SOFTWARE ENGINEERING

COURSE CODE: CSE1005

GATE PASS MANAGEMENT SYSTEM

Team:

- 18BCD7008 - Kathal Aditya Rajendra
- 18BCD7044 - Sharaj Raja Chandran
- 18BCD7094 - Athresh K
- 18BCD7018 - Amandeep Sharma
- 18BCE7343 - Taduru Sindhupriya

VIT AP
AMARAVATI
www.vitap.ac.in

INDEX

OBJECTIVES

In order for a college to run in an efficient manner it is necessary that it has proper systems in place which supports it. With our project, we intend to develop a software which will ensure smooth and secure flow of information and data throughout the organization. As an extension it will also reduce the amount of paperwork and last-minute rush. It will keep concise and reliable records which will also be viewable by the concerned users. All of this would be achieved by the development of our project; Gate Pass Management System.

The main goal of this software is to make the overall process of raising request for leaves by students much more efficient, less tedious and well recorded. By implementing our project all of this should be made possible as the raised requests will instantly travel amongst the concerned users and there will be no need for the use of mails and manual intervention. As all the activities related to this process happens virtually using the software hence everything is well recorded and stored. Additionally, user interface has been developed in such a way to reveal only necessary and vital information which makes the process on users end much quicker.

EXISTING SYSTEM

There are many gate pass systems in effect all over the world in innumerable institutes and organisations. This is because such a system taker care of a very trivial but necessary part of the user’s activities in relation with the institute/organisation. All of these systems are unique even though they might share some resemblance overall. The major model of the application remains the same but the intricate details differ in accordance with the structure of the organisation/institute. Hence the Software which we intend to develop will too be unique but at the same time follow certain broader aspects of the existing systems.

NEED FOR NEW SYSTEM

Every organisation and institute are unique in its structure and working. As a result, it is necessary to implement a Gate Pass Management System that has been specifically created for that institute/organisation in order to cater to all its particular needs. Currently our college has no such system implemented which makes the process of taking leaves for students very difficult which hinders their capacity to put their mind at other important task as it consumes time and requires an unnecessary amount of work. It also makes the process of recording the said leaves difficult as most of the process happens on paper and then needs to be digitalized.

Hence a Gate Pass System needs to be developed specifically for our college by keeping all its intricate structure and features in mind.

HARDWARE and SOFTWARE REQUIREMENT

CLIENT SIDE:

RAM	512 MB
Harddisk	10 GB
Processor	1.0 GHz

SERVER SIDE:

RAM	1 GB
Harddisk	20 GB
Processor	2.0 GHz

FUNCTIONAL SPECIFICATIONS (MODULE WISE DISCUSSION)

We have identified six modules for our project so far. Below listed are the modules in our project in the decreasing order of their priorities.

Login Module: As the name suggests this module's main functionality is providing a login and signup portal for the users. Parents, Faculty, Hostel Wardens etc. will be able to access their respective accounts and additionally Students will also be able to create their accounts through this module. Case insensitive comparison will be done for the usernames whereas case sensitive comparison will be done for the passwords. On the off chance of invalid data being entered into the portal a corresponding error message will also be displayed.

Mentor Module: The mentors will be able to handle requests from their mentees through this module. They can also add comments to the requests which will be viewable by the students. Additionally, the statistical information about a particular mentee will also be made available to the mentor.

Hostel Warden Module: This module will enable the Hostel Warden to view and interact (accept/reject) with requests from students belonging to that particular hostel. Also, the Warden can view the total number of requests raised from the students living in that particular hostel. The live status (in hostel, on leave, running late, etc.) of a student is displayed through this portal. They can also send a request back to the mentors for reconsideration and attach a private comment which will be viewable only by the mentor.

Student Module: The primary function of this module will be to enable Students to raise and view requests. They will also be able to see all other data related to their previous requests and activities.

Guard Module: This module gives the College Guards access to only view the approved requests and also is the primary data entry source for the in-time and out-time of the students.

Computational Module: This module will be responsible for the mathematical computations and generating search results based on different queries risen by the other modules. It cannot be directly accessed by any user; it can only be invoked by other modules.

Admin Module: It will provide the user with administrative powers which include adding new Mentors and Hostel Wardens. It will also give a pan-college statistical view of the data which is held by the software.

SRS (Software Requirement specification)

1. INTRODUCTION

1.1 PURPOSE

The purpose of this project is to provide the College, a software solution that delivers a scalable, secure, and reliable system that manages and reports Hostel Wardens, Parents, Students and Mentors about various requests and other information. The following SRS document will outline the features of the "Gate Pass Management System" and the requirements that the project will adhere to, in developing the software for the organization.

1.2 INTENDED AUDIENCE

This document is used by developers for understanding the requirements of the project; also, the intended users of this document are project managers for planning and scheduling, testers to generate test cases, document writers for preparation of user manual and for other end-users/stakeholders to validate their requirements.

1.3PROJECT SCOPE

Gate Pass Management System is aimed at managing requests from Students. The system is intended to automate the process of addition, updating and reporting new request and updating corresponding records and views of the database. The objective of the project is to help the college speed up the procedure of Students Gate Pass approval, data and information processing and reduces the operational cost. Also, this software reduces the paper work, thus showing the concern towards conserving the environment.

The goal of the project is to reduce the time and cost incurred in an access to the database.

1.4 OPERATING ENVIRONMENT

Programming Language	JAVA

IDE	Eclipse EE
Database	Oracle 19c

1.5 DESIGN AND IMPLEMENTATION CONSTRAINTS

Design Constraints

Design mechanisms are limited to the capabilities of Object-Oriented Paradigm (i.e. Java) and Oracle 19c DB. The user interface must be viewable on a monitor with a 1024x768 resolution or larger.

Safety and Security Considerations:

Since personal information of Mentors and students, and official information about Hostels will be contained and accessed. Safety and Security considerations need to be taken into account that personal and sensitive information are not accessed by unauthorized users.

2. SYSTEM FEATURES

The end user having the software installed at his desktop should be able to login using the valid credentials. Post validation of the credentials entered; the system allows access to the database. Upon access the user should be able to perform operations like insert, delete, and update request as well as Hostel Warden and Mentor records into the database, and also search the database. Provision should be provided for the user to query the database to obtain various views of the data.

2.1 SYSTEM FEATURES

The Gate Pass Management System maintains details about Students, Hostel Wardens, Hostel, Mentors and Security Guards. This section describes the associated Functional requirements in the table below based on priorities

#	Module Name	Description	Priority	Functional Requirement
1.	Login	This will be used for authenticating users and provide an interface to the various users in order to login.	9	**Registration Page**: • Should be able to register new students and **<u>add records in the student database</u>**. • Students are required to enter: ○ Name : Maximum 25 Character Minimum 2 Character ○ Roll No.: 9 alphanumeric string ○ Email: should contain @vitap.ac.in ○ Gender: Choose appropriate option ○ Mentor: Mentor ID should start with EM ○ Hostel Type: Should match the gender specified ○ Room Number: 4 digit numeric ○ Password: Maximum length 10 Minimum length 5 • All fields are mandatory. • If same student tries to register again “Already Registered” error is displayed.

				Login Page: • Should accept the username and password • The same constraints will be followed for Name and Password during login. • A case insensitive comparison of the username to ease login and case sensitive comparison for password. • If the correct user id and password is supplied then the portal corresponding to the user is displayed. • If an invalid user id or password is entered then the system should display error message “Invalid ID or password”. • Username Format: Student : 18XXX7020 Parent : P18XXX7020 Mentor : EM______ Guard: ES_______ Hostel Warden: EHW_____

2.	Menu	This feature behaves as a user interface to display the features of the database and guides the user to access the database	8	Prompts the user to make a choice with the 6 types of menus available. 1. Request Menu: Update, delete or add records to the request database. 2. Summary Menu: View statistical information regarding requests and leaves from the join of records from various databases. 3. Request Status Menu: View and update records in the request database. 4. Admin Menu: View, modify, update, add and delete data from hostel warden, hostel, guard and mentor database. 5. Time Update Menu: View records in the request database and update corresponding records in leaves database.

				6. Profile Menu: View data from corresponding database and update password.
3.	Request Menu	Provides an interface to the request database and modifies it accordingly.	8	This menu will help to add, update and delete request. **Add Request:** 1. Students can raise request. 2. Fields to fill while filling a request: From date : dd/mm/yyyy To date : dd/mm/yyyy Destination: Maximum 10 characters Request Type: Three options will be provided 3. Request ID will be automatically generated. It will be 23 characters long and alpha-numeric in nature and the format is “RollNo-Date-Time” 4. Request ID cannot be duplicated.

				5. Valid date should be mentioned in the text field otherwise subsequent error will be thrown for invalid input or date. To date should be greater than From date. 6. If complete information is not filled before confirmation of generating a new request then an error will be thrown. 7. Three Types of Requests purposes: Medical, Outing and Vacation. **Update Request:** 1. Only mentor and hostel warden can use this service. 2. All pending requests will be shown to the users and options in form of buttons will be provided with each request. 3. Options to Mentors : Accept, Reject

				4. Options to Hostel Warden: Accept Reject Reconsider **Accept Request** 1. Available to two actors; Hostel Warden, Mentor. 2. If Mentor accepts the request then it gets transferred to the Hostel Warden. If the Hostel Warden also accepts the request then it is sent to the Leaves Database. **Decline Request** 1. Available to three actors; Hostel Warden and Mentor. 2. If any of the actors decline the request then it is updated in the request database. **Reconsider Request** 1. Only available to hostel warden. 2. Can send a request for reconsideration back to mentor with an internal private comment attached 5. Request will be shown will following details:

				Request Id Student Name Student Registration Number From Date To Date Destination Request Type Room Number Hostel Name
4.	Summary Menu	This feature generates a log sheet of status all previous requests and current requests of a particular student and various mathematical values.	8	**View Request:** The student user view previous requests. The search option is only available to the Hostel Warden, Mentor and Admin Users. **Single Log format:** 1. Each log has 4 properties: a. Student Name b. Student Registration Number c. Request date d. Request status

				e. Mentor name **Search by student** 1. Accept student first name and last name **or** Registration number of 9 characters. 2. Show all logs of that particular student order by date. **Search by Mentor** 1. Accept Faculty number which should start with ‘EM’ . 2. Show all logs having mentor name as that particular faculty. **Filter by status** 1. Shows all logs filtered by pending requests first 2. Filter by date Shows all logs filtered by latest date first **View All Databases:**

				Various values that will help the users will also be provided. These values will be created with joins of various databases. Student User: Number total requests Number of active requests Number of Approved Requests Number of Rejected Request Times late Mentor User: Total Active requests Total requests Warden User: List of student out of college Total active requests Total requests Parent User: Number total requests Number of active requests Number of Approved Requests

				Number of Rejected Request Times late
5.	Request Status Menu	Provides an interface to view and modify the status of a request.	7	This menu is only visible to the student users where they can view all current requests and present status and also delete them. **Delete Request:** Requests that have not yet reached the hostel warden portal can be deleted by a student from the records database.
6.	Admin Menu	This interface provides administrative powers to the user who can view, delete and update various databases.	7	**Add Mentor:** 1. All fields are mandatory. 2. Mentor ID will start with ‘EM’. 3. Mentor name should not exceed the 25-character limit. **Add Hostel Warden:** 1. All fields are mandatory. 2. Hostel Warden ID will start with ‘EHW’. 3. Warden name should not exceed the 25-character limit. 4. Can only be added if corresponding hostel exists.

				Add Hostel: 1. All fields are mandatory. 2. Hostel ID will start with ‘M/L H’ followed by a number. 3. Two Hostel type entries valid; Men, Ladies. **Delete Mentor:** 1. All fields are mandatory. 2. Mentor ID will start with ‘EM’. **Delete Hostel Warden:** 1. All fields are mandatory. 2. Hostel Warden ID will start with ‘EHW’. 3. Cannot be deleted if only one warden is present for that hostel. **Update Hostel Warden:** 1. All fields are mandatory. 2. Hostel can be updated and Id should start from ‘M\LH’. 3. Warden Id should start from ‘EHW’ **Update Hostel:**

				1. All fields are mandatory. 2. Hostel Id should start with "M\LH' 3. Only hostel total strength can be updated.
7.	Time Update Menu	This module will allow guard to check and confirm the in-time and out time for the student.	6	**Update Request:** This menu is only visible to the "Guard" users. **In-time Menu:** 1. The Student ids should be unique and should be 9 digits long. 2. It should contain 3 alphabets and 6 numeric digits 3. In case of wrong ID, it should display error message. 4. The in-time needs to be verified and should be b/w 6:00AM – 8:00PM if not the leave database is updated to "running late". (only for Outing type requests) 5. The in-time needs to be verified and should be before the leave ends (end date), if not the leave database is updated to "running late". (only for Vacation and Medical type requests)

				6. To automate and make this process fairer, the security guard just needs to click in the “Register Time” button and the system time will be saved in the database. **Out-time Menu:** 1. View all outing requests for a particular day. 2. Can confirm the ID’s of student on leaving form. 3. To automate and make this process fairer, the security guard just needs to click in the “Register Time” button and the system time will be saved in the database. 4. The button will not be activated until In-Time for a request is not registered.
8.	Profile Menu	This interface will help users to update various fields related to their corresponding profile.	6	**Update and View** This menu will help all users to update password for their account and view various information related to the profile which will be fetched from various databases.

				Only Security and Parent user will not be shown this menu. Profile Will contain following information: **Student user:** Name Registration Number Email Id Hostel Name Room Number Hostel Warden Name Hostel Warden Email Mentor Name Mentor Email Times Late **Mentor User:**

				Name Employee Id Email Id Mentee Names Mentee Registration Numbers **Hostel Warden User:** Name Employee ID Employee ID Hostel Name

3. EXTERNAL INTERFACE REQUIREMENTS

3.1 USER INTERFACES

The GPMS should be a simple application used by its stakeholders with/without much technical knowledge.

Login Interface: This interface prompts the user to authenticate his/her credentials by asking username and password, so that only authorized users can log in to the system. Also, it displays error message and exits if the given credentials are wrong.

Request Interface: This interface will help in viewing, updating, adding requests to the database.

Profile Interface: This interface provides the users with the option to update the existing password and also view details of itself.

Time Update Interface: This interface provides the users with the option to update in time and out time of students on various requests.

Admin Interface: This interface provides the users with the option to add a new mentors, hostel warden and update the existing details and also view details of all the existing.

Request Status Interface: This interface provides the users with the option to add a new request, update the existing request details and also view details of all the existing requests.

Summary Interface: This interface provides the users with the option to view statistical information about the requests that have been made and history of all the requests.

3.2 HARDWARE INTERFACES

This software does not require any direct hardware interfaces.

3. SOFTWARE INTERFACES

Operating System	
Programming Language	Java 1.8
IDE	Eclipse EE
Database	Oracle 19c

In this application the software interfaces are:

- Windows 7 operating system serves as a platform for development and deployment of the College Management system software.
- Eclipse EE can be used for implementing the programming language.
- JDK 1.8 can be used for front end development.

