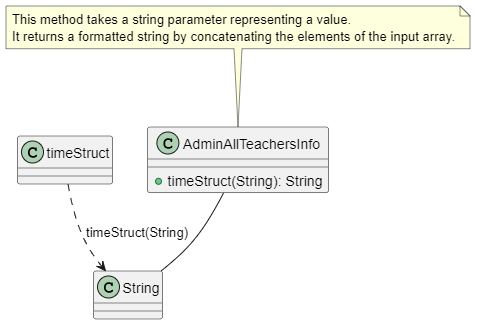
* public static String TeacherDetails(String info)
* public static String timeStruct(String value)
* private SplitPane createContent()
* public void start(Stage primaryStage)

#### 1.4.2.5.3. UML Image String TeacherDetails(String info)



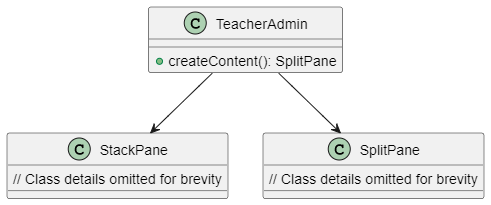
#### *Figure 39 UML Image String TeacherDetails(String info)*

#### 1.4.2.5.4. UML Image String timeStruct(String value)



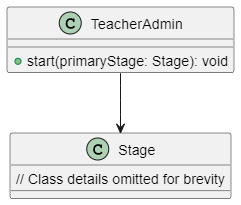
#### *Figure 40 UML Image String timeStruct(String value)*

#### 1.4.2.5.5. UML Image SplitPane createContent()



#### *Figure 41 UML Image SplitPane createContent()*

#### 1.4.2.5.6. UML Image start(Stage primaryStage)



#### *Figure 42 UML Image start(Stage primaryStage)*

## 1.4.3.Teacher-Side Class’s

There are two type’s of class’s side as discussed above, the main difference is nothing but. This admin-side class’s are used by the admin. And this teacher-side class’s are used by all teacher’s. Just that is for admin and this is for teacher’s. again in the below section the output image of the class, uml diagram of the class, uml of the method’s inside the class. Are shown, this Teacher-Side class refers to Sign-up, Sign-in, Updating time-table, applying-Pass to the Admin, Notification etc….

### 1.4.3.1 public class Step1

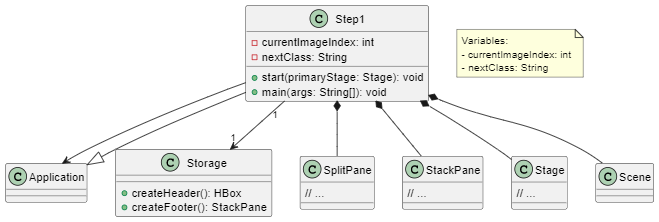
This is the home-page of the teacher-side class’s, where The admin-side and the teacher side class’s are differentiated using Color. We have differentiated the two side of class’s. Red is for the Teacher side class’s, and blue is for Admin Side class’s. This just a hoe page. Displaying some mqoute, and a image.

#### 1.4.3.1.1 Output Image of this Class.



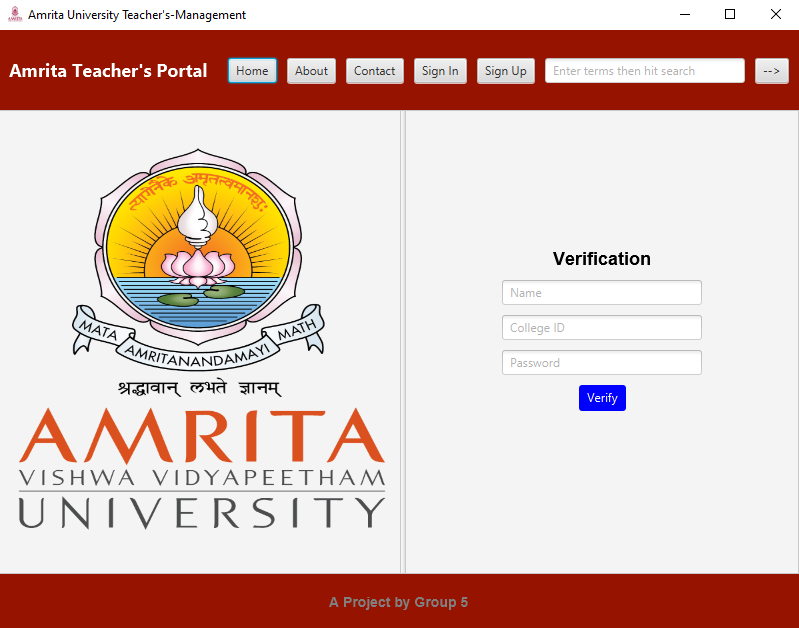
**Figure 43** Output Image of this Class.

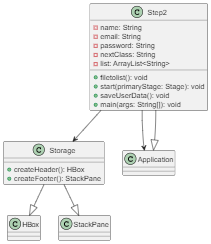
#### 1.4.3.1.2 UML Image of this Class.



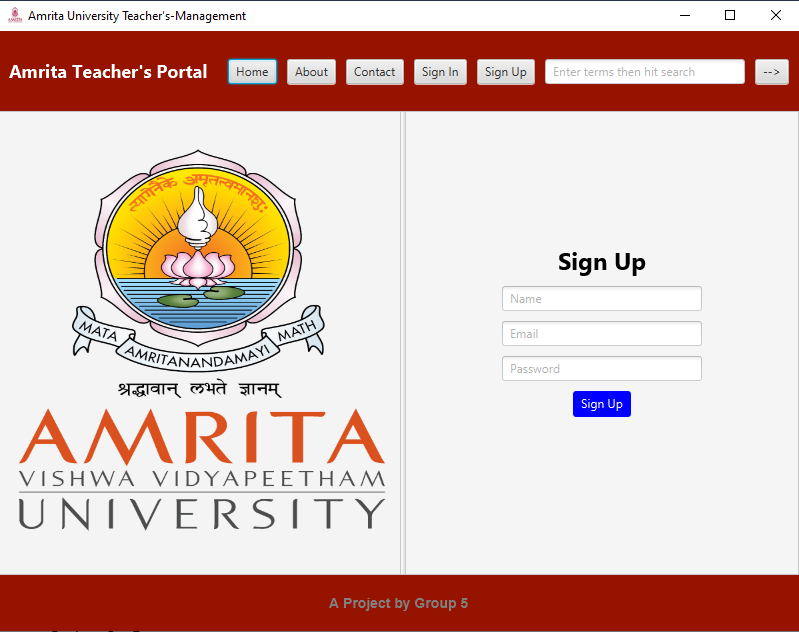
#### *Figure 44 UML Image of this Class.*