3.4 COMMUNICATIONS INTERFACES

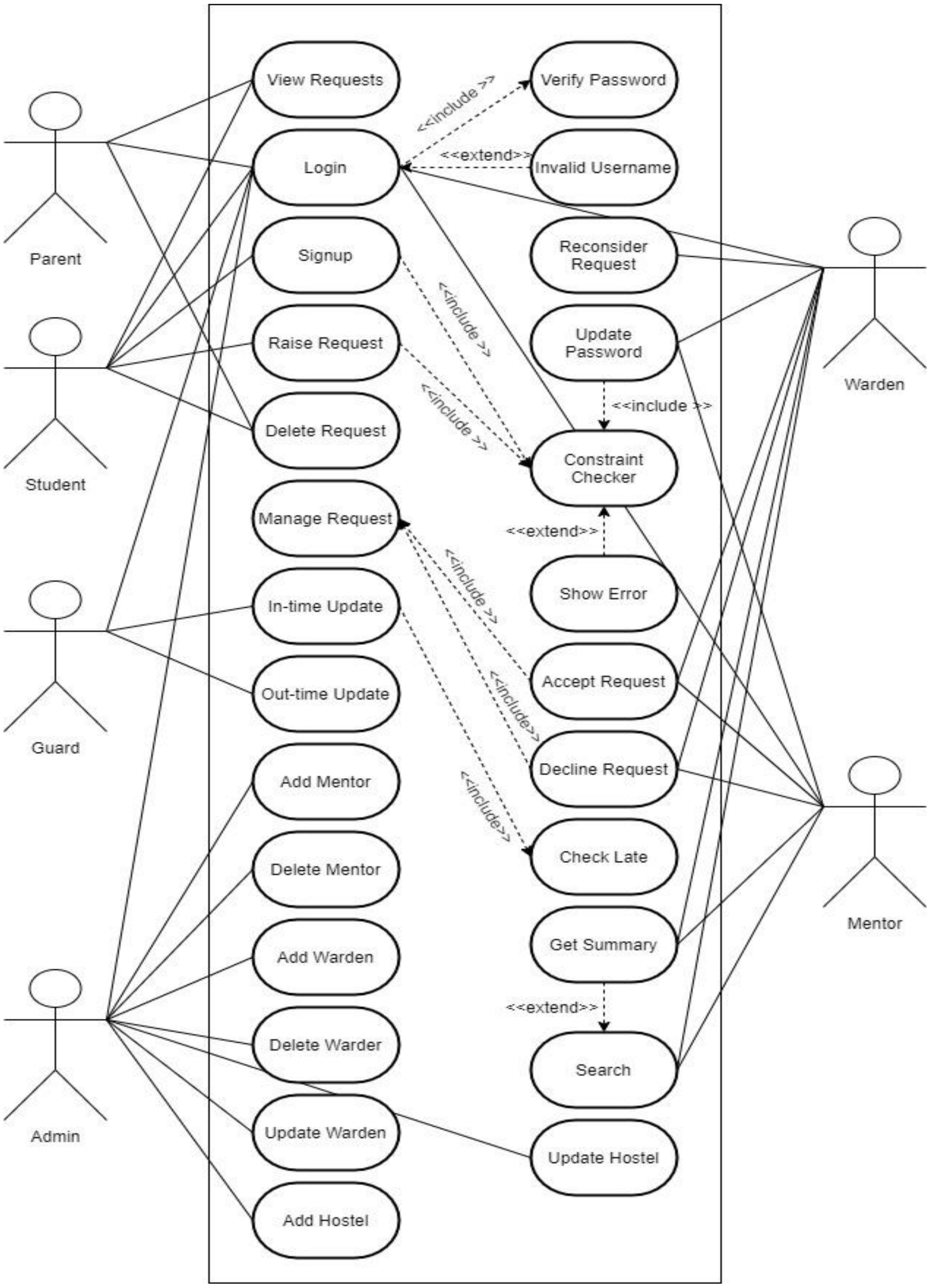
This software is a standalone system hence does not require any communication interfaces.

2. OTHER NONFUNCTIONAL REQUIREMENTS

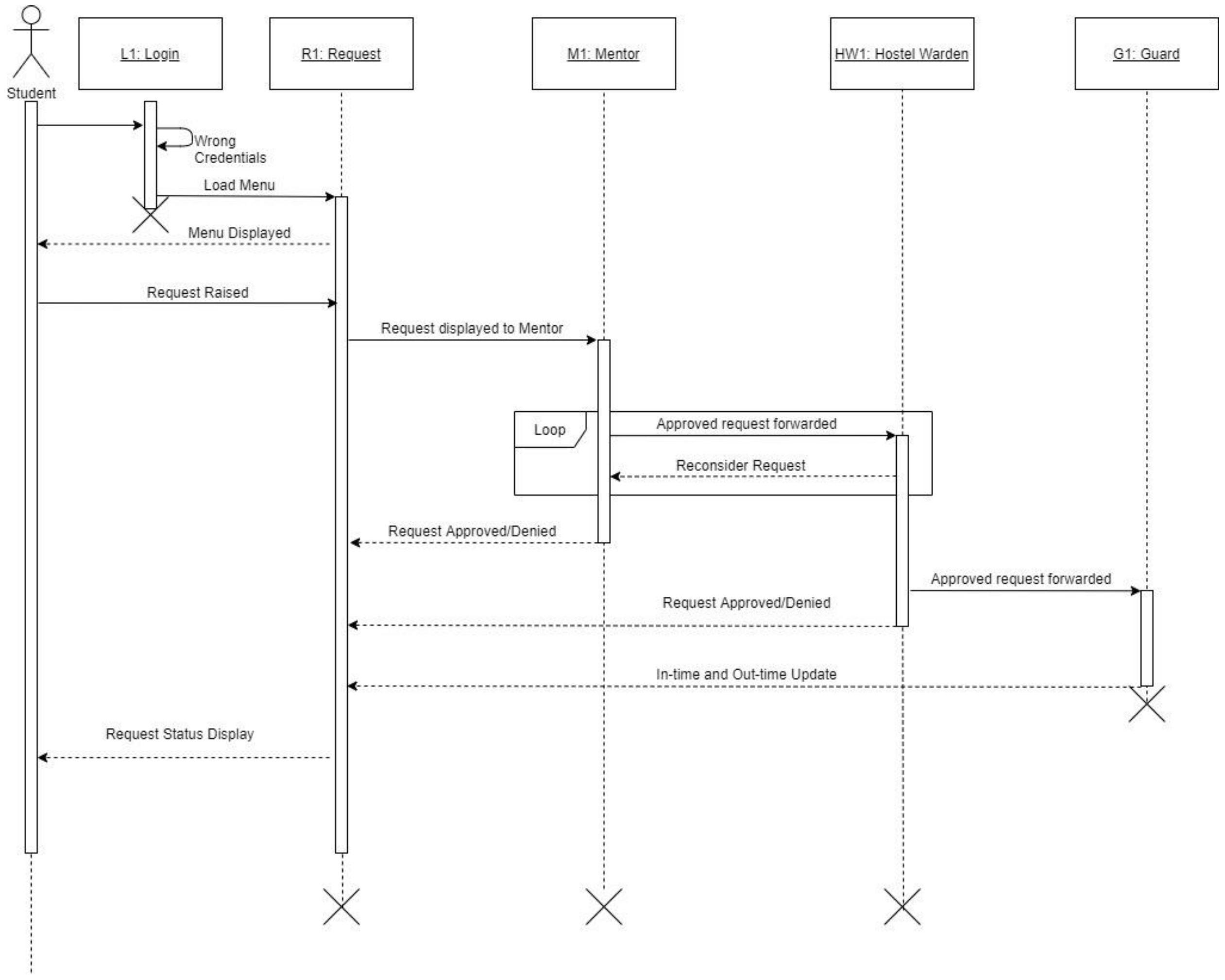
#	Type of Requirement	Details
1	Performance Requirements	The Gate Pass Management system should be able to run on a standalone computer. As the database grows the intended performance with respect to report generation and accessing of data should not be affected.
2	Availability Requirements	This software is completely replacing the traditional system which is in place. Hence the system must be able to provide accurate data to all the users with valid credentials. The scheduled maintenance time should not require more than 3hrs per week and make the software unavailable.
3	Training Requirements	The GUI should be user-friendly and provide self-learning with a training time of not more than 3hrs.

UML DIAGRAMS

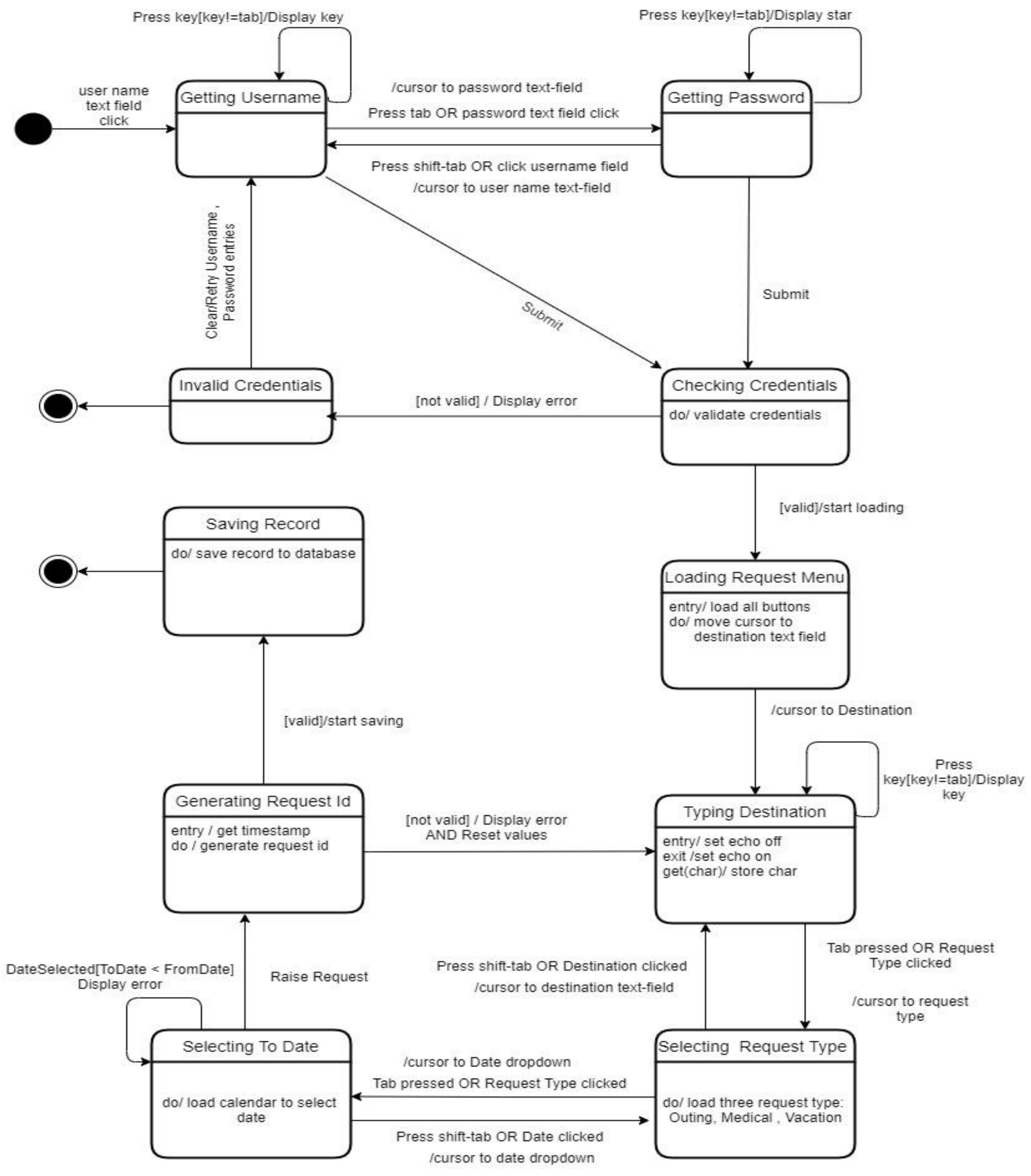
USE CASE DIAGRAM



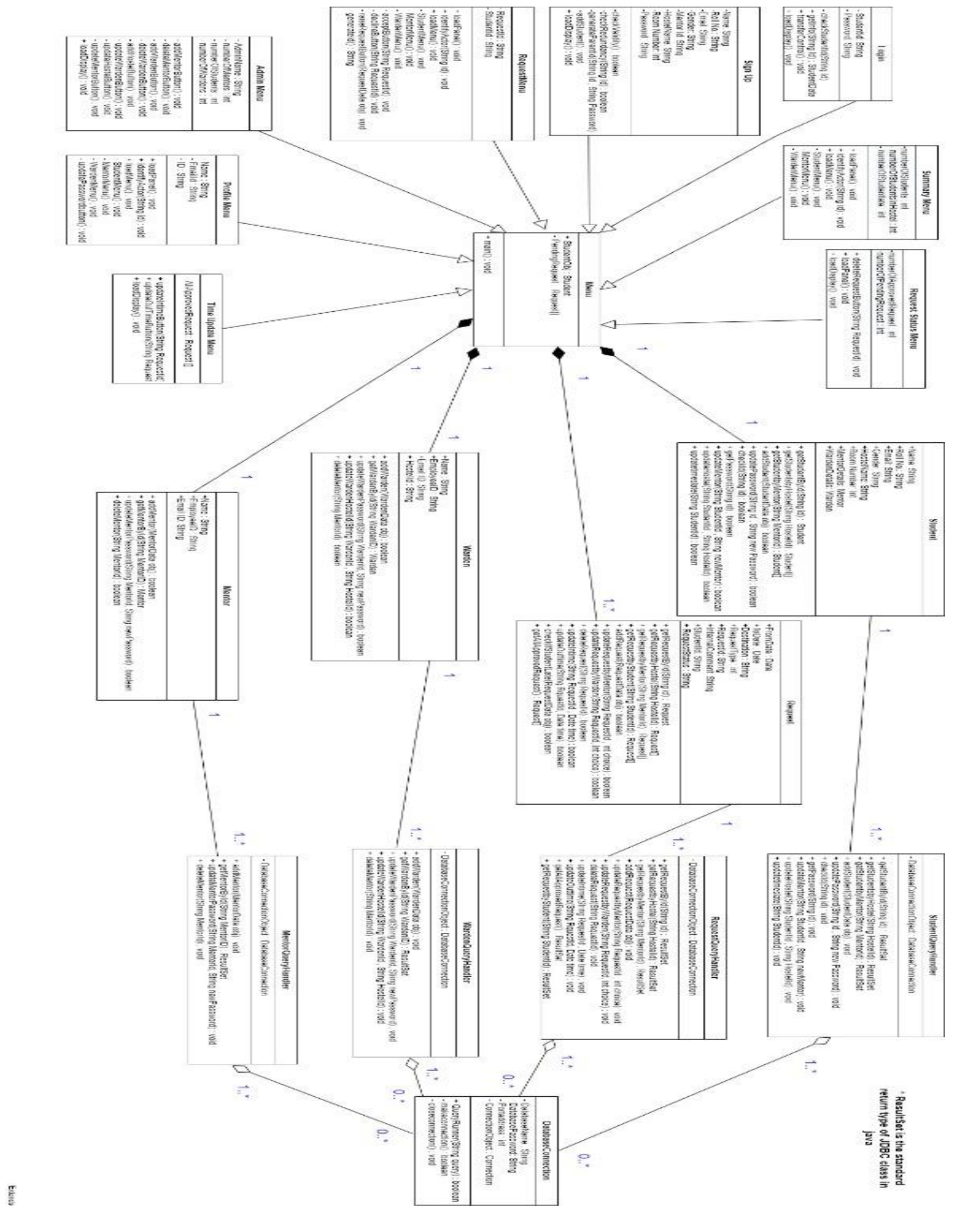
SEQUENCE DIAGRAM



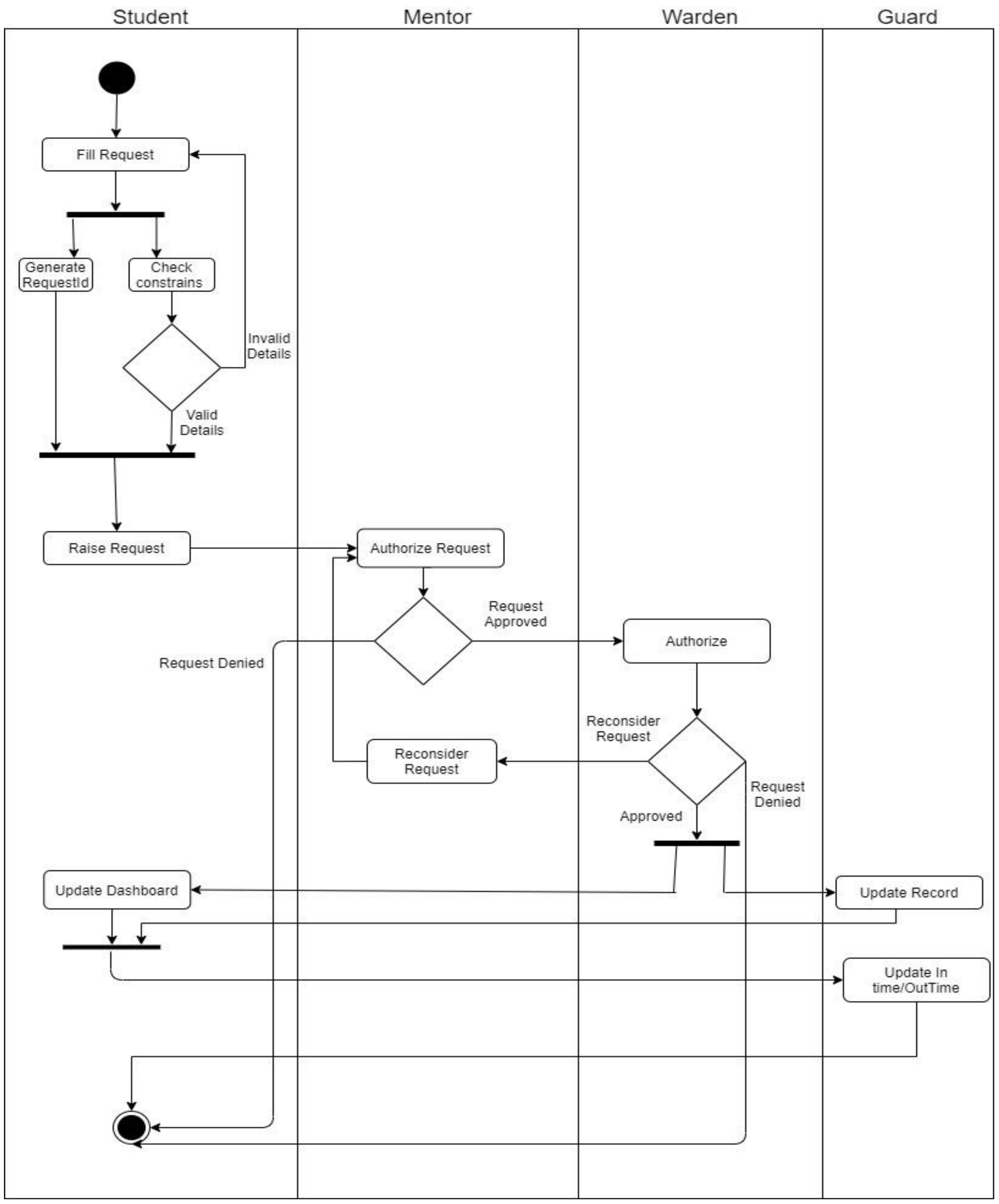
STATE DIAGRAM


Press key[key!=tab]/Display key
Press key[key!=tab]/Display star
user name text field click
Getting Username
/cursor to password text-field
Press tab OR password text field click
Getting Password
Press shift-tab OR click username field
/cursor to user name text-field
Clear/Retry Username , Password entries
Submit
Submit
Invalid Credentials
[not valid] / Display error
Checking Credentials
do/ validate credentials
[valid]/start loading
Saving Record
do/ save record to database
Loading Request Menu
entry/ load all buttons
do/ move cursor to destination text field
/cursor to Destination
[valid]/start saving
Press key[key!=tab]/Display key
Generating Request Id
entry / get timestamp
do / generate request id
[not valid] / Display error AND Reset values
Typing Destination
entry/ set echo off
exit /set echo on
get(char)/ store char
Tab pressed OR Request Type clicked
/cursor to request type
DateSelected[ToDate < FromDate] Display error
Raise Request
Press shift-tab OR Destination clicked
/cursor to destination text-field
Selecting To Date
do/ load calendar to select date
/cursor to Date dropdown
Tab pressed OR Request Type clicked
Selecting Request Type
do/ load three request type: Outing, Medical , Vacation
Press shift-tab OR Date clicked
/cursor to date dropdown

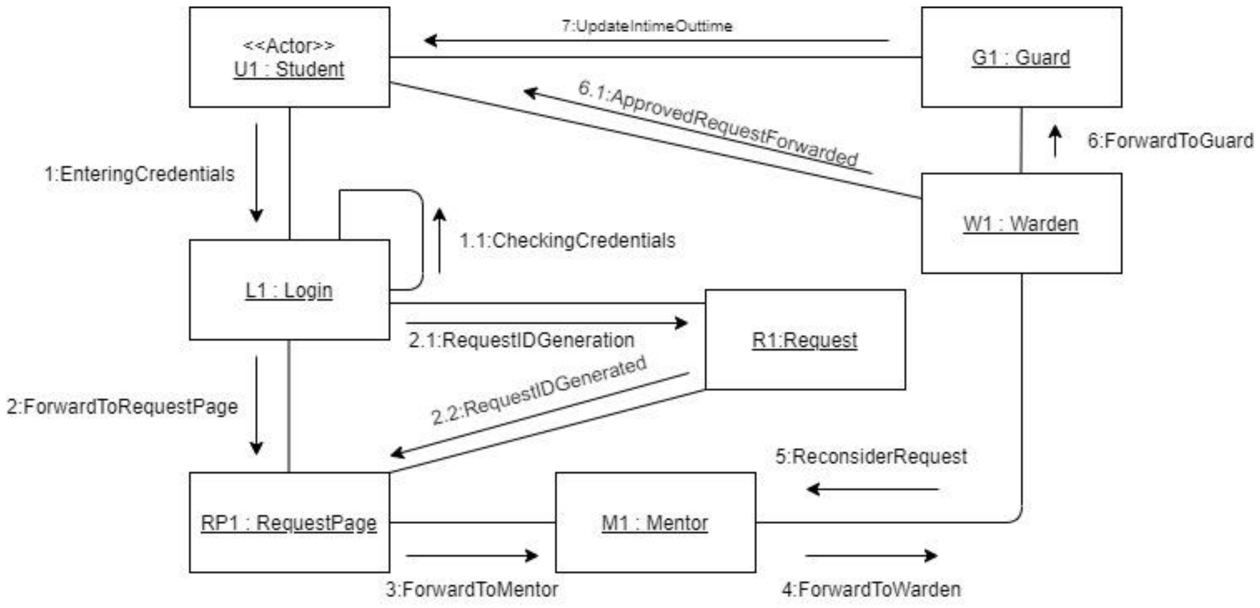
CLASS DIAGRAM



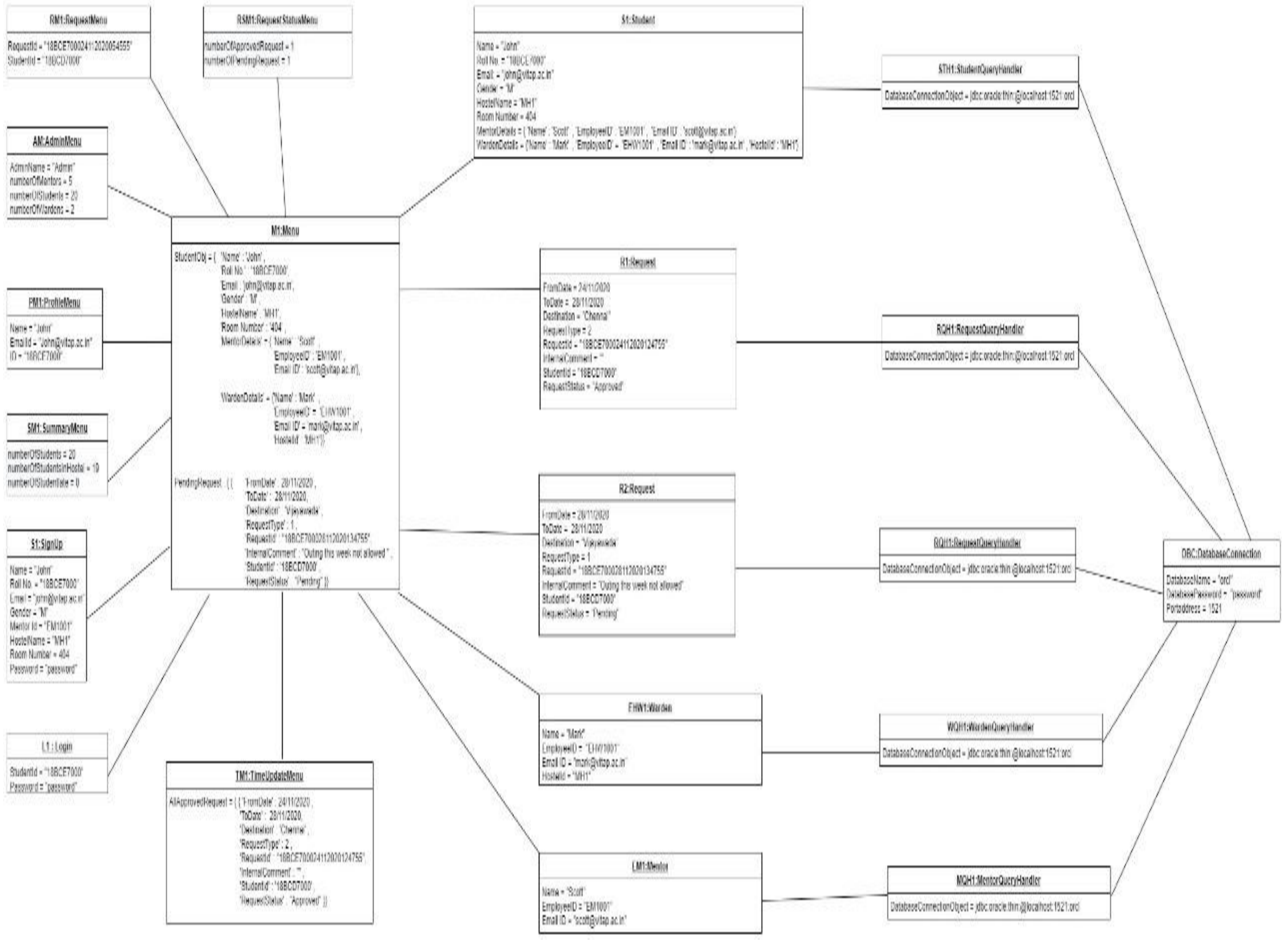
ACTIVITY DIAGRAM



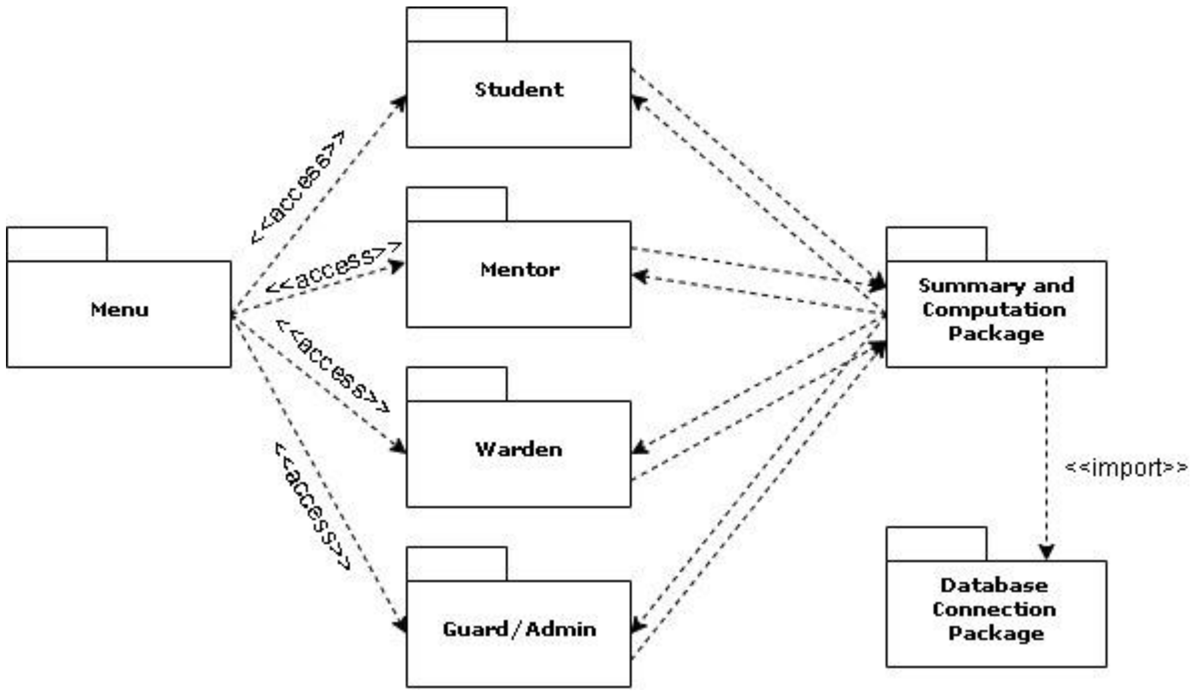
COLLABORATION DIAGRAM



OBJECT DIAGRAM



PACKAGE DIAGRAM



CODING

Database Module:

1) DatabaseConenction Class

```
package se.dbconnect;

import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Connection;

public class DatabaseConnection
{
	private String databasename = "orcl";
	private String databasePassword = "password";
	private int portAddress = 1521;
	private String username = "c##se";
	private String connectionUrl = 
"jdbc:oracle:thin:@localhost:"+portAddress+":"+databasename;
	Connection connectionObject;

	public ResultSet QueryRunner(String query) throws Exception
	{
		boolean status =  makeConnection();
		if(status)
		{
			try
			{
				Statement stmt=connectionObject.createStatement();
				ResultSet rs=stmt.executeQuery(query);
				return rs;
			}
			catch (SQLException e)
			{
				closeConnection();
				e.printStackTrace();
				throw new SQLException();
			}
		}
		else
		{
			return null;
		}
	}
	private boolean makeConnection()
	{
```

```
            try
            {
               connectionObject = DriverManager.getConnection(connectionUrl ,
username , databasePassword);
            }
            catch (SQLException e)
            {
                  return false;
            }

            return true;
      }
      private void closeConnection()
      {
            try
            {
                  connectionObject.close();
            }
            catch (SQLException e)
            {

            }
      }
}
```

2.) StudentQueryHandler Class

```
package se.dbconnect;

import java.sql.ResultSet;
import java.sql.SQLException;
import se.Student.StudentData;

public class StudentQueryHandler
{
      private DatabaseConnection DatabaseConnectionobject;

      public StudentQueryHandler()
      {
            DatabaseConnectionobject = new DatabaseConnection();
      }
      public ResultSet getStudentById(String id) throws Exception
      {
            id = id.toUpperCase();
            String query = "select * from student where studentid = '"+id+"'";
            System.out.println(query);
            ResultSet rs=DatabaseConnectionobject.QueryRunner(query);
            return rs;
      }
      public ResultSet getStudentbyHostel(String HostelId) throws Exception
```

```
	{
		HostelId = HostelId.toUpperCase();
		String query = "select * from student where studentid = " + 
"'"+HostelId+"'";
		System.out.println(query);
		ResultSet rs=DatabaseConnectionobject.QueryRunner(query);
		return rs;
	}
	public ResultSet getStudentbyMentor(String MentorId) throws Exception
	{
		MentorId = MentorId.toUpperCase();
		String query = "select * from student where studentid = " + 
"'"+MentorId+"'";
		System.out.println(query);
		ResultSet rs=DatabaseConnectionobject.QueryRunner(query);
		return rs;
	}
	public void addStudent(StudentData obj , String Password) throws Exception
	{
		String query = "insert into student values('" + obj.Id +"' , '"+ 
obj.Name +"' , '"+ obj.gender+"' , '"+obj.Email+"' , '"+Password+"' , 0 ,  0 , " + 
obj.RoomNumber + 
					", '" + obj.MentorDetails.EmployeeId + "','" + 
obj.HostelName + "')";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public void checkId(String id) throws Exception
	{
		ResultSet rs = getStudentById(id);
		rs.next();
		String getid = rs.getString(1);
	}
	public String getPassword(String id) throws Exception
	{
		ResultSet rs = getStudentById(id);
		rs.next();
		String password = rs.getString(5);
		return password;
	}
	public void updatePassword(String id , String newPassword)  throws Exception
	{
		String query = "UPDATE student SET StudentPassword = '"+newPassword+"' 
WHERE StudentId = '"+id+"'";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public void updateMentor(String StudentId , String newMentor) throws Exception
	{
		String query = "UPDATE student SET MentorId = '"+newMentor+"' WHERE 
StudentId = '"+StudentId+"'";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public void updateHostel(String StudentId , String HostelId) throws Exception
	{
		String query = "UPDATE student SET HostelId = '"+HostelId+"' WHERE 
StudentId = '"+StudentId+"'";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public void updatetimeslate(String StudentId) throws Exception
	{
		ResultSet rs = DatabaseConnectionobject.QueryRunner("select late from 
student where studentid = '"+StudentId+"'");
		rs.next();
		int newcount =  rs.getInt(1) + 1;
		String query = "UPDATE student SET late = '"+newcount+"' WHERE StudentId 
= '"+StudentId+"'";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
}
```