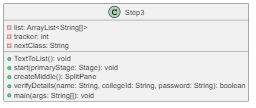
### 1.4.3.2 public class Step2



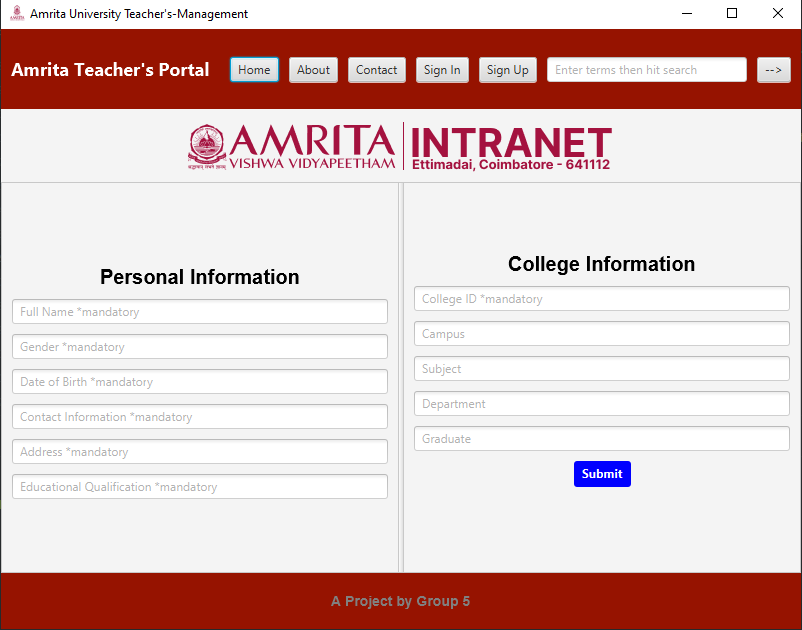


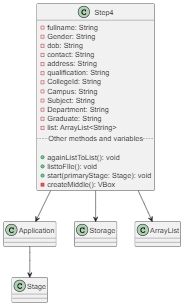
### 1.4.3.3 public class Step3



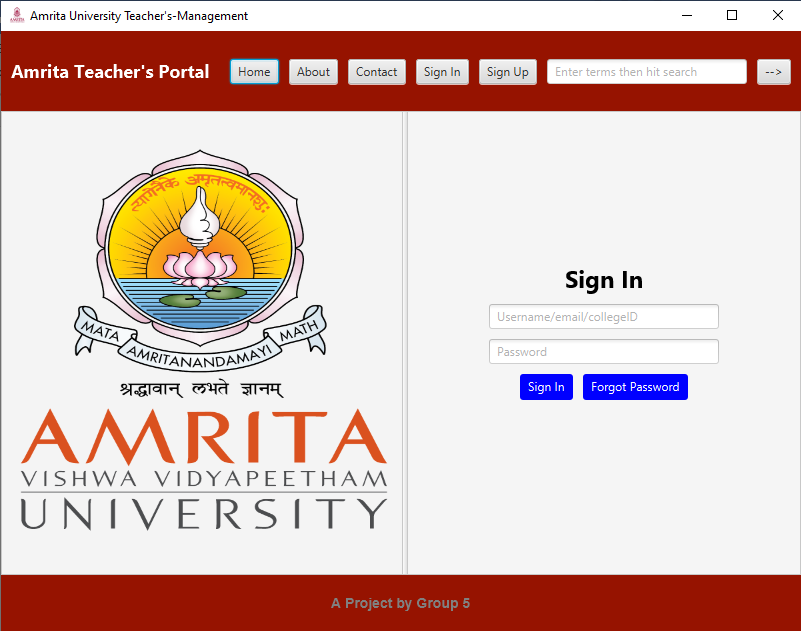


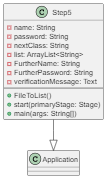
### 1.4.3.4 public class Step4



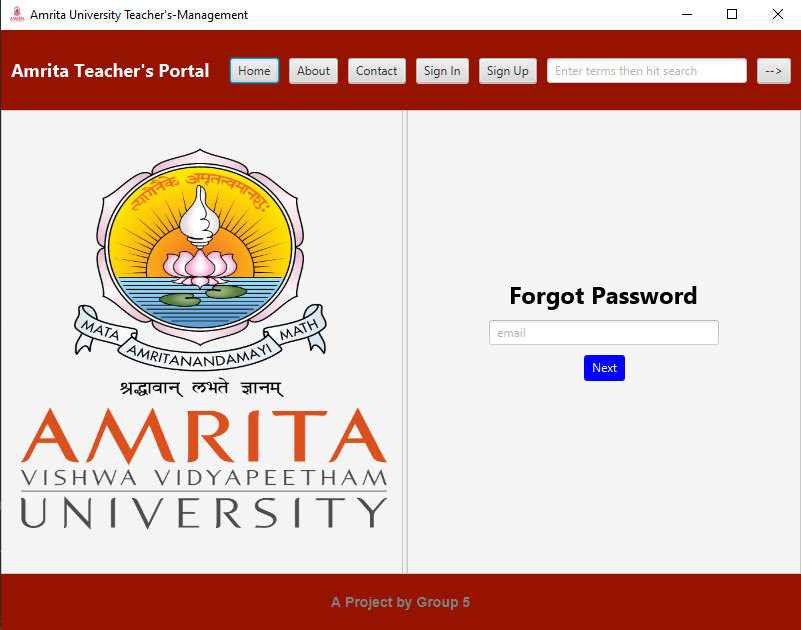


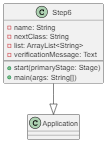
### 1.4.3.5 public class Step5



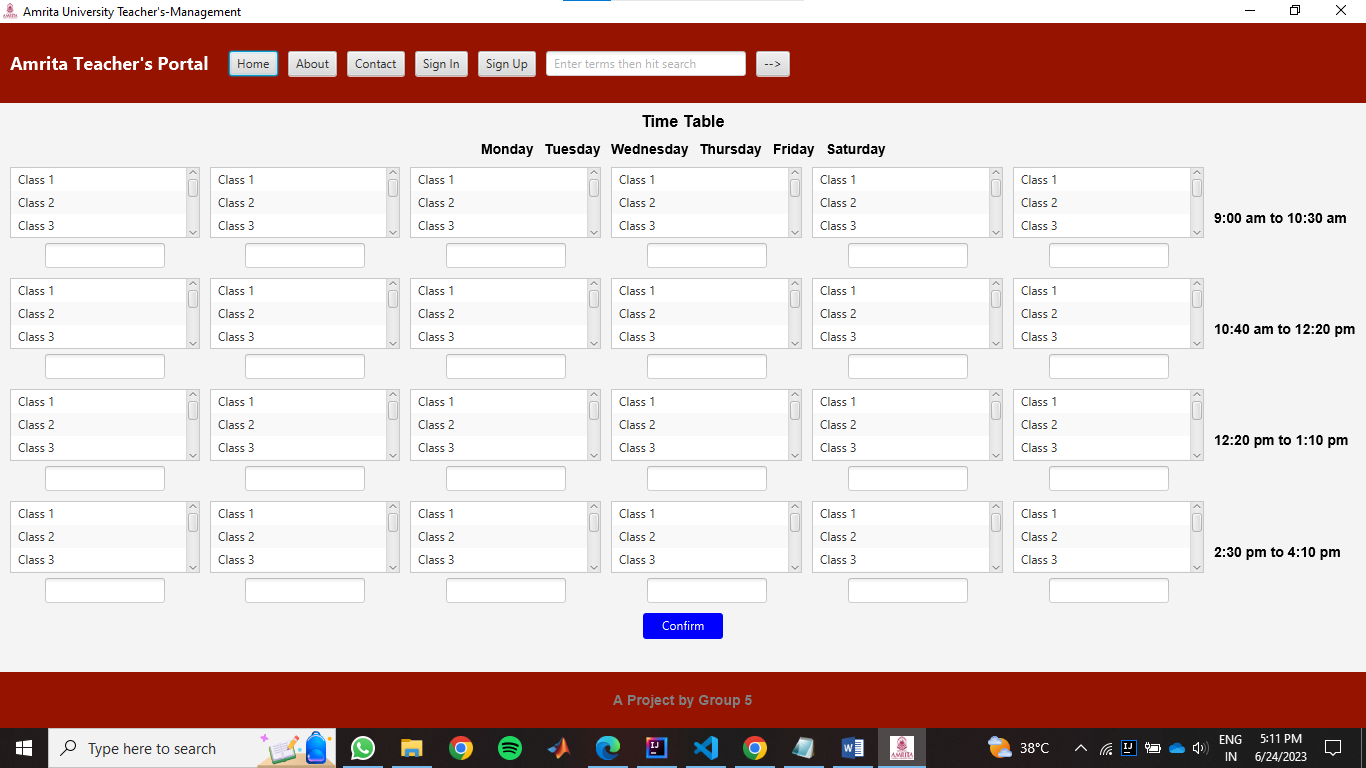


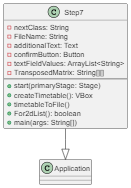
### 1.4.3.6 public class Step6



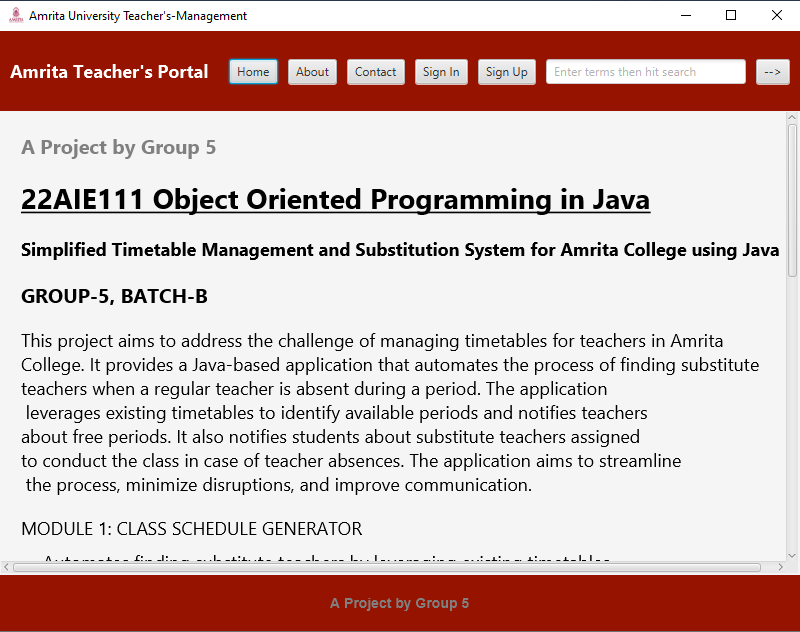


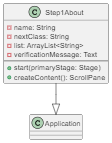
### 1.4.3.7 public class Step7



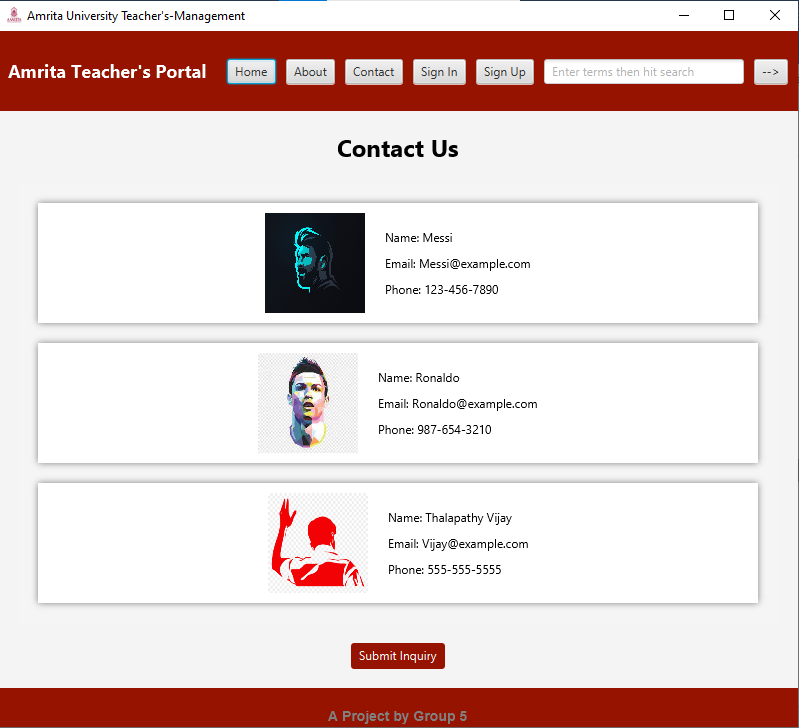


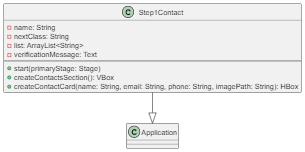
### 1.4.3.8 public class Step1About



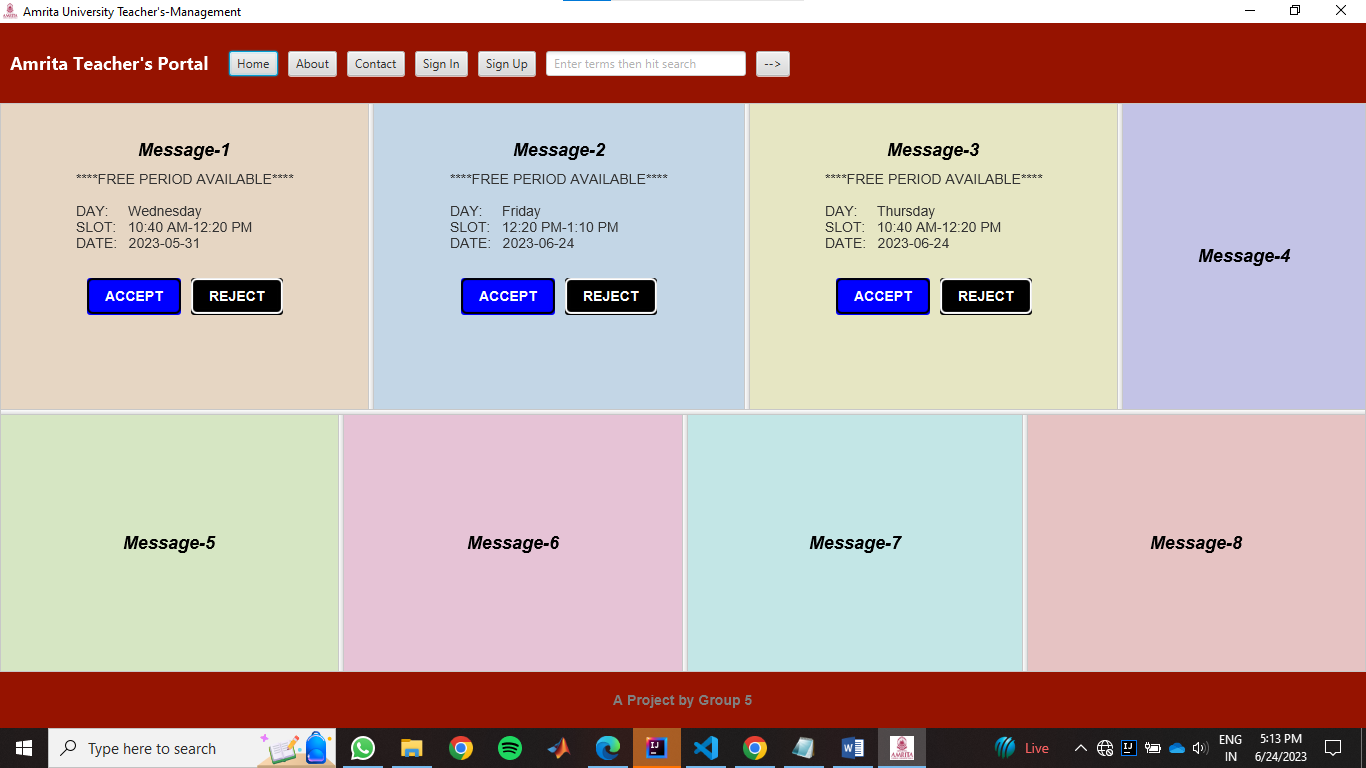


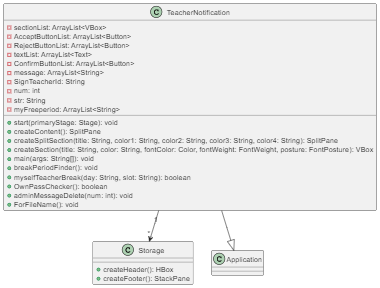
### 1.4.3.9 public class Step1Contact





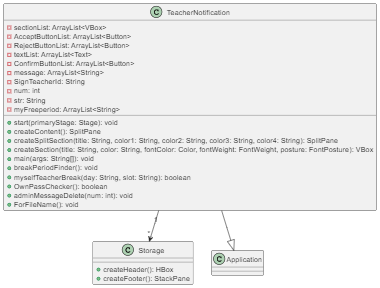
### 1.4.3.10 public class TeacherNotification



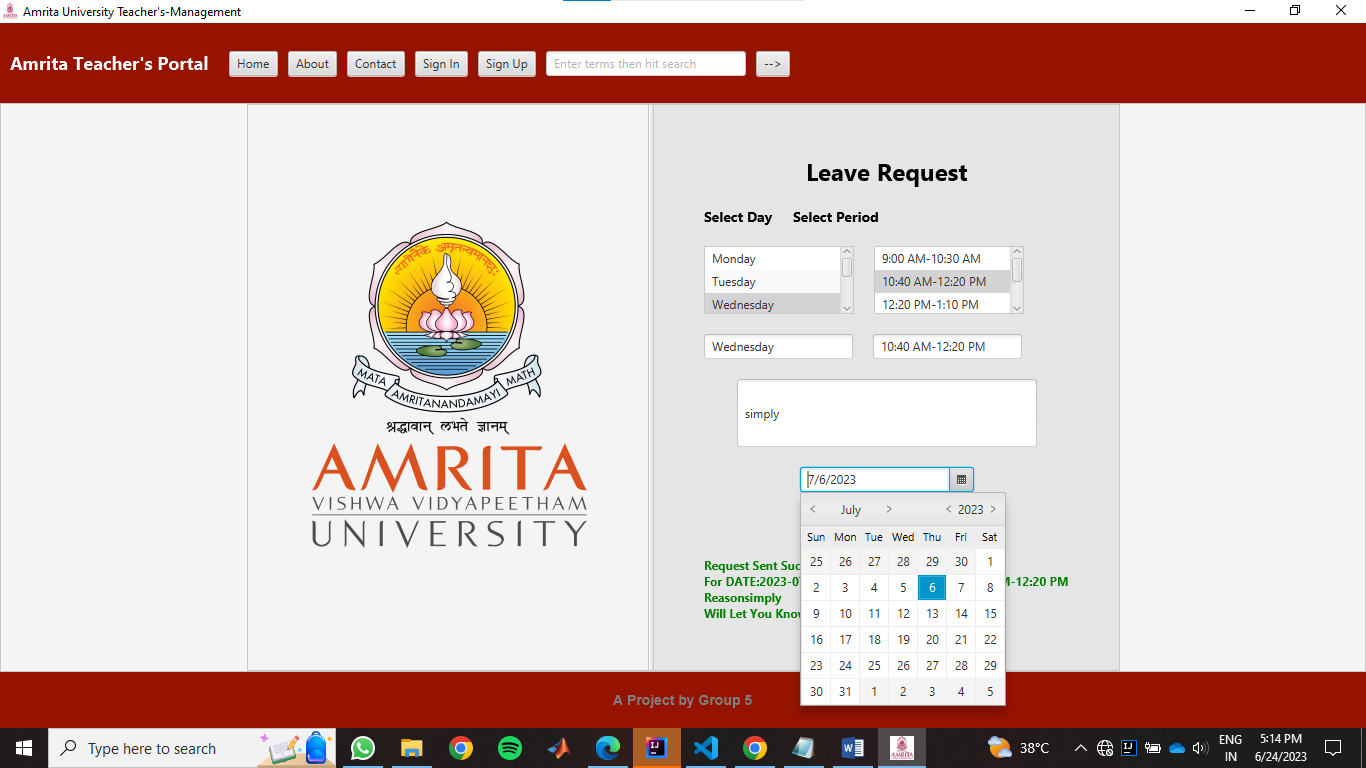


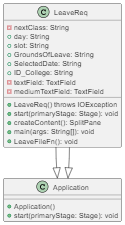
### 1.4.3.11 public class SignInInfoPage





### 1.4.3.12 public class LeaveReq





# 1.5. Methadology.

This part we will will explain the methadology in short using the txt-file, since the core part of this project is the back-end eventhough the front-end look’s a little flashy, the most important part of this project is the back-end. So in this section we will explain the back-end part of the project only, and explain only the basic methadology, and note that the core and the very important part of this project is Back-end, eventhough the Front-end is attractive.

**“** Behind every flashy and attractive front-end, lies the powerful heartbeat of the project - the robust and efficient backend **”,** so in the similar way the heart-of this project lies in the backend.

## 1.5.1.Admin Recruting Teacher’s.

In the above section 1.4. we have clearly shown how each class’s work’s, below section we will explain the logic alone,

|  |  |  |
| --- | --- | --- |
| **name** | **collegeid** | **password** |
| Nilavarasan | 825746 | nilanila123 |
| Manimaran | 830644 | manimani321 |
| Priya | 757712 | 123456pri |

There is a file called user\_data.txt, and date similar to like this stored, in text format seperated by comas. The value’s are stored like below in the txt.file

### 1.5.2.1.Example of How data is Stored-1:

nilavarasan,1234567,nila123

manimaran,123123,mani123

priya,321321,priya123

family,111222333,nilamanipriya

this is how the file’s are stored in txt-file sepearted by coma’s. each data.

## 1.5.2.Adding Details Filled In the Sign-Up Page:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **name** | **College-ID** | **Password1** | **Mail-ID** | **Password2** |
| Nilavarasan | 825746 | nilanila123 | nila012@gmail.com | nilanila321 |
| Manimaran | 830644 | manimani321 | mani@gmail.com | maninila |
| Priya | 757712 | 123456pri | Priya@gmai.com | nilapriya |

In the same file after signing-up, this details are stored. Note this data are stored in the same user\_data.txt file. All this are stored in the same file, eg how this data is stored in the user\_date.txt now.

### 1.5.2.2.Example of How data is Stored-2:

Nilavarasan,825746,nilanila123,nila012@gmail.com,nilanila321

Manimaran,830644,manimani321,mani@gmail.com,maninila

Priya,757712,123456pri,Priya@gmai.com,nilapriya

The data are appended in the same file-with coma’s now.