3.) RequestQueryHandler Class

```
package se.dbconnect;

import java.sql.ResultSet;
import java.sql.SQLException;
import se.request.RequestData;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.text.DateFormat;

public class RequestQueryHandler
{
	private DatabaseConnection DatabaseConnectionobject;
	public RequestQueryHandler()
	{
		DatabaseConnectionobject = new DatabaseConnection();
	}
	public ResultSet getRequestById(String id) throws Exception
	{
		id = id.toUpperCase();
		String query = "select * from request where RequestID = '"+id+"'";
		System.out.println(query);
		ResultSet rs=DatabaseConnectionobject.QueryRunner(query);
		return rs;
	}
	public ResultSet getRequestbyHostel(String HostelId)  throws Exception
	{
		HostelId = HostelId.toUpperCase();
```

```
             String query = "select * from RequestHostel where hostelId = 
'"+HostelId+"'";
             System.out.println(query);
             return DatabaseConnectionobject.QueryRunner(query);
      }
      public ResultSet getRequestbyMentor(String MentorId) throws Exception
      {
             MentorId = MentorId.toUpperCase();
             String query = "select * from Requestmentor where mentorId = 
'"+MentorId+"'";
             System.out.println(query);
             return DatabaseConnectionobject.QueryRunner(query);
      }
      public void addRequest(RequestData obj) throws Exception
      {
             String todate = new SimpleDateFormat("dd-MM-yyyy").format(obj.toDate);
             String fromdate = new SimpleDateFormat("dd-MM-
yyyy").format(obj.fromDate);
             String query = "insert into 
request(requestid,studentid,destination,fromdate,todate,intime,outtime,internalcommen
t,status,requesttype) "
                           + "values 
('"+obj.RequestId+"','"+obj.StudentId+"','"+obj.Destination+"',TO_DATE('"+fromdate+"' ,
'dd-mm-yyyy'),TO_DATE('"+todate+"' , 'dd-mm-yyyy'),"
                           + "NULL,NULL,NULL,1,"+obj.RequestType+")";
             System.out.println(query);
             DatabaseConnectionobject.QueryRunner(query);
      }
      public void updateRequestbyMentor(String RequestId , int choice) throws 
Exception
      {
             RequestId = RequestId.toUpperCase();
             String query = "UPDATE request SET status = '"+choice+"' WHERE RequestId 
= '"+RequestId+"'";
             System.out.println(query);
             DatabaseConnectionobject.QueryRunner(query);
      }
      public void updateRequestbyWarden(String RequestId, int choice , String 
comment) throws Exception
      {
             RequestId = RequestId.toUpperCase();
             String query = "UPDATE request SET status = '"+choice+"' WHERE RequestId 
= '"+RequestId+"'";
             System.out.println(query);
             DatabaseConnectionobject.QueryRunner(query);

             query = "UPDATE request SET internalcomment = '"+comment+"' WHERE 
RequestId = '"+RequestId+"'";
             System.out.println(query);
             DatabaseConnectionobject.QueryRunner(query);
      }
      public void deleteRequest(String RequestId) throws Exception
      {
             RequestId = RequestId.toUpperCase();
```

```
		String query = "DELETE from request where requestId = '"+ RequestId
+"'";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public void updateIntime(String RequestId , Date time) throws Exception
	{
		RequestId = RequestId.toUpperCase();
		DateFormat dateFormat = new SimpleDateFormat("dd-mm-yyyy hh:mm:ss");
	 String strDate = dateFormat.format(time);
	 String query = "UPDATE request SET intime = '"+strDate+"' WHERE RequestId =
'"+RequestId+"'";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);

	}
	public void updateOuttime(String RequestId, Date time) throws Exception
	{
		RequestId = RequestId.toUpperCase();
		DateFormat dateFormat = new SimpleDateFormat("dd-mm-yyyy hh:mm:ss");
	 String strDate = dateFormat.format(time);
	 String query = "UPDATE request SET outtime = '"+strDate+"' WHERE RequestId =
'"+RequestId+"'";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public ResultSet getAllApprovedRequest() throws Exception
	{
		String query = "select * from request where status = 6";
		System.out.println(query);
		return DatabaseConnectionobject.QueryRunner(query);
	}
	public ResultSet getRequestbyStudent(String StudentId) throws Exception
	{
		StudentId = StudentId.toUpperCase();
		String query = "select * from request where studentId = '" + StudentId +
"'";

		System.out.println(query);
		return DatabaseConnectionobject.QueryRunner(query);
	}
	public void checkId(String Id) throws Exception
	{
		ResultSet rs = getRequestById(Id);
		rs.next();
		String getid = rs.getString(1);
	}
}
```

4.) WardenQueryHandler Class

```java
package se.dbconnect;

import java.sql.ResultSet;
import se.warden.WardenData;

public class WardenQueryHandler
{
	private DatabaseConnection DatabaseConnectionobject;
	public WardenQueryHandler()
	{
		DatabaseConnectionobject = new DatabaseConnection();
	}
	public void addWarden(WardenData obj , String password) throws Exception
	{
		String query = "insert into warden values('" + obj.EmployeeId + "' , ' "
+ obj.name + "' , '" + obj.Emailid + "' , '" + password + "' , '" + obj.HostelId
+"')";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public ResultSet getWardenById(String WardenID)  throws Exception
	{
		WardenID = WardenID.toUpperCase();
		String query = "select * from warden where WardenId = '"+WardenID+"'";
		System.out.println(query);
		ResultSet rs=DatabaseConnectionobject.QueryRunner(query);
		return rs;
	}
	public ResultSet getWardenByHostelId(String HostelID) throws Exception
	{
		HostelID = HostelID.toUpperCase();
		String query = "select * from warden where HostelId = '"+HostelID+"'";
		System.out.println(query);
		ResultSet rs=DatabaseConnectionobject.QueryRunner(query);
		return rs;
	}
	public void updateWardenPassword(String WardenId, String newPassword) throws
Exception
	{
		String query = "UPDATE warden SET WardenPassword = '"+newPassword+"'
WHERE WardenId = '"+WardenId+"'";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public void updateWardenHostelId(String WardenId , String HostelId) throws
Exception
	{
		String query = "UPDATE warden SET HostelId = '"+HostelId+"' WHERE
WardenId = '"+WardenId+"'";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public void deleteWarden(String WardenId) throws Exception
	{
		String query = "DELETE from warden where WardenId = '"+ WardenId +"'";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public void checkId(String id) throws Exception
	{
		ResultSet rs = getWardenById(id);
		rs.next();
		String getid = rs.getString(1);
	}

	public String getPassword(String id) throws Exception
	{
		ResultSet rs = getWardenById(id);
		rs.next();
		String password = rs.getString(4);
		return password;
	}
}
```

5.) MentorQueryHandler Class

```
package se.dbconnect;

import java.sql.ResultSet;
import se.mentor.MentorData;

public class MentorQueryHandler
{
	private DatabaseConnection DatabaseConnectionobject;
	public MentorQueryHandler()
	{
		DatabaseConnectionobject = new DatabaseConnection();
	}
	public void addMentor(MentorData obj , String password) throws Exception
	{
		String query = "insert into mentor values('" + obj.EmployeeId + "' , ' "
+ obj.name + "' , '" + obj.Emailid + "' , '" + password + "')";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public ResultSet getMentorById(String MentorID) throws Exception
	{
		MentorID = MentorID.toUpperCase();
		String query = "select * from mentor where MentorId = '"+MentorID+"'";
		System.out.println(query);
		ResultSet rs=DatabaseConnectionobject.QueryRunner(query);
		return rs;
	}
	public void updateMentorPassword(String MentorId, String newPassword) throws 
Exception
	{
		String query = "UPDATE mentor SET MentorPassword = '"+newPassword+"' 
WHERE MentorId = '"+MentorId+"'";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public void deleteMentor(String MentorId) throws Exception
	{
		String query = "DELETE from mentor where MentorId = '"+ MentorId +"'";
		System.out.println(query);
		DatabaseConnectionobject.QueryRunner(query);
	}
	public void checkId(String id) throws Exception
	{
		ResultSet rs = getMentorById(id);
		rs.next();
		String getid = rs.getString(1);
	}

	public String getPassword(String id) throws Exception
	{
		ResultSet rs = getMentorById(id);
		rs.next();
		String password = rs.getString(4);
		return password;
	}
}
```

Student Module:

```
package se.Student;

import se.dbconnect.*;
import java.sql.ResultSet;
import java.util.ArrayList;

import se.mentor.*;
import se.warden.*;

public class Student
{
	StudentQueryHandler obj;
	Mentor mentorobj;
	Warden wardenobj;
	public Student()
	{
	   obj = new StudentQueryHandler();
	   mentorobj = new Mentor();
	   wardenobj = new Warden();
	}
	public StudentData getStudentById(String id)
	{
		if(!checkId(id))
			return null;

		try
		{
			ResultSet rs = obj.getStudentById(id);
			rs.next();
			StudentData data = new StudentData();
			data.Id = rs.getString(1);
			data.Name = rs.getString(2);
			data.gender = rs.getString(3);
			data.Email = rs.getString(4);
			data.status = rs.getInt(6);
			data.late = rs.getInt(7);
			data.RoomNumber = rs.getInt(8);
			data.MentorDetails = mentorobj.getMentorById(rs.getString(9));
			data.HostelName = rs.getString(10);
			data.WardenData = wardenobj.getWardenByHostelId(data.HostelName);
			return data;
		}
		catch(Exception e)
		{
			return null;
		}
	}
	public ArrayList<StudentData> getStudentsbyHostel(String HostelId)
	{
		try
		{
			ResultSet rs = obj.getStudentbyHostel(HostelId);
			ArrayList<StudentData> data = new ArrayList<>();
			while(rs.next())
			{
				StudentData Sdata = new StudentData();
				Sdata.Id = rs.getString(1);
				Sdata.Name = rs.getString(2);
				Sdata.gender = rs.getString(3);
				Sdata.Email = rs.getString(4);
				Sdata.status = rs.getInt(6);
				Sdata.late = rs.getInt(7);
				Sdata.RoomNumber = rs.getInt(8);
				Sdata.MentorDetails = 
mentorobj.getMentorById(rs.getString(9));
				Sdata.HostelName = rs.getString(10);
				Sdata.WardenData = 
wardenobj.getWardenByHostelId(Sdata.HostelName);
				data.add(Sdata);
			}
			return data;
		}
		catch(Exception e)
		{
			return null;
		}
	}
	public ArrayList<StudentData> getStudentbyMentor(String MentorId)
	{
		try
		{
			ResultSet rs = obj.getStudentbyMentor(MentorId);
			ArrayList<StudentData> data = new ArrayList<>();
			while(rs.next())
			{
				StudentData Sdata = new StudentData();
				Sdata.Id = rs.getString(1);
				Sdata.Name = rs.getString(2);
				Sdata.gender = rs.getString(3);
				Sdata.Email = rs.getString(4);
				Sdata.status = rs.getInt(6);
				Sdata.late = rs.getInt(7);
				Sdata.RoomNumber = rs.getInt(8);
				Sdata.MentorDetails = 
mentorobj.getMentorById(rs.getString(9));
				Sdata.HostelName = rs.getString(10);
				Sdata.WardenData = 
wardenobj.getWardenByHostelId(Sdata.HostelName);
```

```
                        data.add(Sdata);
                    }
                    return data;
                }
                catch(Exception e)
                {
                    return null;
                }
            }
          public boolean addStudent(StudentData sobj,String password)
            {
                try
                {
                    obj.addStudent(sobj,password);
                    return true;
                }
                catch(Exception e)
                {
                    return false;
                }
            }
            public boolean updatePassword(String id , String newPassword)
            {
                if(!checkId(id))
                    return false;

                try
                {
                    obj.updatePassword(id, newPassword);
                    return true;
                }
                catch(Exception e)
                {
                    return false;
                }
            }
            public boolean checkId(String id)
            {

                try
                {
                    obj.checkId(id);
                    return true;
                }
                catch(Exception e)
                {
                    return false;
                }
            }
            public String getPassword(String id)
            {

                try
                {
                    if(checkId(id))
```

```
			{
				String password = obj.getPassword(id);
				return password;
			}
			else
				return null;
		}
		catch(Exception e)
		{
			return null;
		}
	}
	public boolean updateMentor(String StudentId , String newMentor)
	{
		if(!checkId(StudentId))
			return false;
		try
		{
			obj.updateMentor(StudentId, newMentor);
			return true;
		}
		catch(Exception e)
		{
			return false;
		}
	}
	public boolean updateHostel(String StudentId , String HostelId)
	{
		if(!checkId(StudentId))
			return false;
		try
		{
			obj.updateHostel(StudentId, HostelId);
			return true;
		}
		catch(Exception e)
		{
			return false;
		}
	}
	public boolean updatetimeslate(String StudentId)
	{
		if(!checkId(StudentId))
			return false;
		try
		{
			obj.updatetimeslate(StudentId);;
			return true;
		}
		catch(Exception e)
		{
			return false;
		}
	}
}
```

```
package se.Student;

import se.mentor.MentorData;
import se.warden.WardenData;

public class StudentData
{
	public String Name;
	public String Id;
	public String Email;
	public String gender;
	public String HostelName;
	public int RoomNumber;
	public int status;
	public int late;
	public MentorData MentorDetails;
	public WardenData WardenData;
}
```

Warden Module:

```
package se.warden;

import se.dbconnect.*;
import java.sql.ResultSet;

public class Warden
{
	WardenQueryHandler obj;
	public Warden()
	{
		obj = new WardenQueryHandler();
	}
	public boolean addWarden(WardenData wobj , String password)
	{
		try
		{
			obj.addWarden(wobj,password);
			return true;
		}
		catch(Exception e)
		{
			return false;
		}
```

```
}
public WardenData getWardenById(String WardenID)
{
	try
	{
		ResultSet rs = obj.getWardenById(WardenID);
		rs.next();
		WardenData data = new WardenData();
		data.EmployeeId = rs.getString(1);
		data.name = rs.getString(2);
		data.Emailid = rs.getString(3);
		data.HostelId = rs.getString(4);
		return data;
	}
	catch(Exception e)
	{
		return null;
	}
}
public WardenData getWardenByHostelId(String HostelId)
{
	try
	{
		ResultSet rs = obj.getWardenByHostelId(HostelId);
		rs.next();
		WardenData data = new WardenData();
		data.EmployeeId = rs.getString(1);
		data.name = rs.getString(2);
		data.Emailid = rs.getString(3);
		data.HostelId = rs.getString(4);
		return data;
	}
	catch(Exception e)
	{
		return null;
	}
}
public boolean updateWardenPassword(String WardenId, String newPassword)
{
	try
	{
		obj.updateWardenPassword(WardenId, newPassword);
		return true;
	}
	catch(Exception e)
	{
		return false;
	}
}
public boolean updateWardenHostelId(String WardenId , String HostelId)
{
	try
	{
		obj.updateWardenHostelId(WardenId, HostelId);
		return true;
```

```
				}
				catch(Exception e)
				{
					return false;
				}
		}
		public boolean deleteMentor(String WardenId)
		{
				try
				{
					obj.deleteWarden(WardenId);
					return true;
				}
				catch(Exception e)
				{
					return false;
				}
		}
		public boolean checkId(String id)
		{

				try
				{
					obj.checkId(id);
					return true;
				}
				catch(Exception e)
				{
					return false;
				}
		}
		public String getPassword(String id)
		{

				try
				{
					if(checkId(id))
					{
						String password = obj.getPassword(id);
						return password;
					}
					else
						return null;
				}
				catch(Exception e)
				{
					return null;
				}
		}

}
package se.warden;

public class WardenData
{
	public String name;
	public String EmployeeId;
	public String Emailid;
	public String HostelId;
}
```

Mentor Module:

```
package se.mentor;

import se.dbconnect.*;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Mentor
{
	MentorQueryHandler obj;
	public Mentor()
	{
		obj = new MentorQueryHandler();
	}
	public boolean addMentor(MentorData mobj , String password)
	{
		try
		{
			obj.addMentor(mobj,password);
			return true;
		}
		catch(Exception e)
		{
			return false;
		}
	}
	public MentorData getMentorById(String MentorID)
	{
		try
		{
			ResultSet rs = obj.getMentorById(MentorID);
			rs.next();
			MentorData data = new MentorData();
			data.EmployeeId = rs.getString(1);
			data.name = rs.getString(2);
			data.Emailid = rs.getString(3);
			return data;
		}
		catch(Exception e)
		{
			return null;
		}
	}
	public boolean updateMentorPassword(String MentorId, String newPassword)
	{
		try
		{
			obj.updateMentorPassword(MentorId, newPassword);
```

```
				return true;
			}
			catch(Exception e)
			{
				return false;
			}
		}
		public boolean deleteMentor(String MentorId)
		{
			try
			{
				obj.deleteMentor(MentorId);
			}
			catch(Exception e)
			{
				return false;
			}
			return true;
		}
		public boolean checkId(String id)
		{

			try
			{
				obj.checkId(id);
				return true;
			}
			catch(Exception e)
			{
				return false;
			}
		}
		public String getPassword(String id)
		{

			try
			{
				if(checkId(id))
				{
					String password = obj.getPassword(id);
					return password;
				}
				else
					return null;
			}
			catch(Exception e)
			{
				return null;
			}
		}
}
package se.mentor;

public class MentorData
{
	public String name;
	public String EmployeeId;
	public String Emailid;
}
```

Request Module:

```
package se.request;

import java.sql.ResultSet;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import se.Student.Student;
import se.Student.StudentData;
import se.dbconnect.RequestQueryHandler;
import se.mentor.Mentor;
import se.warden.Warden;

public class Request
{
	RequestQueryHandler Robj;
	Mentor mentorobj;
	Warden wardenobj;
	Student Sobj;
	Request()
	{
		mentorobj = new Mentor();
		wardenobj = new Warden();
		Sobj = new Student();
		Robj = new RequestQueryHandler();
	}
	public RequestData getRequestById(String id)
	{
		if(!checkId(id))
			return null;

		try
		{
			SimpleDateFormat dateFormat = new SimpleDateFormat("dd-mm-yyyy
hh:mm:ss");
			ResultSet rs = Robj.getRequestById(id);
			rs.next();
			RequestData data = new RequestData();
			data.RequestId = rs.getString(1);
			data.StudentId = rs.getString(2);
			data.Destination  = rs.getString(3);
			data.fromDate  = rs.getDate(4);
			data.toDate  = rs.getDate(4);
			data.intime  = dateFormat.parse(rs.getString(6));
			data.outtime = dateFormat.parse(rs.getString(7));
			data.internalComment = rs.getString(8);
			data.Status = rs.getInt(9);
```

```
                data.RequestType = rs.getInt(10);

                return data;
            }
            catch(Exception e)
            {
                return null;
            }
        }
        public  ArrayList<RequestData> getRequestbyHostel(String HostelId)
        {
            try
            {
                ResultSet rs = Robj.getRequestbyHostel(HostelId);
                ArrayList<RequestData> data = new ArrayList<>();
                SimpleDateFormat dateFormat = new SimpleDateFormat("dd-mm-yyyy
    hh:mm:ss");
                while(rs.next())
                {
                    RequestData Rdata = new RequestData();
                    Rdata.RequestId = rs.getString(1);
                    Rdata.StudentId = rs.getString(2);
                    Rdata.Destination  = rs.getString(3);
                    Rdata.fromDate  = rs.getDate(4);
                    Rdata.toDate  = rs.getDate(4);
                    Rdata.intime  = dateFormat.parse(rs.getString(6));
                    Rdata.outtime = dateFormat.parse(rs.getString(7));
                    Rdata.internalComment = rs.getString(8);
                    Rdata.Status = rs.getInt(9);
                    Rdata.RequestType = rs.getInt(10);
                }
                return data;
            }
            catch(Exception e)
            {
                return null;
            }
        }
        public  ArrayList<RequestData> getRequestbyMentor(String MentorId)
        {
            if(!mentorobj.checkId(MentorId))
                return null;

            try
            {
                ResultSet rs = Robj.getRequestbyMentor(MentorId);
                ArrayList<RequestData> data = new ArrayList<>();
                SimpleDateFormat dateFormat = new SimpleDateFormat("dd-mm-yyyy
    hh:mm:ss");
                while(rs.next())
                {
                    RequestData Rdata = new RequestData();
                    Rdata.RequestId = rs.getString(1);
                    Rdata.StudentId = rs.getString(2);
                    Rdata.Destination  = rs.getString(3);
```

```
                        Rdata.fromDate  = rs.getDate(4);
                        Rdata.toDate  = rs.getDate(4);
                        Rdata.intime  = dateFormat.parse(rs.getString(6));
                        Rdata.outtime = dateFormat.parse(rs.getString(7));
                        Rdata.internalComment = rs.getString(8);
                        Rdata.Status = rs.getInt(9);
                        Rdata.RequestType = rs.getInt(10);
                    }
                    return data;
                }
                catch(Exception e)
                {
                    return null;
                }
            }
            public  ArrayList<RequestData> getRequestbyStudent(String StudentId)
            {
                if(!Sobj.checkId(StudentId))
                    return null;

                try
                {
                    ResultSet rs = Robj.getRequestbyStudent(StudentId);
                    ArrayList<RequestData> data = new ArrayList<>();
                    SimpleDateFormat dateFormat = new SimpleDateFormat("dd-mm-yyyy
hh:mm:ss");
                    while(rs.next())
                    {
                        RequestData Rdata = new RequestData();
                        Rdata.RequestId = rs.getString(1);
                        Rdata.StudentId = rs.getString(2);
                        Rdata.Destination  = rs.getString(3);
                        Rdata.fromDate  = rs.getDate(4);
                        Rdata.toDate  = rs.getDate(4);
                        Rdata.intime  = dateFormat.parse(rs.getString(6));
                        Rdata.outtime = dateFormat.parse(rs.getString(7));
                        Rdata.internalComment = rs.getString(8);
                        Rdata.Status = rs.getInt(9);
                        Rdata.RequestType = rs.getInt(10);
                    }
                    return data;
                }
                catch(Exception e)
                {
                    return null;
                }
            }
            public  ArrayList<RequestData> getAllApprovedRequest()
            {
                try
                {
                    ResultSet rs = Robj.getAllApprovedRequest();
                    ArrayList<RequestData> data = new ArrayList<>();
                    SimpleDateFormat dateFormat = new SimpleDateFormat("dd-mm-yyyy
hh:mm:ss");
```

```
                        while(rs.next())
                        {
                                RequestData Rdata = new RequestData();
                                Rdata.RequestId = rs.getString(1);
                                Rdata.StudentId = rs.getString(2);
                                Rdata.Destination  = rs.getString(3);
                                Rdata.fromDate  = rs.getDate(4);
                                Rdata.toDate  = rs.getDate(4);
                                Rdata.intime  = dateFormat.parse(rs.getString(6));
                                Rdata.outtime = dateFormat.parse(rs.getString(7));
                                Rdata.internalComment = rs.getString(8);
                                Rdata.Status = rs.getInt(9);
                                Rdata.RequestType = rs.getInt(10);
                        }
                        return data;
                }
                catch(Exception e)
                {
                        return null;
                }
        }
        public  boolean addRequest(RequestData obj)
        {
                try
                {
                        Robj.addRequest(obj);
                        return true;
                }
                catch(Exception e)
                {
                        return false;
                }
        }
        public  boolean updateRequestbyMentor(String RequestId , int choice)
        {
                if(!checkId(RequestId))
                        return false;

                try
                {
                        Robj.updateRequestbyMentor(RequestId, choice);
                        return true;
                }
                catch(Exception e)
                {
                        return false;
                }
        }
        public  boolean updateRequestbyWarden(String RequestId, int choice, String
comment)
        {
                if(!checkId(RequestId))
                        return false;

                try
                {
                        Robj.updateRequestbyWarden(RequestId, choice,comment);
                        return true;
                }
                catch(Exception e)
                {
                        return false;
                }
        }
        public  boolean deleteRequest(String RequestId)
        {
                if(!checkId(RequestId))
                        return false;

                try
                {
                        Robj.deleteRequest(RequestId);
                        return true;
                }
                catch(Exception e)
                {
                        return false;
                }
        }
        public  boolean updateIntime(String RequestId , Date time)
        {
                if(!checkId(RequestId))
                        return false;

                try
                {
                        Robj.updateIntime(RequestId, time);
                        return true;
                }
                catch(Exception e)
                {
                        return false;
                }
        }
        public  boolean updateOuttime(String RequestId, Date time)
        {
                if(!checkId(RequestId))
                        return false;

                try
                {
                        Robj.updateIntime(RequestId, time);
                        return true;
                }
                catch(Exception e)
                {
                        return false;
                }
        }
        public  boolean checkIfStudentLate(RequestData obj)
```

```
{
        if(obj.RequestType == 1)
        {
                if(obj.intime.~~getHours()~~ > 20)
                        return true;
        }
        else
        {
                if(obj.intime.~~getDate()~~ > obj.toDate.~~getDate()~~)
                        return true;
        }

        return false;
}
public String getRequestId(String StudentId)
{
        Date date = Calendar.getInstance().getTime();
 DateFormat dateFormat = new SimpleDateFormat("yyyy-mm-dd hh:mm:ss");
 String strDate = dateFormat.format(date);
 strDate=strDate.replaceAll("-","");
 strDate=strDate.replaceAll(":","");
 strDate=strDate.replaceAll(" ","");
        String id = StudentId + strDate;
        return id;
}
public boolean checkId(String id)
{
        try
        {
                Robj.checkId(id);
                return true;
        }
        catch(Exception e)
        {
                return false;
        }
}
public ArrayList<StudentData> StudentsOut(String HostelId)
{
        try
        {
                ArrayList<RequestData> RDobj = getRequestbyHostel(HostelId);
                ArrayList<StudentData> SDobj = new ArrayList<>();
                for(RequestData x : RDobj)
                {
                        if(x.intime == null && x.outtime != null)
                                SDobj.add(Sobj.getStudentById(x.StudentId));
                }
                return SDobj;
        }
        catch(Exception e)
        {
                return null;
        }
}
```

```
}
package se.request;

import java.util.Date;

public class RequestData
{
	public Date fromDate;
	public Date toDate;
	public String Destination;
	public int RequestType;
	public String RequestId;
	public String internalComment;
	public String StudentId;
	public int Status;
	public Date intime;
	public Date outtime;
}
```

Admin Portal:

```
package se.menu;

import java.awt.CardLayout;
import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.regex.Pattern;
import se.Student.*;
import se.mentor.Mentor;
import se.mentor.MentorData;
import se.warden.Warden;
import se.warden.WardenData;
import javax.swing.*;

public class Admin extends JFrame
{
	JPanel sidepanel;
	JPanel mainpanel;
	JPanel addmentor;

	Admin(JFrame prev)
	{
		setSize(1030,550);
		addSidePanel();
		addMainPanel();
		setLayout(null);
		setResizable(false);
		setVisible(true);
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		prev.dispose();
	}

	private void addSidePanel()
	{
		JPanel sidepanel = new JPanel();
		JLabel menuname = new JLabel("Admin Portal");
		menuname.setForeground(Color.white);
		menuname.setBounds(50,0,200,50);

		JButton RRequest = new JButton("Add Mentor");
		RRequest.setBounds(0,100,200,30);
		RRequest.setBackground(Color.WHITE);
		RRequest.setForeground(Color.BLACK);
		RRequest.addActionListener(new MenuButton());

		JButton RReques = new JButton("Add Warden");
		RReques.setBounds(0,150,200,30);
		RReques.setBackground(Color.WHITE);
		RReques.setForeground(Color.BLACK);
		RReques.addActionListener(new MenuButton());

		JButton StatusMenu = new JButton("Delete");
		StatusMenu.setBounds(0,200,200,30);
		StatusMenu.setBackground(Color.WHITE);
		StatusMenu.setForeground(Color.BLACK);
		StatusMenu.addActionListener(new MenuButton());

		JButton logout = new JButton("Logout");
		logout.setBounds(0,300,200,30);
		logout.setBackground(Color.WHITE);
		logout.setForeground(Color.BLACK);
		logout.addActionListener(new ActionListener()
				{
					public void actionPerformed(ActionEvent arg0)
					{
						int result =
JOptionPane.showConfirmDialog(sidepanel,"Sure? You want to exit?", "Swing Tester",
				                JOptionPane.YES_NO_OPTION,
				                JOptionPane.QUESTION_MESSAGE);
						if(result == JOptionPane.YES_OPTION)
						{
							JFrame del =
(JFrame)sidepanel.getParent().getParent().getParent().getParent();
							del.dispose();
				        }
					}

				});
		sidepanel.add(logout);
		sidepanel.add(RRequest);
```

```
        sidepanel.add(menuname);
        sidepanel.add(StatusMenu);
        sidepanel.add(RReques);

        sidepanel.setBounds(0, 0,200,500);
        sidepanel.setBackground(new Color(39,39,39));
        sidepanel.setLayout(null);
        this.add(sidepanel);
}
private void addMainPanel()
{
        mainpanel = new JPanel();
        mainpanel.setBounds(200,0,810,500);
        mainpanel.setBackground(Color.WHITE);
        mainpanel.setLayout(new CardLayout());
        AddMentor(mainpanel);
        AddWarden(mainpanel);
        addDelete(mainpanel);
        this.add(mainpanel);
}

private void addDelete(JPanel obj)
{
        JPanel panel = new JPanel();
        panel.setLayout(null);

        JLabel sear = new JLabel("Delete Mentor");
        sear.setFont(new Font("Tahoma", 1, 24));
        sear.setBounds(10,50,500,30);
        panel.add(sear);

        JTextField idS = new JTextField("Enter ID");
        idS.setBounds(10,100,200,30);
        panel.add(idS);

        JTextField idss = new JTextField("New Mentor");
        idss.setBounds(220,100,300,30);
        panel.add(idss);

        JButton search = new JButton("Delete");
        search.setBounds(570,100,100,30);
        search.addActionListener(
                        new ActionListener()
                        {
                                public void actionPerformed(ActionEvent arg0)
                                {
                                        String from = idS.getText();
                                        String to = idss.getText();
                                        if(from.length() == 0 || to.length() == 0)
                                        {

JOptionPane.showMessageDialog(panel,"InComplete Form");
                                                return;
                                        }
                                        Mentor mobj = new Mentor();
                                        boolean one =
Pattern.matches("[E][M]\\d*",from) && mobj.checkId(from);
                                        boolean two =
Pattern.matches("[E][M]\\d*",to) && mobj.checkId(to);
                                        if(!(one && two))
                                        {

     JOptionPane.showMessageDialog(panel,"Invalid Mentor ID");
                                             return;
                                        }

                                        Student Sobj = new Student();
                                        ArrayList<StudentData> Sdata =
Sobj.getStudentbyMentor(from);
                                        if(Sdata == null || Sdata.size() == 0)
                                        {

     JOptionPane.showMessageDialog(mainpanel,"No Mentee");
                                        }
                                        else
                                             for(StudentData x : Sdata)
                                             {
                                                  String id = x.Id;
                                                  if(!Sobj.updateMentor(id, to))
                                                  {

     JOptionPane.showMessageDialog(mainpanel,"Try Again");
                                                       return;
                                                  }
                                             }
                                        mobj.deleteMentor(from);

     JOptionPane.showMessageDialog(mainpanel,"Mentor Deleted and Swap Complete");
                                   }
                              });
          panel.add(search);



          JLabel Wsear = new JLabel("Delete Warden");
          Wsear.setFont(new Font("Tahoma", 1, 24));
          Wsear.setBounds(10,200,500,30);
          panel.add(Wsear);

          JTextField WidS = new JTextField("Enter ID");
          WidS.setBounds(10,250,200,30);
          panel.add(WidS);

          JTextField Widss = new JTextField("New Warden");
          Widss.setBounds(220,250,300,30);
          panel.add(Widss);

          JButton Wsearch = new JButton("Delete");
          Wsearch.setBounds(570,250,100,30);
          panel.add(Wsearch);
```

```
		Wsearch.addActionListener(new ActionListener()
				{
					public void actionPerformed(ActionEvent arg0)
					{
						String from = WidS.getText();
						String to = Widss.getText();
						if(from.length() == 0 || to.length() == 0)
						{
	JOptionPane.showMessageDialog(panel,"InComplete Form");
							return;
						}

						Warden mobj = new Warden();
						boolean one =
Pattern.matches("[E][H][W]\\d*",from) && mobj.checkId(from);
						boolean two =
Pattern.matches("[E][H][W]\\d*", to) && mobj.checkId(to);
						if(!(one && two))
						{
	JOptionPane.showMessageDialog(panel,"Invalid Warden ID");
							return;
						}

						WardenData oned = mobj.getWardenById(from);
						WardenData twod = mobj.getWardenById(to);
						if(!oned.HostelId.equals(twod.HostelId))
						{
	JOptionPane.showMessageDialog(panel,"New Warden of other hostel");
							return;
						}
						if(mobj.deleteMentor(from))
						{
	JOptionPane.showMessageDialog(panel,"Completed");
						}
						else
						{
	JOptionPane.showMessageDialog(panel,"Try Again");
						}
					}

				});

		obj.add(panel,"Delete");

	}
	private void AddWarden(JPanel obj)
	{
		JPanel panel = new JPanel();
		panel.setLayout(null);
```

```
            String names[] = new String[] {"Employee Id" , "Name" , "EmailID" ,
"Password" , "Hostel Id"};
            JTextField[] text = new JTextField[names.length];
            int y = 50;
            int i = 0;
            for(String x : names)
            {
                JLabel lo = new JLabel(x);
                lo.setBounds(10,y,200,30);
                text[i] = new JTextField();
                text[i].setBounds(220, y,200,30);
                panel.add(lo);
                panel.add(text[i]);
                y = y+50;
                i++;
            }

            JButton save = new JButton("Save");
            save.setBounds(320,y,100,30);
            save.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent arg0)
                {
                    Warden wobj = new Warden();
                    for(JTextField x : text)
                        if(x.getText().length() == 0)
                        {
                            JOptionPane.showMessageDialog(panel,"Fill
complete form");

                            return;
                        }
                    WardenData wdata = new WardenData();
                    wdata.EmployeeId = text[0].getText();
                    wdata.name = text[1].getText();
                    wdata.Emailid = text[2].getText();
                    String password = text[3].getText();
                    wdata.HostelId = text[4].getText();
                    boolean check = wobj.checkId(wdata.EmployeeId);
                    boolean pass = Pattern.matches("[E][H][W]\\d*" ,
wdata.EmployeeId);
                    if(check)
                    {
                        JOptionPane.showMessageDialog(panel,"User Already
Present");

                        return;
                    }
                    if(!pass)
                    {
                        JOptionPane.showMessageDialog(panel,"Invalid ID");
                        return;
                    }
                    if(!(wdata.HostelId.equals("MH1") ||
wdata.HostelId.equals("MH2") || wdata.HostelId.equals("LH1")))
                    {
                        JOptionPane.showMessageDialog(panel,"Invalid Hostel
ID");
```

```
                                        return;
                                }
                                if(password.length() == 0 || password.length() > 10)
                                {
                                        JOptionPane.showMessageDialog(panel,"Password length
incorrect");
                                        return;
                                }
                                if(wobj.addWarden(wdata, password))
                                {
                                        JOptionPane.showMessageDialog(panel,"Warden Added");
                                        return;
                                }
                        }

                });
                panel.add(save);
                obj.add(panel , "Add Warden");
        }
        private void AddMentor(JPanel obj)
        {
                JPanel panel = new JPanel();
                panel.setLayout(null);
                String names[] = new String[] {"Employee Id" , "Name" , "EmailID" ,
"Password"};
                JTextField[] text = new JTextField[names.length];
                int y = 50;
                int i = 0;
                for(String x : names)
                {
                        JLabel lo = new JLabel(x);
                        lo.setBounds(10,y,200,30);
                        text[i] = new JTextField();
                        text[i].setBounds(220, y,200,30);
                        panel.add(lo);
                        panel.add(text[i]);
                        y = y+50;
                        i++;
                }
                JButton save = new JButton("Save");
                save.setBounds(320,y,100,30);
                save.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0)
                        {
                                Mentor mobj = new Mentor();
                                for(JTextField x : text)
                                        if(x.getText().length() == 0)
                                        {
                                                JOptionPane.showMessageDialog(panel,"Fill
complete form");
                                                return;
                                        }
                                MentorData mdata = new MentorData();
                                mdata.EmployeeId = text[0].getText();
                                mdata.name = text[1].getText();
```

```
                                        mdata.Emailid = text[2].getText();
                                        String password = text[3].getText();
                                        boolean check = mobj.checkId(mdata.EmployeeId);
                                        boolean pass = Pattern.matches("[E][M]\\d*" ,
mdata.EmployeeId);
                                        if(check)
                                        {
                                                JOptionPane.showMessageDialog(panel,"User Already
Present");
                                                return;
                                        }
                                        if(!pass)
                                        {
                                                JOptionPane.showMessageDialog(panel,"Invalid ID");
                                                return;
                                        }
                                        if(password.length() == 0 || password.length() > 10)
                                        {
                                                JOptionPane.showMessageDialog(panel,"Password length
incorrect");
                                                return;
                                        }
                                        if(mobj.addMentor(mdata, password))
                                        {
                                                JOptionPane.showMessageDialog(panel,"Mentor Added");
                                                return;
                                        }
                                }

                        });
                        panel.add(save);
                        obj.add(panel , "Add Mentor");
        }
        public static void main(String agrs[])
        {
                new Admin(new JFrame());
        }
        class MenuButton implements ActionListener
        {
                public void actionPerformed(ActionEvent e)
                {
                        String name = e.getActionCommand();
                        CardLayout obj = (CardLayout) mainpanel.getLayout();
                        obj.show(mainpanel, name);
                }

        }
}
```