## 1.5.3.Adding Details Filled Teacher-info Page.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **name** | **College-ID** | **Password1** | **Mail-ID** | **Password2** |
| Nilavarasan | 825746 | nilanila123 | nila012@gmail.com | nilanila321 |
| Manimaran | 830644 | manimani321 | mani@gmail.com | maninila |
| Priya | 757712 | 123456pri | Priya@gmai.com | nilapriya |

Continuation of the above table.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **gender** | **D.O.B** | **contact** | **address** | **Edu-qualify** | **College-id** | **campus** |
| a | a | a | a | a | a | a | a |
| b | b | b | b | b | b | b | b |
| c | c | c | c | c | c | c | c |

This is how the data is stored after the teacher has fille his/her credendetails after sign-up page, note the data is stored in the same user\_data.txt file.

### 1.5.2.3.Example of How data is Stored-3:

Nilavarasan,825746,nilanila123,nila012@gmail.co,nilanila321,a,a,a,a,a,a,a,a,a,a,a

Manimaran,830644,manimani321,mani@gmail.com,maninila,b,b,b,b,b,b,b,b,b,b,b

Priya,757712,123456pri,Priya@gmai.com,nilapriya,c,c,c,c,c,c,c,c,c,c,c

## 1.5.4.Timetable Of each teacher.

Since each teacher has a unique college-ID, so this is the key-logic in storing the time-table of the teacher’s. since the college-ID of all teacher’s are unique, we will take input of the time-table for the the teacher, after he/she click the submit button, the time table is stored as (this is the number that each teacher has)“collegeid.txt”.

So in this fasion the time-table is stored for each teacher, below is a example of how time-table stored for he collegeID 825746,830644.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Day** | **Period1** | **Period2** | **Period3** | **Period4** | **Day** | **Period1** | **Period2** | **Period3** | **Period4** |
| Monday | Break | Class 2 | Class 3 | Break | Monday | Class 1 | Break | Break | Break |
| Tuesday | Class 6 | Class 4 | Class 1 | Break | Tuesday | Break | Class 5 | Class 3 | Break |
| Wednesday | Break | Break | Class 5 | Class 2 | Wednesday | Break | Break | Break | Class 2 |
| Thursday | Break | Class 4 | Break | Break | Thursday | Break | Class 4 | Class 5 | Class 1 |
| Friday | Class 2 | Class 3 | Class 4 | Break | Friday | Class 2 | Break | Class 4 | Break |
| Saturday | Break | Class 1 | Class 3 | Break | Saturday | Break | Class 1 | Class 3 | Class 4 |

This is the timetable of the teacher and it is saved as “825746.txt”, “830644.txt”. this is how this table’s are stored.

### 1.5.4.1.Example of How data is Stored-3:

Example of how it is actually stored in the .txt file, where they are sepersted by comma’s the .txt file.

#### 1.5.4.1.1. Example of 825746.txt

Monday,Break,Class 2,Class 3,Break

Tuesday,Class 6,Class 4,Class 1,Break

Wednesday,Break,Break,Class 5,Class 2

Thursday,Break,Class 4,Break,Break

Friday,Class 2,Class 3,Class 4,Break

Saturday,Break,Class 1,Class 3,Break

#### 1.5.4.1.2. Example of 825746.txt

Monday,Class 1,Break,Break,Break

Tuesday,Break,Class 5,Class 3,Break

Wednesday,Class 1,Class 2,Break,Class 6

Thursday,Break,Class 3,Class 2,Class 1

Friday,Break,Class 2,Class 3,Break

Saturday,Break,Class 6,Class 3,Class 2

## 1.5.5.When teacher Request pass.

Suppose when a teacher request a pass, for a peroid to the admin. That data is stored in another file “LeaveFile.txt”. Below is the format in which the file is stored when a teacher request-pass to the admin.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CollegeID** | **Day** | **Slot(time)** | **Date** | **reason** |
| 298861 | Tuesday | 10:40 AM-12:20 PM | 2023-06-25 | simply |
| 825746 | Tuesday | 9:00 AM-10:30 AM | 2023-06-01 | Sim1 |
| 825746 | Monday | 12:20 PM-1:10 PM | 2023-06-01 | Sim2 |
| 825746 | Monday | 10:40 AM-12:20 PM | 2023-06-01 | Sim3 |

#### 1.5.5.1. Example of LeaveRequest of all Teacher’s:

298861,Tuesday,10:40 AM-12:20 PM,2023-06-25,simply

825746,Tuesday,9:00 AM-10:30 AM,2023-06-01,sim1

825746,Monday,12:20 PM-1:10 PM,2023-06-01,sim3

825746,Tuesday,10:40 AM-12:20 PM,2023-06-01,sim4

This is how the data is tored, in the txt file. Where with the help of this college-ID the admin will know who have given pass request- and he can act accordingly. All the teacher’s who request pass are stored, in this same file. Where suppose if a teacher request’s another pass, then it is appended in this same file.

## 1.5.6.Pass Accept Or Reject By admin.

Now one of the important part of this project is admin accepting the pass, so now this data is feed to the admin, now he know’s which teacher’s has requested pass and on which day etc, using the above file. Now there are two file produced

1. one, file is to notify that respective teacher, that his/her pass is approved or cancelled.
2. Second. File is suppose if the admin has approved any pass, that data’s alone are stored in this file. This for the other teacher’s to notify that there is free-period, at this day,date,time etc.

So basically the first-one is for the teacher who requested pass, and the second file is for send notification to the teacher’s indicating that there is a free-period.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CollegeID | Pass-Status | Day | Period | Date |
| 298861 | approved | Wednesday | 12:20 PM-1:10 PM | 2023-07-06 |
| 886431 | rejected | Monday | 10:40 AM-12:20 PM | 2023-07-09 |

This is how the data is stored in the file.

#### 1.5.5.1. Example of File-type “1”.

298861,approved,Wednesday, 12:20 PM-1:10 PM ,2023-07-06

886431,rejected, Monday ,10:40 AM-12:20 PM,2023-07-06

This is how the it is tored, and with the help of the college-id, and status in the second-column the teacher is informed about the status using notification.

Now the second file is where the pass’s which were approved by the admin is stored, and with the help of this file the other teacher’s will get informed stating that there is a free-period now.

|  |  |  |  |
| --- | --- | --- | --- |
| CollegeID | Day | Period | Date |
| 298861 | Wednesday | 12:20 PM-1:10 PM | 2023-07-06 |

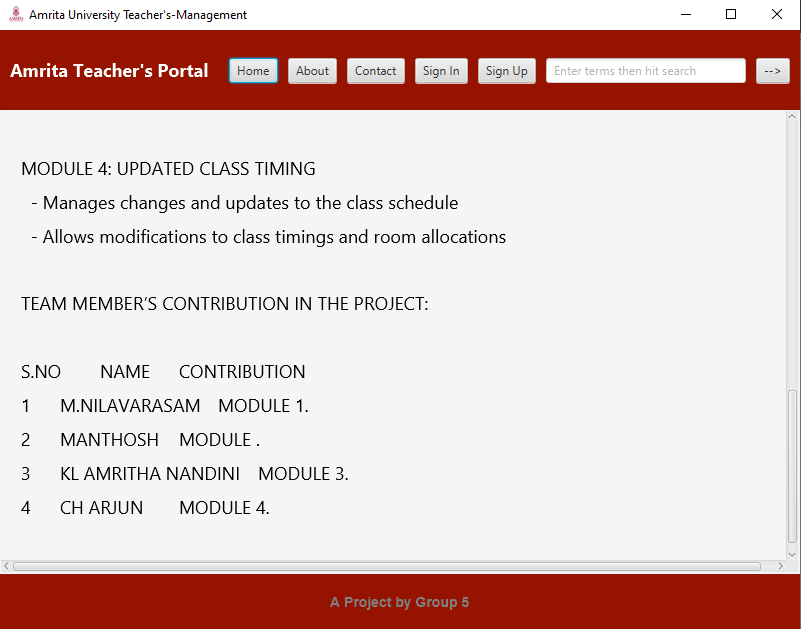
#### 1.5.5.2. Example of File-type “2”.

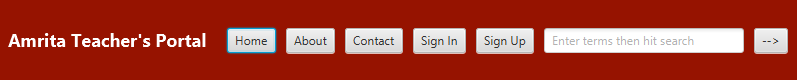
298861,Wednesday, 12:20 PM-1:10 PM ,2023-07-06

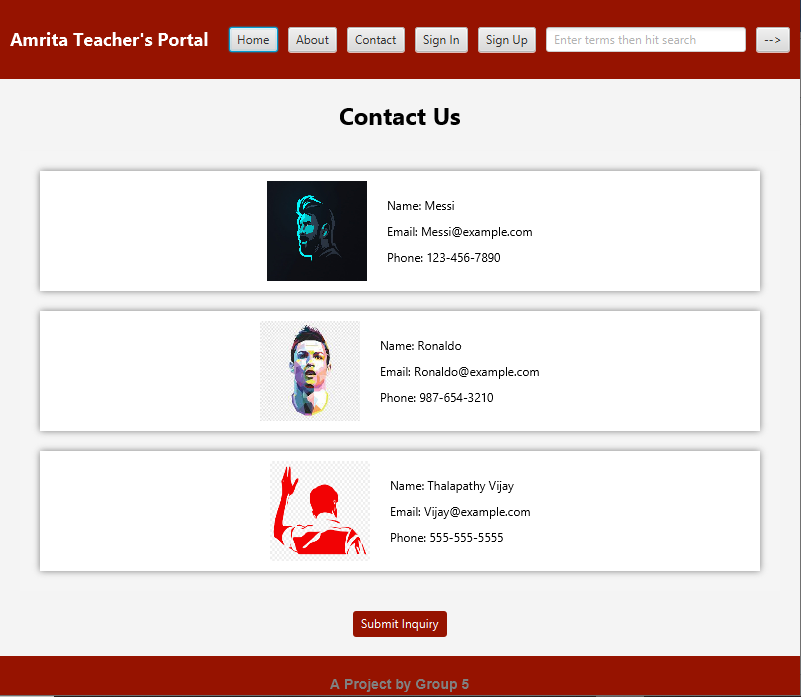
# 1.5. Result.