Student Portal:

```
package se.menu;

import java.awt.CardLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import se.Student.*;
import se.computation.StudentSummary;
import se.request.*;
import javax.swing.*;
import javax.swing.border.Border;

import com.toedter.calendar.JDateChooser;


public class StudentMenu extends JFrame
{
	JPanel sidepanel;
	JPanel mainpanel;
	JPanel summary;
	StudentData data;
	Student Sobj;
	Request Robj;
	JPanel requestmenu;
	int[] values;
	StudentSummary SSobj = new StudentSummary();
	StudentMenu(StudentData obj , JFrame prev)
	{
		this.data = obj;
		Sobj = new Student();
		Robj = new Request();
		setSize(1030,550);
		addSidePanel();
		addMainPanel();
		setLayout(null);
		setResizable(false);
		setVisible(true);
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		prev.dispose();
	}
	private void addSidePanel()
	{
		JPanel sidepanel = new JPanel();
		JLabel menuname = new JLabel("Student Portal");
		menuname.setForeground(Color.white);
		menuname.setBounds(50,0,200,50);
```

```
            JButton RRequest = new JButton("Raise Request");
            RRequest.setBounds(0,150,200,30);
            RRequest.setBackground(Color.WHITE);
            RRequest.setForeground(Color.BLACK);
            RRequest.addActionListener(new MenuButton());

            JButton Summary = new JButton("Summary");
            Summary.setBounds(0,200,200,30);
            Summary.setBackground(Color.WHITE);
            Summary.setForeground(Color.BLACK);
            Summary.addActionListener(new MenuButton());

            JButton StatusMenu = new JButton("Status");
            StatusMenu.setBounds(0,250,200,30);
            StatusMenu.setBackground(Color.WHITE);
            StatusMenu.setForeground(Color.BLACK);
            StatusMenu.addActionListener(new MenuButton());

            JButton Profile = new JButton("Profile");
            Profile.setBounds(0,300,200,30);
            Profile.setBackground(Color.WHITE);
            Profile.setForeground(Color.BLACK);
            Profile.addActionListener(new MenuButton());

            JButton logout = new JButton("Logout");
            logout.setBounds(0,350,200,30);
            logout.setBackground(Color.WHITE);
            logout.setForeground(Color.BLACK);
            logout.addActionListener(new ActionListener()
                         {
                                public void actionPerformed(ActionEvent arg0)
                                {
                                       int result =
JOptionPane.showConfirmDialog(sidepanel,"Sure? You want to exit?", "Swing Tester",
                                               JOptionPane.YES_NO_OPTION,
                                               JOptionPane.QUESTION_MESSAGE);
                                       if(result == JOptionPane.YES_OPTION)
                                       {
                                              JFrame del =
(JFrame)sidepanel.getParent().getParent().getParent().getParent();
                                              del.dispose();
                                    }
                                }

                         });
            sidepanel.add(logout);
            sidepanel.add(RRequest);
            sidepanel.add(Summary);
            sidepanel.add(menuname);
            sidepanel.add(StatusMenu);
            sidepanel.add(Profile);

            sidepanel.setBounds(0, 0,200,500);
            sidepanel.setBackground(new Color(39,39,39));
```

```
        sidepanel.setLayout(null);
        this.add(sidepanel);
}
private void addMainPanel()
{
        mainpanel = new JPanel();
        mainpanel.setBounds(200,0,810,500);
        mainpanel.setBackground(Color.WHITE);
        mainpanel.setLayout(new CardLayout());
        addRRMenu(mainpanel);
        addSummary(mainpanel);
        addStatus(mainpanel);
        addProfile(mainpanel);
        this.add(mainpanel);
}
private void addRRMenu(JPanel obj)
{
        JPanel panel = new JPanel();
        panel.setLayout(null);
        JLabel heading = new JLabel("Raise Request Menu");
        heading.setBounds(405,10,810,30);
        panel.add(heading);
        JLabel destination = new JLabel("Destination");
        destination.setBounds(10,75,200,30);
        panel.add(destination);

        JTextField desvalue = new JTextField();
        desvalue.setColumns(10);
        desvalue.setBounds(220, 75,200,20);
        panel.add(desvalue);

        JLabel type = new JLabel("Request Type");
        type.setBounds(10,110,200,30);
        panel.add(type);

        String request[] = new String[]{"Outing"  , "Vacation" , "Medical"};
        JComboBox cb=new JComboBox(request);
     cb.setBounds(220,110,90,20);
     panel.add(cb);

     JLabel fdate = new JLabel("From Date");
     fdate.setBounds(10,150,200,30);
     panel.add(fdate);

     JDateChooser fchooser = new JDateChooser();
     fchooser.setBounds(220,150,200,30);
     //fchooser.setDateFormatString("dd-mm-yyyy");
     panel.add(fchooser);

     JLabel tdate = new JLabel("To Date");
     tdate.setBounds(10,190,200,30);
     panel.add(tdate);

     JDateChooser tchooser = new JDateChooser();
     tchooser.setBounds(220,190,200,30);
```

```
            //tchooser.setDateFormatString("dd-mm-yyyy");
            panel.add(tchooser);

            JButton raise = new JButton("Raise Request");
            raise.setBounds(10,250,200,30);
            raise.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e)
              {
                        String destination = desvalue.getText();
                        int type = cb.getSelectedIndex()+1;
                        Date fromdate = fchooser.getDate();
                        Date todate = tchooser.getDate();

                        if(destination.length() == 0 || type == -1 || fromdate == null ||
todate == null)
                        {
                                  JOptionPane.showMessageDialog(panel,"Fill complete form");
                                  return;
                        }

                        if(destination.length() > 10)
                        {
                                  JOptionPane.showMessageDialog(panel,"Check Destination
length.");

                                  return;
                        }

                        System.out.println(new Date());
                        System.out.println(fromdate);
                        if(fromdate.compareTo(new Date()) < 0)
                        {
                                  JOptionPane.showMessageDialog(panel,"Please select proper
dates");

                                  return;
                        }

                        if(fromdate.compareTo(todate) >= 0)
                        {
                                  JOptionPane.showMessageDialog(panel,"The to-date should be
after the from-date");
                                  return;
                        }

                        RequestData Rdata = new RequestData();
                        Rdata.StudentId = data.Id;
                        Rdata.RequestId = Robj.getRequestId(data.Id);
                        Rdata.fromDate = fromdate;
                        Rdata.toDate = todate;
                        Rdata.Destination = destination;
                        Rdata.RequestType = type;
                        Rdata.Status = 1;

                        if(Robj.addRequest(Rdata))
                        {
                                  requestmenu.add(getRequest(Rdata , true));
```

```
                                requestmenu.revalidate();
                                requestmenu.repaint();
                                JOptionPane.showMessageDialog(panel,("Request Saved\nId:"
+ Rdata.RequestId));
                                desvalue.setText("");
                                fchooser.setDate(null);
                                tchooser.setDate(null);
                        }
                }
            });
            panel.add(raise);

                obj.add(panel , "Raise Request");
        }
        private void addSummary(JPanel obj)
        {
                values = SSobj.getData(data.Id);
                summary = new JPanel();
                summary.setLayout(null);
                JLabel heading = new JLabel("Summary");
                heading.setBounds(405,10,810,30);
                summary.add(heading);
                String[] head = new String[] {"Number Total Requests" , "Number of
Active Requests" ,
                                "Number of Approved Requests","Number of Rejected
Request","Time's Late"};

                int y = 100;
                int i = 0;
                for(String x : head)
                {
                        JLabel Lobj = new JLabel(x);
                        Lobj.setBounds(10,y,200,30);
                        summary.add(Lobj);
                        JLabel Lobjd;
                        if(i < 4)
                                Lobjd = new JLabel(""+values[i]);
                        else
                                Lobjd = new JLabel(""+data.late);
                        Lobjd.setBounds(220,y,50,30);
                        summary.add(Lobjd);
                        i++;
                        y +=50;
                }
                obj.add(summary , "Summary");
        }
        private void addStatus(JPanel obj)
        {
                requestmenu = new JPanel();
                JScrollPane listScroller = new JScrollPane(requestmenu);
                listScroller.setPreferredSize(new Dimension(800,550));
                requestmenu.setPreferredSize(new Dimension(800, 550));
                listScroller.setAlignmentX(RIGHT_ALIGNMENT);
                ArrayList<RequestData> rdata = Robj.getRequestbyStudent(data.Id);
                requestmenu.setLayout(new BoxLayout(requestmenu, BoxLayout.PAGE_AXIS));
            if(rdata != null)
                  for(RequestData x : rdata)
                  {
                        if(x.Status != 6 && x.Status != 3 && x.Status != 5)
                              requestmenu.add(getRequest(x , true));
                        else if(x.Status == 6 || x.Status == 3 || x.Status == 5)
                              requestmenu.add(getRequest(x , false));
                        requestmenu.add(Box.createRigidArea(new Dimension(0,5)));
                  }
            obj.add(listScroller , "Status");
      }
      private JPanel getRequest(RequestData request , boolean add)
      {
            Border blackline = BorderFactory.createLineBorder(Color.black);
            JPanel panel = new JPanel();
            panel.setLayout(null);
            panel.setBorder(blackline);
            panel.setSize(new Dimension(800,200));

            JLabel requestId = new JLabel("Request ID: " + request.RequestId);
            requestId.setBounds(5,0,250,20);
            panel.add(requestId);

            JLabel StudentId = new JLabel("Student ID: "+request.StudentId);
            StudentId.setBounds(255,0,200,20);
            panel.add(StudentId);

            String todate = new SimpleDateFormat("dd-MM-
yyyy").format(request.toDate);
            String fromdate = new SimpleDateFormat("dd-MM-
yyyy").format(request.fromDate);

            JLabel From = new JLabel("From Date: " + fromdate);
            From.setBounds(450,0,200,20);
            panel.add(From);

            JLabel To = new JLabel("To Date: "+ todate);
            To.setBounds(650,0,200,20);
            panel.add(To);

            JLabel Destination = new JLabel("Destination: " + request.Destination);
            Destination.setBounds(5,40,250,20);
            panel.add(Destination);

            String type = "";
            if(request.RequestType == 1)
                  type = "Outing";
            else if(request.RequestType == 2)
                  type = "Vacation";
            else if(request.RequestType == 3)
                  type = "Medical";
            JLabel Type = new JLabel("Type: "+ type);
            Type.setBounds(255,40,250,20);
            panel.add(Type);
```

```
        String status = "";
        int SR = request.Status;
        if(SR == 1)
                status = "At Mentor";
        else if(SR == 3 || SR == 5)
                status = "Declined";
        else if(SR == 6)
                status = "Approved";
        else if(SR == 2)
                status = "At Warden";
        else if(SR == 4)
                status = "Reconsideration";

        JLabel stat = new JLabel("Status: " + status);
        stat.setBounds(450,40,150,30);
        panel.add(stat);

        if(add)
        {
                JButton delete = new JButton("Delete");
                delete.setBounds(620,40,100,30);
                panel.add(delete);
                delete.addActionListener( new ActionListener()
                {
                        public void actionPerformed(ActionEvent e)
                        {
                                if(Robj.deleteRequest(request.RequestId))
                                {
                                        requestmenu.remove(panel);
                                        requestmenu.revalidate();
                                        requestmenu.repaint();
                                }
                        }
                });
        }
        return panel;
}
private void addProfile(JPanel obj)
{
        JPanel panel = new JPanel();
        JLabel heading = new JLabel("Profile");
        heading.setBounds(405,5,810,30);
        panel.add(heading);

        JLabel id = new JLabel("Student Id");
        id.setBounds(10,45,200,30);
        JLabel idvalue = new JLabel(data.Id);
        idvalue.setBounds(220,45,200,30);
        JLabel name = new JLabel("Name");
        name.setBounds(10,75,200, 30);
        JLabel namevalue = new JLabel(data.Name);
        namevalue.setBounds(220, 75, 200, 30);
        JLabel email = new JLabel("Email:");
        email.setBounds(10,105,200,30);
        JLabel valueemail = new JLabel(data.Email);
```

```
valueemail.setBounds(220,105,200,30);
JLabel room = new JLabel("Room No.:");
room.setBounds(10,135, 200, 30);
JLabel roomvalue = new JLabel(""+data.RoomNumber);
roomvalue.setBounds(220,135,200,30);

JLabel head = new JLabel("Mentor details");
head.setBounds(405,175,200,30);
JLabel Mname = new JLabel("Name");
Mname.setBounds(10,205,200, 30);
JLabel Mnamevalue = new JLabel(data.MentorDetails.name);
Mnamevalue.setBounds(220,205, 200, 30);
JLabel Memail = new JLabel("Email:");
Memail.setBounds(10,235,200,30);
JLabel Mvalueemail = new JLabel(data.MentorDetails.Emailid);
Mvalueemail.setBounds(220,235,200,30);

JLabel head2 = new JLabel("Warden details");
head2.setBounds(405,275,200,30);
panel.add(head2);
JLabel Wname = new JLabel("Name");
Wname.setBounds(10,305,200, 30);
panel.add(Wname);
JLabel Wnamevalue = new JLabel(data.WardenData.name);
Wnamevalue.setBounds(220,305, 200, 30);
panel.add(Wnamevalue);
JLabel Wemail = new JLabel("Email:");
Wemail.setBounds(10,335,200,30);
panel.add(Wemail);
JLabel Wvalueemail = new JLabel(data.WardenData.Emailid);
Wvalueemail.setBounds(220,335,200,30);
panel.add(Wvalueemail);
JLabel hostel = new JLabel("Hostel ID");
hostel.setBounds(10,365, 200, 30);
panel.add(hostel);
JLabel hostelvalue = new JLabel(data.WardenData.HostelId);
hostelvalue.setBounds(220,365, 200, 30);
panel.add(hostelvalue);

panel.setLayout(null);
panel.add(id);
panel.add(idvalue);
panel.add(name);
panel.add(namevalue);
panel.add(email);
panel.add(valueemail);
panel.add(room);
panel.add(roomvalue);
panel.add(head);
panel.add(Mname);
panel.add(Mnamevalue);
panel.add(Memail);
panel.add(Memail);
panel.add(Mvalueemail);
```

```
            JLabel uppass = new JLabel("Update Password");
            uppass.setBounds(10, 445 ,200,30);
            panel.add(uppass);

            JTextField newpass = new JTextField("New Pasword");
            newpass.setBounds(220,450,100,20);
            panel.add(newpass);

            JButton update = new JButton("Update");
            update.setBounds(350,450,100,20);
            update.addActionListener(new ActionListener() {

                    public void actionPerformed(ActionEvent e)
                    {
                            String newpassword = newpass.getText();
                            if(newpassword.length() == 0 || newpassword.length() > 10)
                            {
                                    JOptionPane.showMessageDialog(panel,"Enter correct
password");
                                    return;
                            }

                            if(Sobj.updatePassword(data.Id, newpassword))
                            {
                                    JOptionPane.showMessageDialog(panel,"Password
Updated");
                                    newpass.setText("");
                            }
                    }

            });
            panel.add(update);

            obj.add(panel , "Profile");
    }
    public static void main(String agrs[])
    {
            Student obj = new Student();
            new StudentMenu(obj.getStudentById("21BEC2929") , new JFrame());
    }
    class MenuButton implements ActionListener
    {
            public void actionPerformed(ActionEvent e)
            {
                    String name = e.getActionCommand();
                    CardLayout obj = (CardLayout) mainpanel.getLayout();
                    obj.show(mainpanel, name);
            }

    }
}
```