We have Shown our Output after running each class in the compiler In the above section Itself, this section is just to show how is each class are connected, when this button is clicked, will go to which class. So this section is just the flow of program when runned.

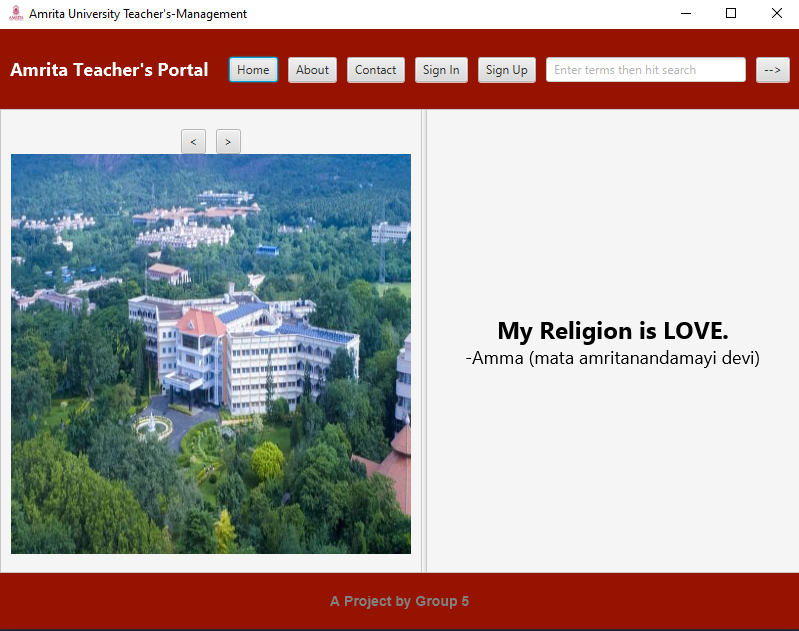
## **1.5.1. When Contact and About Button’s Present in the Header is Clicked.**

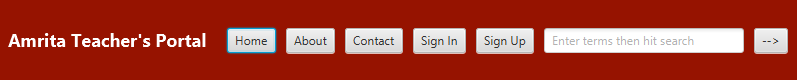


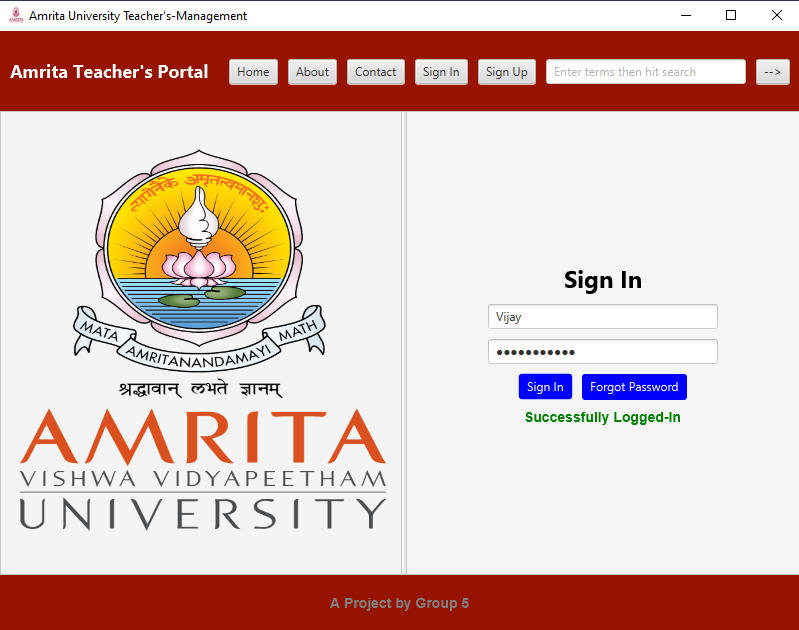




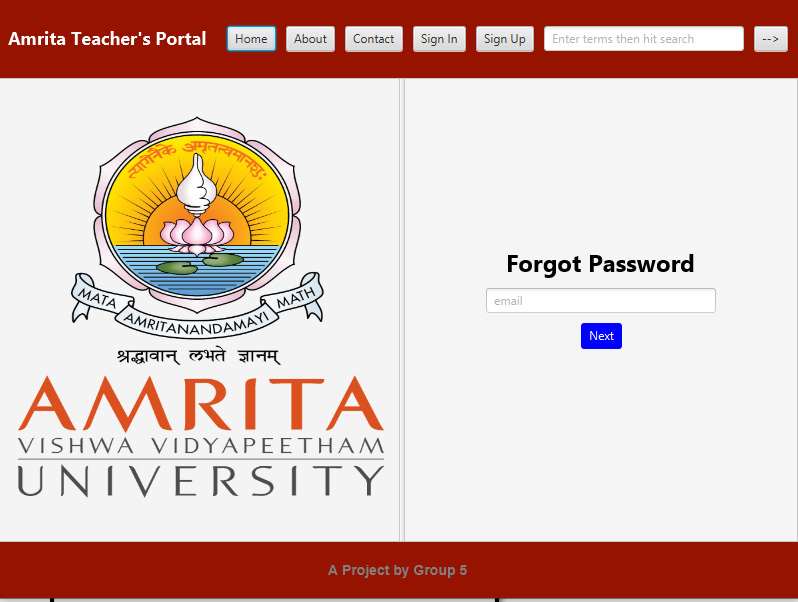
## **1.5.2. When Home and Sign-in Button’s Present in the Header is Clicked.**



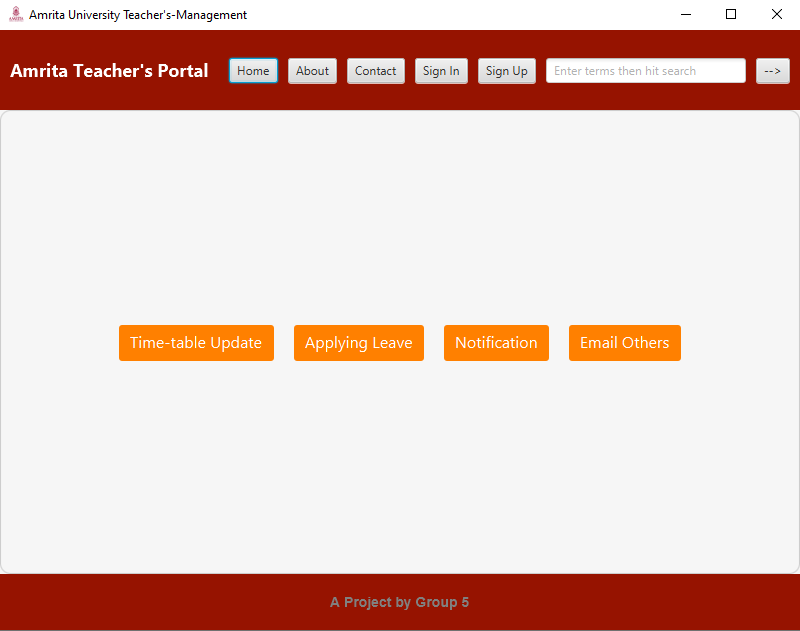




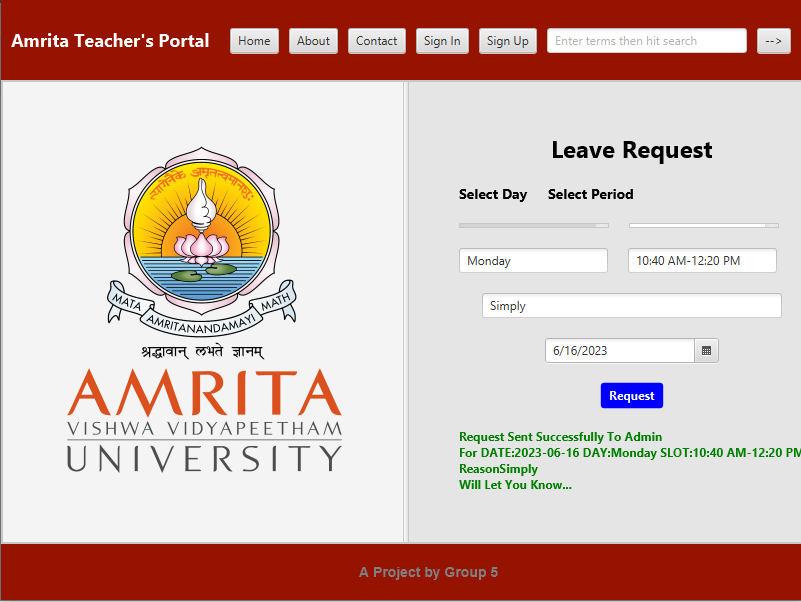
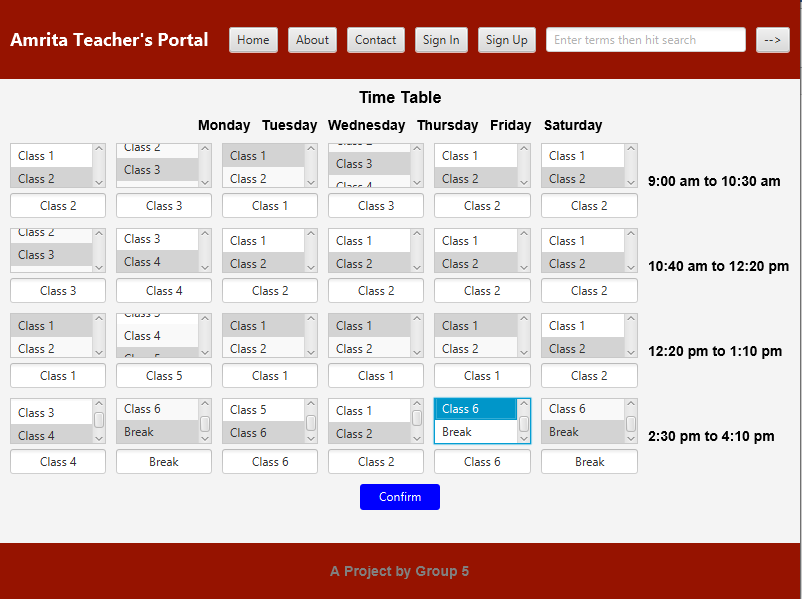
## **1.5.2. When Button Present in Sign-in Page Clicked.**

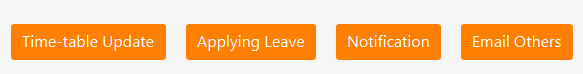
****

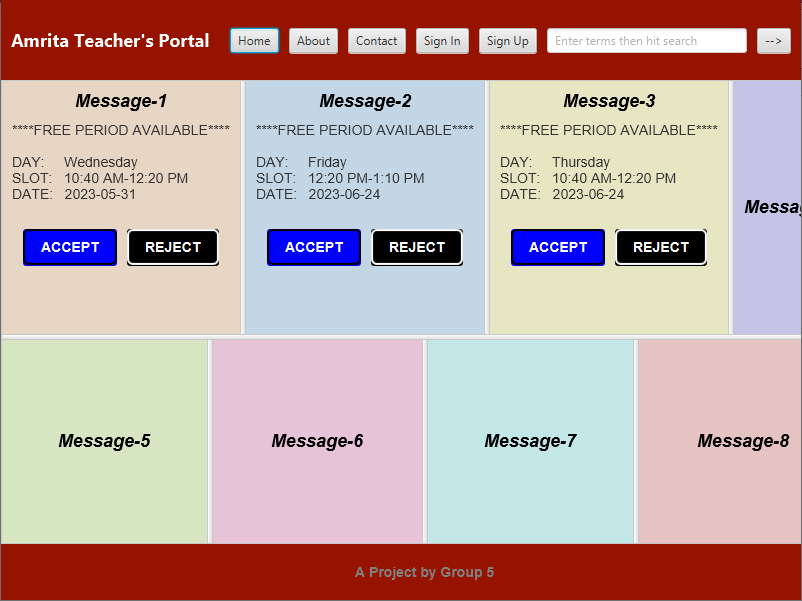
****



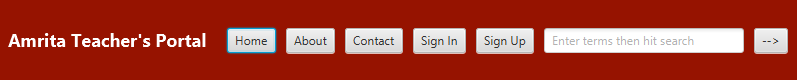
## **1.5.3. When The Button’s Present After Sign-in Are Clicked.**

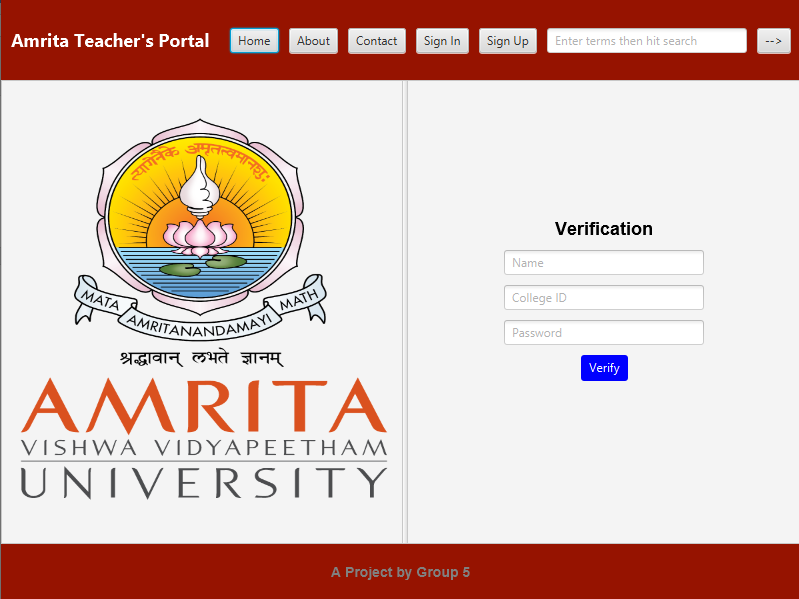




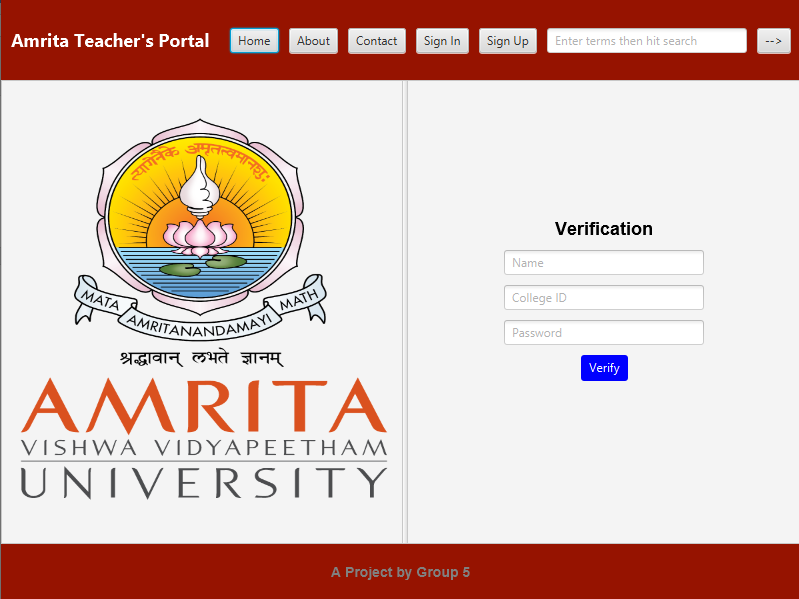


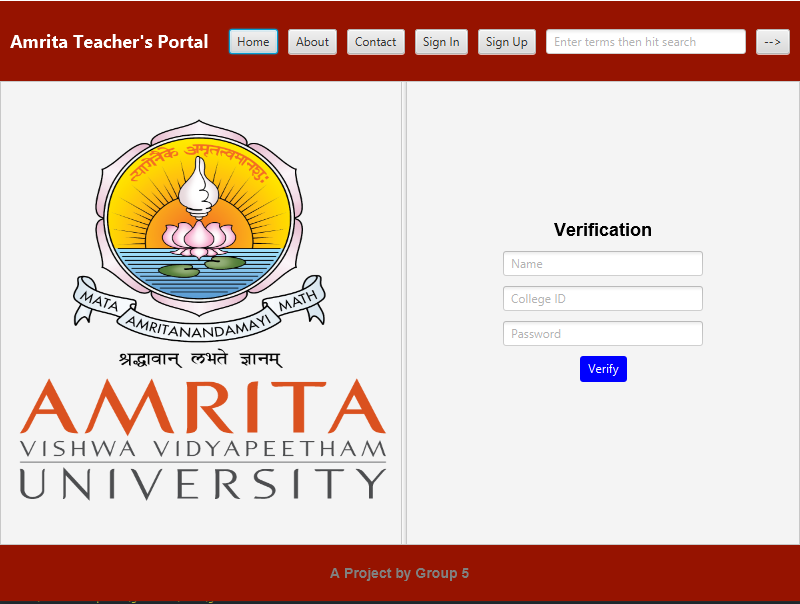
## **1.5.4. When Sign-up Button Present in the Header is Clicked.**