Mentor Portal:

```
package se.menu;

import java.awt.CardLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.regex.Pattern;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextField;
import javax.swing.ScrollPaneLayout;
import javax.swing.border.Border;

import se.Student.Student;
import se.Student.StudentData;
import se.computation.MS;
import se.computation.ProfileGenerator;
import se.mentor.Mentor;
import se.mentor.MentorData;
import se.request.Request;
import se.request.RequestData;

public class MentorMenu extends JFrame
{
	JPanel sidepanel;
	JPanel mainpanel;
	JPanel requestmenu;
	JPanel summarymenu;
	JPanel profile;
	MentorData data;
	Mentor Mobj;
	Request Robj;

	MentorMenu(MentorData obj , JFrame prev)
	{
		this.data = obj;
		Mobj = new Mentor();
		Robj = new Request();
		setSize(1030,550);
		addSidePanel();
		addMainPanel();
		setLayout(null);
		setVisible(true);
		setResizable(false);
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		prev.dispose();
	}

	private void addSidePanel()
	{
		JPanel sidepanel = new JPanel();
		JLabel menuname = new JLabel("Mentor Portal");
		menuname.setForeground(Color.white);
		menuname.setBounds(50,0,200,50);

		JButton RRequest = new JButton("Requests");
		RRequest.setBounds(0,150,200,30);
		RRequest.setBackground(Color.WHITE);
		RRequest.setForeground(Color.BLACK);
		RRequest.addActionListener(new MenuButton());

		JButton Summary = new JButton("Summary");
		Summary.setBounds(0,200,200,30);
		Summary.setBackground(Color.WHITE);
		Summary.setForeground(Color.BLACK);
		Summary.addActionListener(new MenuButton());

		JButton Profile = new JButton("Profile");
		Profile.setBounds(0,250,200,30);
		Profile.setBackground(Color.WHITE);
		Profile.setForeground(Color.BLACK);
		Profile.addActionListener(new MenuButton());

		JButton logout = new JButton("Logout");
		logout.setBounds(0,300,200,30);
		logout.setBackground(Color.WHITE);
		logout.setForeground(Color.BLACK);
		logout.addActionListener(new ActionListener()
				{
					public void actionPerformed(ActionEvent arg0)
					{
						int result =
JOptionPane.showConfirmDialog(sidepanel,"Sure? You want to exit?", "Swing Tester",
							JOptionPane.YES_NO_OPTION,
							JOptionPane.QUESTION_MESSAGE);
						if(result == JOptionPane.YES_OPTION)
						{
							JFrame del =
(JFrame)sidepanel.getParent().getParent().getParent().getParent();
							del.dispose();
						}
					}
```

```
                });
        sidepanel.add(logout);
        sidepanel.add(RRequest);
        sidepanel.add(Summary);
        sidepanel.add(menuname);
        sidepanel.add(Profile);

        sidepanel.setBounds(0, 0,200,500);
        sidepanel.setBackground(new Color(39,39,39));
        sidepanel.setLayout(null);
        this.add(sidepanel);
}
private void addMainPanel()
{
        mainpanel = new JPanel();
        mainpanel.setBounds(200,0,810,500);
        mainpanel.setBackground(Color.WHITE);
        mainpanel.setLayout(new CardLayout());
        addRMenu(mainpanel);
        addSummary(mainpanel);
        addProfile(mainpanel);
        this.add(mainpanel);
}
private void addRMenu(JPanel obj)
{
        requestmenu = new JPanel();
        JScrollPane listScroller = new JScrollPane(requestmenu);
        listScroller.setPreferredSize(new Dimension(800,550));
        requestmenu.setPreferredSize(new Dimension(800, 550));
        listScroller.setAlignmentX(RIGHT_ALIGNMENT);
        ArrayList<RequestData> rdata = Robj.getRequestbyMentor(data.EmployeeId);
        requestmenu.setLayout(new BoxLayout(requestmenu, BoxLayout.PAGE_AXIS));
        if(rdata != null)
                for(RequestData x : rdata)
                {
                        if(x.Status == 1)
                                requestmenu.add(getRequest(x , false));
                        else if(x.Status == 4)
                                requestmenu.add(getRequest(x , true));
                        requestmenu.add(Box.createRigidArea(new Dimension(0,5)));
                }
        obj.add(listScroller , "Requests");
}
private JPanel getRequest(RequestData request , boolean add)
{
        Border blackline = BorderFactory.createLineBorder(Color.black);
        JPanel panel = new JPanel();
        panel.setLayout(null);
        panel.setBorder(blackline);
        panel.setSize(new Dimension(800,200));

        JLabel requestId = new JLabel("Request ID: " + request.RequestId);
        requestId.setBounds(5,0,250,20);
        panel.add(requestId);
```

```
            JLabel StudentId = new JLabel("Student ID: "+request.StudentId);
            StudentId.setBounds(255,0,200,20);
            panel.add(StudentId);

            JLabel From = new JLabel("From Date: " + request.fromDate);
            From.setBounds(450,0,200,20);
            panel.add(From);

            JLabel To = new JLabel("To Date: "+ request.toDate);
            To.setBounds(650,0,200,20);
            panel.add(To);

            JLabel Destination = new JLabel("Destination: " + request.Destination);
            Destination.setBounds(5,40,250,20);
            panel.add(Destination);

            String type = "";
            if(request.RequestType == 1)
                   type = "Outing";
            else if(request.RequestType == 2)
                   type = "Vacation";
            else
                   type = "Medical";
            JLabel Type = new JLabel("Type: "+ type);
            Type.setBounds(255,40,250,20);
            panel.add(Type);

            JButton accept = new JButton("Accept");
            accept.setBounds(450,40,100,30);
            panel.add(accept);
            accept.addActionListener( new ActionListener()
            {
                   public void actionPerformed(ActionEvent e)
                   {
                          if(request.Status == 4 ?
    Robj.updateRequestbyMentor(request.RequestId, 6) :
    Robj.updateRequestbyMentor(request.RequestId, 2))
                          {
                                 requestmenu.remove(panel);
                                 requestmenu.revalidate();
                                 requestmenu.repaint();
                          }
                   }

            });

            JButton decline = new JButton("Decline");
            decline.setBounds(550,40,100,30);
            panel.add(decline);
            decline.addActionListener(new ActionListener()
                          {
                                 public void actionPerformed(ActionEvent e)
                                 {

       if(Robj.updateRequestbyMentor(request.RequestId, 3))
```

```
                                        {
                                                requestmenu.remove(panel);
                                                requestmenu.revalidate();
                                                requestmenu.repaint();
                                        }
                                }
                        });


                if(add)
                {
                        JButton view = new JButton("View Comment");
                        view.setBounds(650,40,130,30);
                        view.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e)
                                {
        JOptionPane.showMessageDialog(requestmenu,request.internalComment);
                                }
                        });
                        panel.add(view);
                }
                return panel;
        }
        private void addSummary(JPanel obj)
        {
                MS MSobj = new MS();
                int[] values = MSobj.getDataM(data.EmployeeId);

                JPanel panel = new JPanel();
                JLabel heading = new JLabel("Summary");
                heading.setBounds(405,5,810,30);
                panel.add(heading);
                JLabel one = new JLabel("Total Active requests");
                one.setBounds(10, 100,200, 30);
                JLabel oned = new JLabel(""+values[1]);
                oned.setBounds(220, 100,200, 30);
                JLabel two = new JLabel("Total Requests");
                two.setBounds(10,150,200,30);
                JLabel twod = new JLabel(""+values[0]);
                twod.setBounds(200,150,200,30);
                panel.add(one);
                panel.add(two);
                panel.add(oned);
                panel.add(twod);
                panel.setLayout(null);

                JLabel sear = new JLabel("Search for Student");
                sear.setFont(new Font("Tahoma", 1, 24));
                sear.setBounds(10,200,500,30);
                panel.add(sear);

                JTextField idS = new JTextField("Enter ID");
                idS.setBounds(10,250,300,30);
                panel.add(idS);
```

```
			JButton search = new JButton("Search");
			search.setBounds(370,250,100,30);
			panel.add(search);
			search.addActionListener(new ActionListener() {
				public void actionPerformed(ActionEvent arg0)
				{
					String id = idS.getText();
					Student delobj = new Student();
					if(id.length() == 0 || !Pattern.matches("\\d{2}[a-zA-
Z]{3}\\d{4}",id) || !delobj.checkId(id))
					{
						JOptionPane.showMessageDialog(panel,"Incorrect Id");
						idS.setText("");
						return;
					}

					new ProfileGenerator(id);
				}
			});

			JLabel vmentee = new JLabel("View Mentee List");
			vmentee.setFont(new Font("Tahoma", 1, 24));
			vmentee.setBounds(10,400,300,30);
			panel.add(vmentee);

			JButton view = new JButton("View");
			view.setBounds(370,400,100,30);
			panel.add(view);
			view.addActionListener(new MenteeList());

			obj.add(panel , "Summary");
		}
		private void addProfile(JPanel obj)
		{
			JPanel panel = new JPanel();
			panel.setLayout(null);
			JLabel heading = new JLabel("Profile");
			heading.setBounds(405,5,810,30);
			panel.add(heading);
			JLabel id = new JLabel("Employee Id");
			id.setBounds(10,100,200,30);
			JLabel idvalue = new JLabel(data.EmployeeId);
			idvalue.setBounds(220,100,200,30);
			JLabel name = new JLabel("Name");
			name.setBounds(10,150,200, 30);
			JLabel namevalue = new JLabel(data.name);
			namevalue.setBounds(220, 150, 200, 30);
			JLabel email = new JLabel("Email:");
			email.setBounds(10,200,200,30);
			JLabel valueemail = new JLabel(data.Emailid);
			valueemail.setBounds(220,200,200,30);

			panel.add(id);
			panel.add(idvalue);
```

```
        panel.add(name);
        panel.add(namevalue);
        panel.add(email);
        panel.add(valueemail);

        JLabel uppass = new JLabel("Update Password");
        uppass.setBounds(10, 345 ,200,30);
        panel.add(uppass);

        JTextField newpass = new JTextField("New Pasword");
        newpass.setBounds(220,350,100,20);
        panel.add(newpass);

        JButton update = new JButton("Update");
        update.setBounds(350,350,100,20);
        update.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e)
            {
                String newpassword = newpass.getText();
                if(newpassword.length() == 0 || newpassword.length() > 10)
                {
                    JOptionPane.showMessageDialog(panel,"Enter correct
password");
                    return;
                }

                if(Mobj.updateMentorPassword(data.EmployeeId,newpassword))
                {
                    JOptionPane.showMessageDialog(panel,"Password
Updated");
                    newpass.setText("");
                }
            }

        });
        panel.add(update);

        obj.add(panel , "Profile");
    }
    public static void main(String agrs[])
    {
        Mentor obj = new Mentor();
        new MentorMenu(obj.getMentorById("EM7812") , new JFrame());
    }
    class MenuButton implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            String name = e.getActionCommand();
            CardLayout obj = (CardLayout) mainpanel.getLayout();
            obj.show(mainpanel, name);
        }

    }
```

```
        class MenteeList implements ActionListener
        {
                public void actionPerformed(ActionEvent arg0)
                {
                        JFrame f = new JFrame();
                        f.setSize(810,500);
                        Student obj = new Student();
                        ArrayList<StudentData> Sdata =
obj.getStudentbyMentor(data.EmployeeId);
                        if(Sdata == null || Sdata.size() == 0)
                        {
                                JOptionPane.showMessageDialog(mainpanel,"No Mentee");
                                return;
                        }

                        JPanel del = new JPanel();
                        JScrollPane scr = new JScrollPane(del);
                        scr.setPreferredSize(new Dimension(800,450));
                        del.setPreferredSize(new Dimension(800, 450));
                        scr.setAlignmentX(RIGHT_ALIGNMENT);
                        del.setLayout(new BoxLayout(del, BoxLayout.PAGE_AXIS));

                        for(StudentData x : Sdata)
                        {
                                JPanel m = makepanel(x);
                                del.add(m);
                        }
                        f.getContentPane().add(scr);
                        f.setVisible(true);
                }

                public JPanel makepanel(StudentData x)
                {
                        Border blackline = BorderFactory.createLineBorder(Color.black);
                        JPanel rp = new JPanel();
                        rp.setLayout(null);
                        rp.setBorder(blackline);
                        rp.setSize(700, 45);
                        JLabel name = new JLabel("Name: " + x.Name);
                        JLabel Id = new JLabel("Roll No.: " + x.Id);
                        JButton view = new JButton("View Full Profile");

                        name.setBounds(10,10,100,30);
                        Id.setBounds(150,10,200,30);
                        view.setBounds(450,10,250,30);

                        rp.add(name);
                        rp.add(Id);
                        rp.add(view);

                        view.addActionListener(
                                        new ActionListener()
                                        {
                                                public void actionPerformed(ActionEvent e)
                                                {
```

```
                                                new ProfileGenerator(x);
                                        }

                                });
                        return rp;
                }

        }
}
```