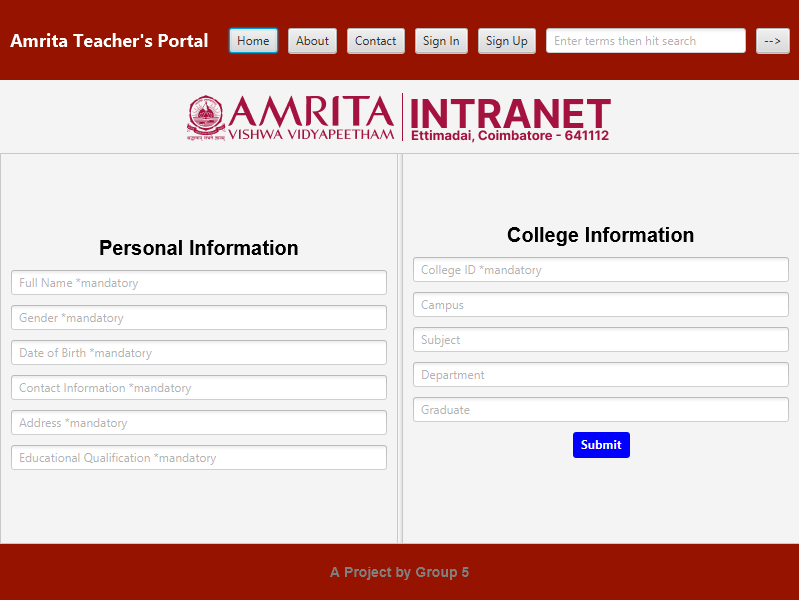




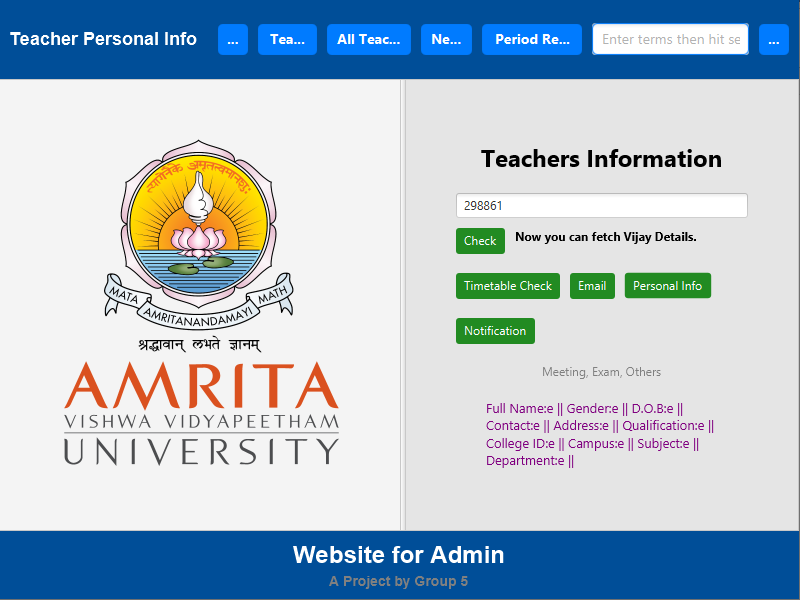
## **1.5.5. Flow Of Sign-up Page**

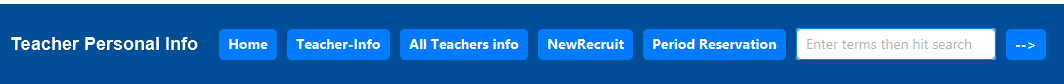


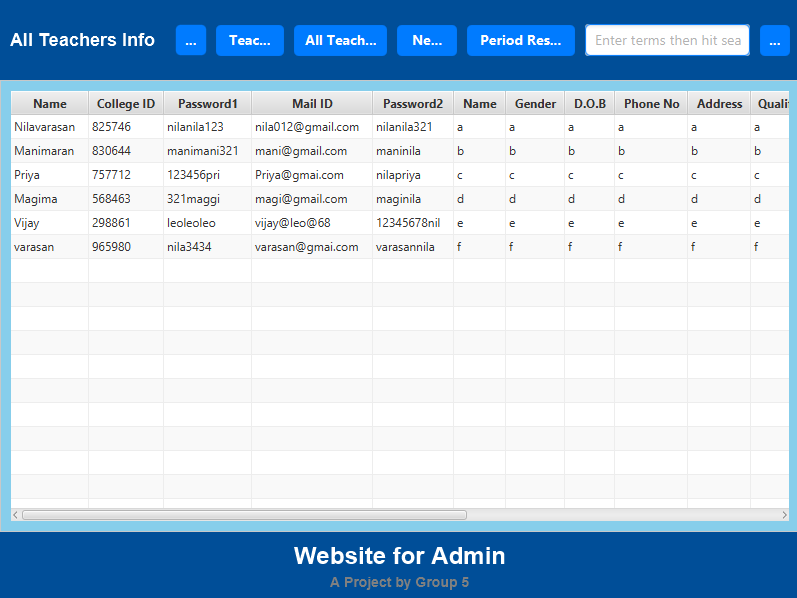




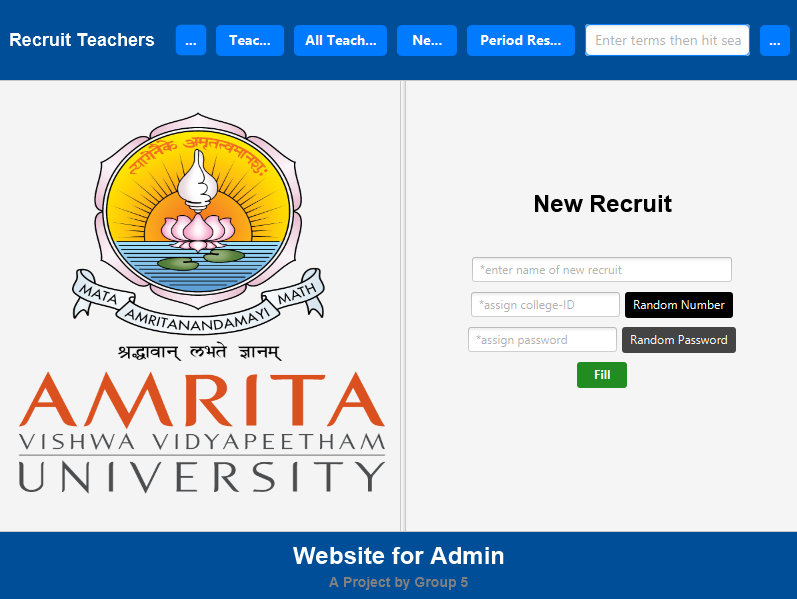
## **1.5.4.When Button’s Present in Header Clicked-1**

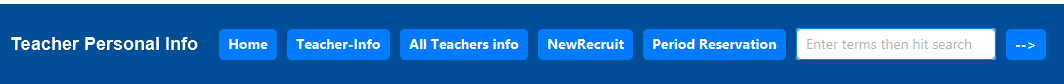


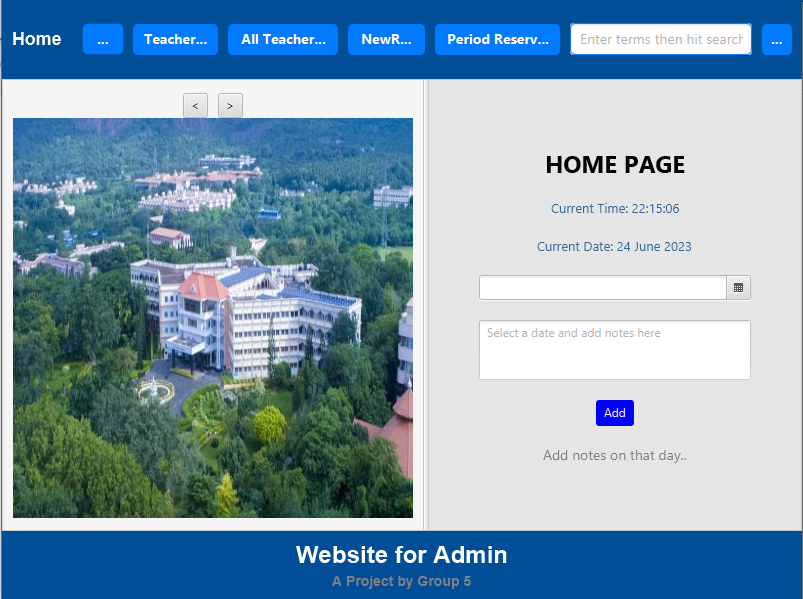




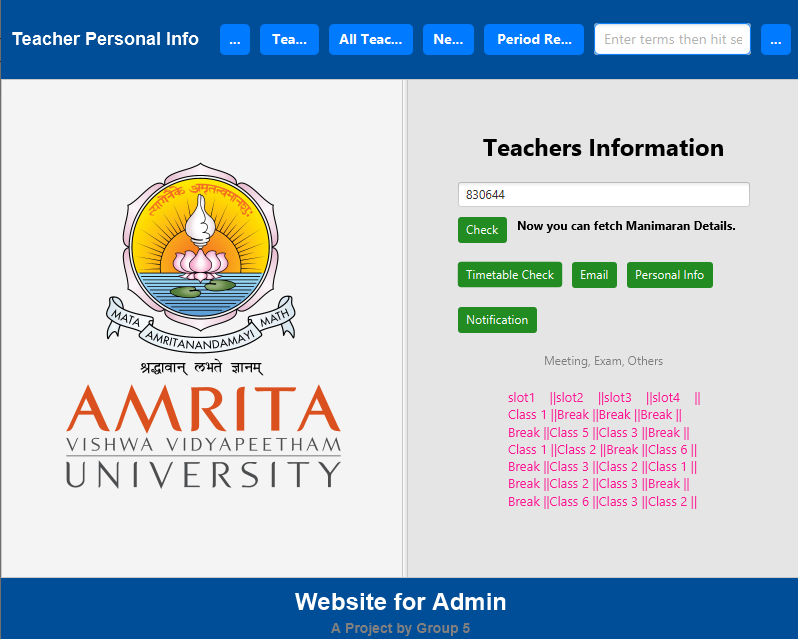
## **1.5.5.When Button’s Present in Header Clicked-2.**

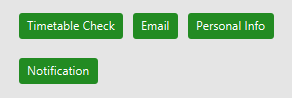


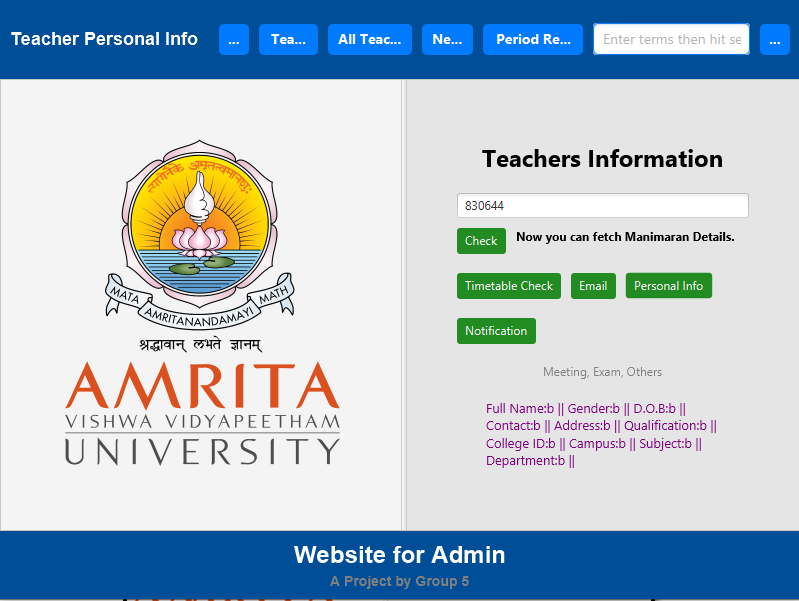




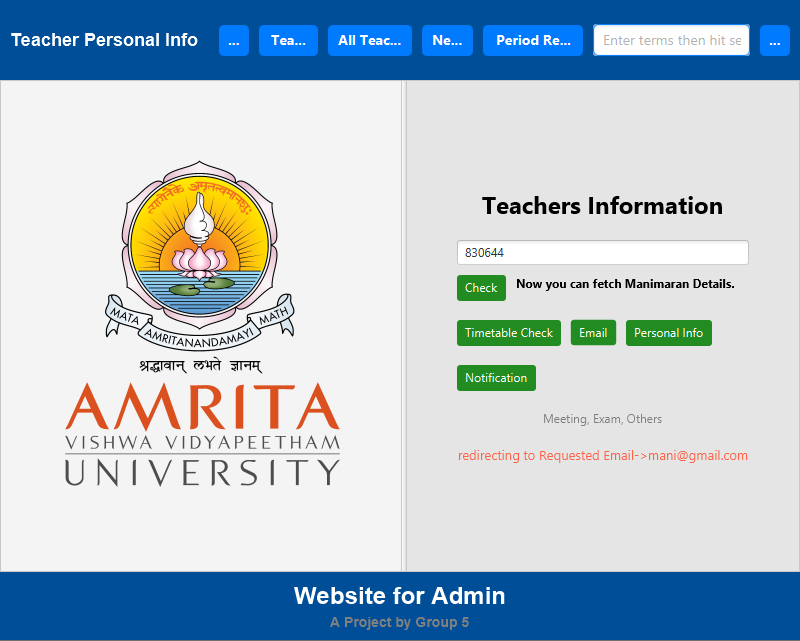
## **1.5.6.When Button’s In Teacher’s Info Clicked-1**

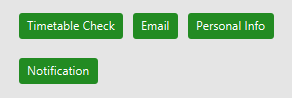


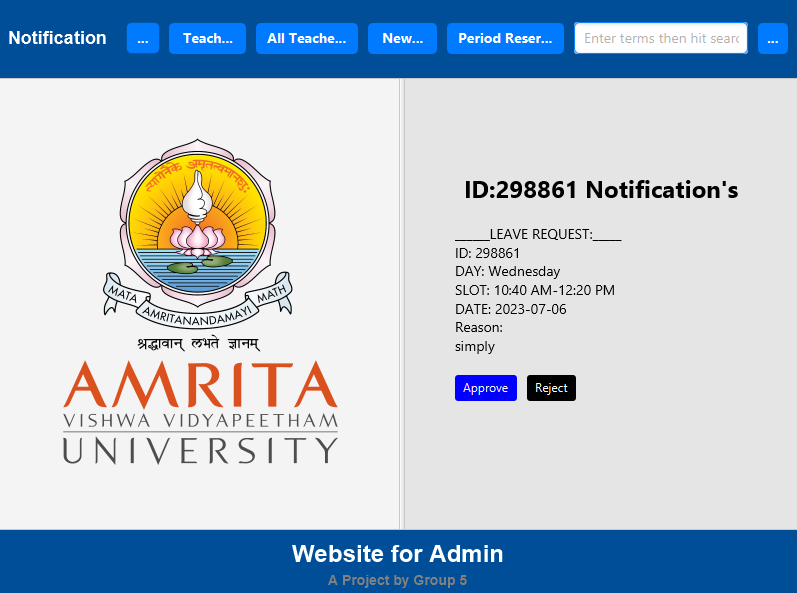




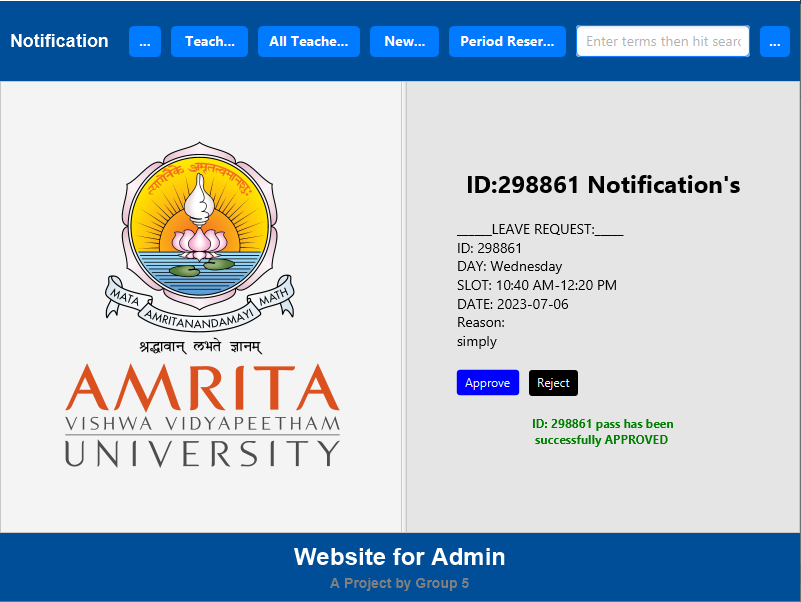
## **1.5.7.When Button’s In Teacher’s Info Clicked-2.**



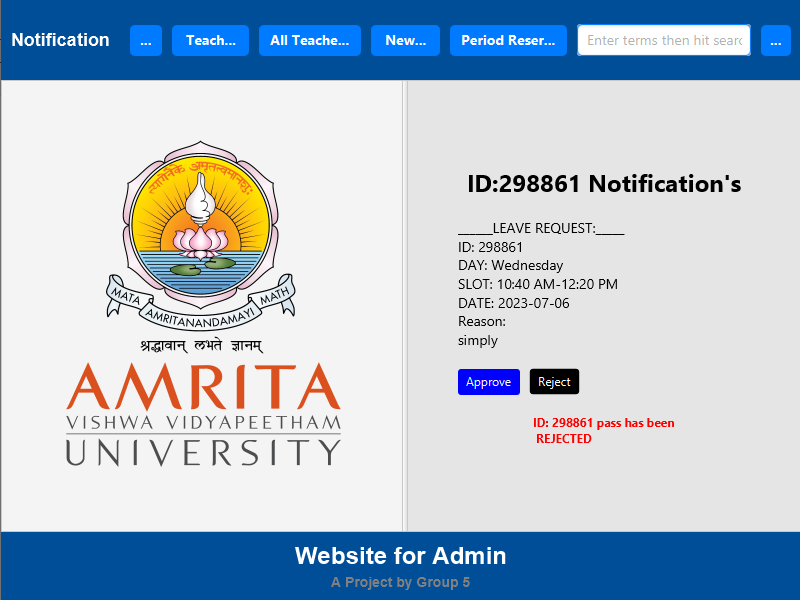




## **1.5.8.When Button’s In Teacher’s Info Clicked-3.**







# 1.6.Conclusion.

In a world filled with mundane class schedules and teacher absences, our team emerged as the unlikely heroes of Amrita University. Armed with Java and a knack for mischief, we set out to vanquish the chaos and restore order to the campus.

With our ingenious Class Schedule Generator module, we turned the tiresome task of finding substitute teachers into a thrilling adventure. Gone are the days of administrators diving into a labyrinth of timetables like Indiana Jones searching for the Holy Grail. Our application swooped in like Batman, scanning schedules and sending signals faster than the Flash on caffeine.

But that's not all, folks! Our Test Reservation, General Meeting, Public Events, and Holiday module became the master of ceremonies, ensuring that exams and impromptu dance parties no longer clash. We banished scheduling conflicts to a dimension where they belong—far, far away from the students' quest for knowledge.

In the Free Period Notification and Reservation module, we channeled our inner matchmakers, playing cupid between free periods and eager teachers. With a swipe left or right (just kidding, it's more like a click), teachers could claim those precious vacant slots and become the heroes their students never knew they needed. The students, unsuspecting pawns in this grand chessboard, received notifications like love letters from Cupid, revealing the substitute teacher assigned to guide them through the abyss of knowledge.

Throughout this epic adventure, we wielded the mighty power of Java, our trusty sidekick IntelliJ by our side. Our data storage, hidden away in the depths of text files, remained loyal companions—like loyal henchmen, minus the evil intentions.

However, even superheroes have their kryptonite. Our project's one-sided nature sometimes feels like a comedy skit without a punchline. We yearn for the day when teachers can communicate their desires, requesting a day off with the eloquence of Shakespeare or at least a funny meme.

While our project may not be ready to conquer the world outside the campus gates, we dream of expanding its reach. We envision a day when our application connects to the internet, spreading its superhero powers across the digital realm. Privacy concerns shall be defeated, ensuring that only the chosen ones can access the sacred halls of teacher schedules.

As our project grows and evolves, we acknowledge the challenges that lie ahead. Text files, like ancient scrolls, may buckle under the weight of endless schedules. We may need to tap into the hidden powers of advanced data storage solutions to keep our superhero headquarters in order.

In the end, our laughter-infused Java adventure has brought smiles and a touch of mischief to Amrita University. We've transformed the mundane into the extraordinary, the chaotic into the organized, and the ordinary into the heroic. As the curtain falls on this chapter, we eagerly await the next installment, armed with wit, laughter, and a burning desire to make education a truly epic journey. Stay tuned for the sequel, where the heroes of Amrita face even more outrageous challenges!

# 1.7. Result.

* Sharan, Kishori, and Peter Späth. *Learn JavaFX 17: Building User Experience and Interfaces with Java*. Apress, 2022.
* Schildt, Herbert. "Java: the complete reference." (2007).
* Sharan, Kishori, and Peter Späth. *Learn JavaFX 17: Building User Experience and Interfaces with Java*. Apress, 2022.
* chat-gpt.
* forefront-AI.