Warden Portal:

```
package se.menu;

import java.awt.CardLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.regex.Pattern;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextField;
import javax.swing.border.Border;

import se.Student.Student;
import se.Student.StudentData;
import se.computation.MS;
import se.computation.ProfileGenerator;
import se.menu.MentorMenu.MenteeList;
import se.request.Request;
import se.request.RequestData;
import se.warden.*;

public class WardenMenu extends JFrame
{
        JPanel sidepanel;
        JPanel mainpanel;
```

```
	JPanel requestmenu;
	JPanel summarymenu;
	JPanel profile;
	WardenData data;
	Warden Mobj;
	Request Robj;
	WardenMenu(WardenData obj , JFrame prev)
	{
		this.data = obj;
		Mobj = new Warden();
		Robj = new Request();
		setSize(1030,550);
		addSidePanel();
		addMainPanel();
		setLayout(null);
		setResizable(false);
		setVisible(true);
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		prev.dispose();
	}
	private void addSidePanel()
	{
		JPanel sidepanel = new JPanel();
		JLabel menuname = new JLabel("Warden Portal");
		menuname.setForeground(Color.white);
		menuname.setBounds(50,0,200,50);

		JButton RRequest = new JButton("Requests");
		RRequest.setBounds(0,150,200,30);
		RRequest.setBackground(Color.WHITE);
		RRequest.setForeground(Color.BLACK);
		RRequest.addActionListener(new MenuButton());

		JButton Summary = new JButton("Summary");
		Summary.setBounds(0,200,200,30);
		Summary.setBackground(Color.WHITE);
		Summary.setForeground(Color.BLACK);
		Summary.addActionListener(new MenuButton());

		JButton Profile = new JButton("Profile");
		Profile.setBounds(0,250,200,30);
		Profile.setBackground(Color.WHITE);
		Profile.setForeground(Color.BLACK);
		Profile.addActionListener(new MenuButton());

		JButton logout = new JButton("Logout");
		logout.setBounds(0,300,200,30);
		logout.setBackground(Color.WHITE);
		logout.setForeground(Color.BLACK);
		logout.addActionListener(new ActionListener()
				{
					public void actionPerformed(ActionEvent arg0)
					{
						int result =
JOptionPane.showConfirmDialog(sidepanel,"Sure? You want to exit?", "Swing Tester",
                                                JOptionPane.YES_NO_OPTION,
                                                JOptionPane.QUESTION_MESSAGE);
                                        if(result == JOptionPane.YES_OPTION)
                                        {
                                                JFrame del =
(JFrame)sidepanel.getParent().getParent().getParent().getParent();
                                                del.dispose();
                                        }
                                }

                        });
                sidepanel.add(logout);
                sidepanel.add(RRequest);
                sidepanel.add(Summary);
                sidepanel.add(menuname);
                sidepanel.add(Profile);

                sidepanel.setBounds(0, 0,200,500);
                sidepanel.setBackground(new Color(39,39,39));
                sidepanel.setLayout(null);
                this.add(sidepanel);
        }
        private void addMainPanel()
        {
                mainpanel = new JPanel();
                mainpanel.setBounds(200,0,810,500);
                mainpanel.setBackground(Color.WHITE);
                mainpanel.setLayout(new CardLayout());
                addRMenu(mainpanel);
                addSummary(mainpanel);
                addProfile(mainpanel);
                this.add(mainpanel);
        }
        private void addRMenu(JPanel obj)
        {
                requestmenu = new JPanel();
                JScrollPane listScroller = new JScrollPane(requestmenu);
                listScroller.setPreferredSize(new Dimension(800,550));
                requestmenu.setPreferredSize(new Dimension(800, 550));
                listScroller.setAlignmentX(RIGHT_ALIGNMENT);
                ArrayList<RequestData> rdata = Robj.getRequestbyHostel(data.HostelId);
                requestmenu.setLayout(new BoxLayout(requestmenu, BoxLayout.PAGE_AXIS));
                if(rdata != null)
                        for(RequestData x : rdata)
                        {
                                if(x.Status == 2)
                                        requestmenu.add(getRequest(x));
                                requestmenu.add(Box.createRigidArea(new Dimension(0,5)));
                        }
                obj.add(listScroller , "Requests");
        }
        private JPanel getRequest(RequestData request)
        {
                Border blackline = BorderFactory.createLineBorder(Color.black);
                JPanel panel = new JPanel();
```

```java
panel.setLayout(null);
panel.setBorder(blackline);
panel.setSize(new Dimension(800,200));

JLabel requestId = new JLabel("Request ID: " + request.RequestId);
requestId.setBounds(5,0,250,20);
panel.add(requestId);

JLabel StudentId = new JLabel("Student ID: "+request.StudentId);
StudentId.setBounds(255,0,200,20);
panel.add(StudentId);

JLabel From = new JLabel("From Date: " + request.fromDate);
From.setBounds(450,0,200,20);
panel.add(From);

JLabel To = new JLabel("To Date: "+ request.toDate);
To.setBounds(650,0,200,20);
panel.add(To);

JLabel Destination = new JLabel("Destination: " + request.Destination);
Destination.setBounds(5,40,250,20);
panel.add(Destination);

String type = "";
if(request.RequestType == 1)
	type = "Outing";
else if(request.RequestType == 2)
	type = "Vacation";
else
	type = "Medical";
JLabel Type = new JLabel("Type: "+ type);
Type.setBounds(255,40,250,20);
panel.add(Type);

JButton accept = new JButton("Accept");
accept.setBounds(450,40,100,30);
panel.add(accept);
accept.addActionListener( new ActionListener()
{
	public void actionPerformed(ActionEvent e)
	{
		if(Robj.updateRequestbyWarden(request.RequestId,6," "))
		{
			requestmenu.remove(panel);
			requestmenu.revalidate();
			requestmenu.repaint();
		}
	}

});

JButton decline = new JButton("Decline");
decline.setBounds(550,40,100,30);
panel.add(decline);
```

```
        decline.addActionListener(new ActionListener()
                    {
                          public void actionPerformed(ActionEvent e)
                          {
    if(Robj.updateRequestbyWarden(request.RequestId,5," "))
                                {
                                      requestmenu.remove(panel);
                                      requestmenu.revalidate();
                                      requestmenu.repaint();
                                }
                          }
                    });

        JButton view = new JButton("Reconsider");
        view.setBounds(650,40,130,30);
        panel.add(view);
        view.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e)
              {
                    String comment = JOptionPane.showInputDialog(panel,"Enter
Comment");

    if(Robj.updateRequestbyWarden(request.RequestId,4,comment))
                    {
                          requestmenu.remove(panel);
                          requestmenu.revalidate();
                          requestmenu.repaint();
                    }
              }
        });

        return panel;
    }
    private void addSummary(JPanel obj)
    {
        MS MSobj = new MS();
        int[] values = MSobj.getDataW(data.HostelId);

        JPanel panel = new JPanel();
        JLabel heading = new JLabel("Summary");
        heading.setBounds(405,5,810,30);
        panel.add(heading);
        JLabel one = new JLabel("Total Active requests");
        one.setBounds(10, 100,200, 30);
        JLabel two = new JLabel("Total Requests");
        two.setBounds(10,150,200,30);
        JLabel three = new JLabel("Students out of campus");
        three.setBounds(10, 200, 200, 30);
        JLabel oned = new JLabel("" + values[1]);
        oned.setBounds(220, 100,200, 30);
        JLabel twod = new JLabel("" + values[0]);
        twod.setBounds(220,150,200,30);
        JLabel threed = new JLabel(""+values[2]);
        threed.setBounds(220, 200, 200, 30);
```

```
		panel.add(three);
		panel.add(one);
		panel.add(two);
		panel.add(threed);
		panel.add(oned);
		panel.add(twod);
		panel.setLayout(null);

		JLabel sear = new JLabel("Search for Student");
		sear.setFont(new Font("Tahoma", 1, 24));
		sear.setBounds(10,300,500,30);
		panel.add(sear);

		JTextField idS = new JTextField("Enter ID");
		idS.setBounds(10,350,300,30);
		panel.add(idS);

		JButton search = new JButton("Search");
		search.setBounds(370,350,100,30);
		panel.add(search);
		search.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0)
			{
				String id = idS.getText();
				Student delobj = new Student();
				if(id.length() == 0 || !Pattern.matches("\\d{2}[a-zA-
Z]{3}\\d{4}",id) || !delobj.checkId(id))
				{
					JOptionPane.showMessageDialog(panel,"Incorrect Id");
					idS.setText("");
					return;
				}

				new ProfileGenerator(id);
			}
		});

		JLabel vmentee = new JLabel("View Hosteler List");
		vmentee.setFont(new Font("Tahoma", 1, 24));
		vmentee.setBounds(10,400,300,30);
		panel.add(vmentee);

		JButton view = new JButton("View");
		view.setBounds(370,400,100,30);
		panel.add(view);
		view.addActionListener(new HostelList());

		obj.add(panel , "Summary");
	}
	private void addProfile(JPanel obj)
	{
		JPanel panel = new JPanel();
		panel.setLayout(null);
		JLabel heading = new JLabel("Profile");
		heading.setBounds(405,5,810,30);
```

```
            panel.add(heading);
            JLabel id = new JLabel("New Employee Id");
            id.setBounds(10,100,200,30);
            JLabel idvalue = new JLabel(data.EmployeeId);
            idvalue.setBounds(220,100,200,30);
            JLabel name = new JLabel("Name");
            name.setBounds(10,150,200, 30);
            JLabel namevalue = new JLabel(data.name);
            namevalue.setBounds(220, 150, 200, 30);
            JLabel email = new JLabel("Email:");
            email.setBounds(10,200,200,30);
            JLabel valueemail = new JLabel(data.Emailid);
            valueemail.setBounds(220,200,200,30);
            JLabel hostel = new JLabel("Hostel ID");
            hostel.setBounds(10, 250, 200, 30);
            JLabel hostelvalue = new JLabel(data.HostelId);
            hostelvalue.setBounds(220, 250, 200, 30);

            panel.add(hostel);
            panel.add(hostelvalue);
            panel.add(id);
            panel.add(idvalue);
            panel.add(name);
            panel.add(namevalue);
            panel.add(email);
            panel.add(valueemail);

            JLabel uppass = new JLabel("Update Password");
            uppass.setBounds(10, 345 ,200,30);
            panel.add(uppass);

            JTextField newpass = new JTextField("New Pasword");
            newpass.setBounds(220,350,100,20);
            panel.add(newpass);

            JButton update = new JButton("Update");
            update.setBounds(350,350,100,20);
            update.addActionListener(new ActionListener() {

                public void actionPerformed(ActionEvent e)
                {
                    String newpassword = newpass.getText();
                    if(newpassword.length() == 0 || newpassword.length() > 10)
                    {
                        JOptionPane.showMessageDialog(panel,"Enter correct 
password");
                        return;
                    }

                    if(Mobj.updateWardenPassword(data.EmployeeId,newpassword))
                    {
                        JOptionPane.showMessageDialog(panel,"Password 
Updated");
                        newpass.setText("");
                    }
```

```
				}

			});
			panel.add(update);

			obj.add(panel , "Profile");
		}
		public static void main(String agrs[])
		{
			Warden obj = new Warden();
			new WardenMenu(obj.getWardenById("EHW2019") , new JFrame());
		}
		class MenuButton implements ActionListener
		{
			public void actionPerformed(ActionEvent e)
			{
				String name = e.getActionCommand();
				CardLayout obj = (CardLayout) mainpanel.getLayout();
				obj.show(mainpanel, name);
			}

		}

		class HostelList implements ActionListener
		{
			public void actionPerformed(ActionEvent arg0)
			{
				JFrame f = new JFrame();
				f.setSize(810,500);
				Student obj = new Student();
				ArrayList<StudentData> Sdata = 
	obj.getStudentsbyHostel(data.HostelId);
				if(Sdata == null || Sdata.size() == 0)
				{
					JOptionPane.showMessageDialog(mainpanel,"No Mentee");
					return;
				}

				JPanel del = new JPanel();
				JScrollPane scr = new JScrollPane(del);
				scr.setPreferredSize(new Dimension(800,450));
				del.setPreferredSize(new Dimension(800, 450));
				scr.setAlignmentX(RIGHT_ALIGNMENT);
				del.setLayout(new BoxLayout(del, BoxLayout.PAGE_AXIS));

				for(StudentData x : Sdata)
				{
					JPanel m = makepanel(x);
					del.add(m);
				}
				f.getContentPane().add(scr);
				f.setVisible(true);
			}

			public JPanel makepanel(StudentData x)
```

```
				{
					Border blackline = BorderFactory.createLineBorder(Color.black);
					JPanel rp = new JPanel();
					rp.setLayout(null);
					rp.setBorder(blackline);
					rp.setSize(700, 45);
					JLabel name = new JLabel("Name: " + x.Name);
					JLabel Id = new JLabel("Roll No.: " + x.Id);
					JButton view = new JButton("View Full Profile");

					name.setBounds(10,10,100,30);
					Id.setBounds(150,10,200,30);
					view.setBounds(450,10,250,30);

					rp.add(name);
					rp.add(Id);
					rp.add(view);

					view.addActionListener(
							new ActionListener()
							{
								public void actionPerformed(ActionEvent e)
								{
									new ProfileGenerator(x);
								}

							});
					return rp;
				}

		}
}
```

Security Portal:

```
package se.menu;

import java.awt.CardLayout;
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Date;
import se.Student.Student;
import se.request.*;
import javax.swing.*;
import javax.swing.border.Border;

public class SecurityMenu extends JFrame
{
	JPanel mainpanel;
	JPanel inpanel;
	JPanel outpanel;
    ArrayList<RequestData> allapprovedrequest;
	Request getRequest = new Request();
	Student Sobj = new Student();
	SecurityMenu(JFrame prev)
	{
		allapprovedrequest = getRequest.getAllApprovedRequest();
		setSize(1010,500);
		addSidePanel();
		addMainPanel();
		setLayout(null);
		setResizable(false);
		setVisible(true);
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		prev.dispose();
	}
	private void addSidePanel()
	{
		JPanel sidepanel = new JPanel();
		JLabel menuname = new JLabel("Security Menu");
		menuname.setForeground(Color.white);
		menuname.setBounds(50,0,200,50);
		JLabel title = new JLabel("Requests");
		title.setBounds(600,0,410,50);
		JButton IntimeMenu = new JButton("In-Time");
		IntimeMenu.setBounds(0,150,200,30);
		IntimeMenu.setBackground(Color.WHITE);
		IntimeMenu.setForeground(Color.BLACK);
		IntimeMenu.addActionListener(new MenuButton());
		JButton OuttimeMenu = new JButton("Out-Time");
		OuttimeMenu.setBounds(0,200,200,30);
		OuttimeMenu.setBackground(Color.WHITE);
		OuttimeMenu.setForeground(Color.BLACK);
		OuttimeMenu.addActionListener(new MenuButton());
		sidepanel.add(IntimeMenu);
		sidepanel.add(OuttimeMenu);
		sidepanel.add(menuname);
		this.add(title);

		JButton logout = new JButton("Logout");
		logout.setBounds(0,250,200,30);
		logout.setBackground(Color.WHITE);
		logout.setForeground(Color.BLACK);
		logout.addActionListener(new ActionListener()
				{
					public void actionPerformed(ActionEvent arg0)
					{
						int result = 
JOptionPane.showConfirmDialog(sidepanel,"Sure? You want to exit?", "Swing Tester",
                                                    JOptionPane.YES_NO_OPTION,
                                                    JOptionPane.QUESTION_MESSAGE);
                                            if(result == JOptionPane.YES_OPTION)
                                            {
                                                    JFrame del =
(JFrame)sidepanel.getParent().getParent().getParent().getParent();
                                                    del.dispose();
                                            }
                                    }

                            });
                    sidepanel.add(logout);

                    sidepanel.setBounds(0, 0,200,500);
                    sidepanel.setBackground(new Color(39,39,39));
                    sidepanel.setLayout(null);
                    this.add(sidepanel);
            }
            private void addMainPanel()
            {

                    mainpanel = new JPanel();
                    mainpanel.setBounds(200,60,800,400);
                    mainpanel.setBackground(Color.WHITE);
                    mainpanel.setLayout(new CardLayout());
                    addInrequest(mainpanel);
                    addOutrequest(mainpanel);
                    this.add(mainpanel);
            }
            private void addInrequest(JPanel obj)
            {
                    inpanel = new JPanel();
                    inpanel.setBounds(200,50,750,500);
                    inpanel.setBackground(Color.WHITE);
                    inpanel.setLayout(new BoxLayout (inpanel, BoxLayout.Y_AXIS));
                    for(RequestData x : allapprovedrequest)
                            if(x.intime == null && x.outtime != null)
                                    inpanel.add(getRequestPanel(x,true));
                    obj.add(inpanel , "inmenu");
            }
            private void addOutrequest(JPanel obj)
            {
                    outpanel = new JPanel();
                    outpanel.setBounds(200,50,750,500);
                    outpanel.setBackground(Color.WHITE);
                    outpanel.setLayout(new BoxLayout (outpanel, BoxLayout.Y_AXIS));
                    for(RequestData x : allapprovedrequest)
                    {
                            if(x.intime == null && x.outtime == null)
                                    outpanel.add(getRequestPanel(x,false));
                    }
                    obj.add(outpanel , "outmenu");
            }
            private JPanel getRequestPanel(RequestData request , boolean type)
            {
```

```
            Border blackline = BorderFactory.createLineBorder(Color.black);
            JPanel panel = new JPanel();
            panel.setLayout(null);
            panel.setBorder(blackline);
            JLabel requestId = new JLabel("Request ID: " + request.RequestId);
            requestId.setBounds(5,0,500,20);
            panel.add(requestId);

            JLabel StudentId = new JLabel("Student ID: "+request.StudentId);
            StudentId.setBounds(5,30,500,20);
            panel.add(StudentId);

            JButton button = new JButton("Update");
            button.setBounds(600,12,100,26);
            if(type)
                  button.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              System.out.println(request.RequestId);
                              String id = request.RequestId;
                              Date time = new Date();
                              if(getRequest.updateIntime(id, time) &&
Sobj.updatestatus(request.StudentId, 0))
                              {
                                    inpanel.remove(panel);
                                    inpanel.revalidate();
                                    inpanel.repaint();
                                    RequestData deldata =
getRequest.getRequestById(request.RequestId);
                                    if(getRequest.checkIfStudentLate(deldata))
                                    {

      JOptionPane.showMessageDialog(panel,"Student is late");
                                          Student obj = new Student();
                                          obj.updatetimeslate(request.StudentId);
                                    }
                              }
                        }
                  });
            else
            {
                  button.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              System.out.println(request.RequestId);
                              String id = request.RequestId;
                              Date time = new Date();
                              if(getRequest.updateOuttime(id, time) &&
Sobj.updatestatus(request.StudentId, 1))
                              {
                                    RequestData deldata =
getRequest.getRequestById(request.RequestId);
                                    if(deldata == null)
                                          return;
                                    inpanel.add(getRequestPanel(deldata,true));
                                    inpanel.revalidate();
                                    inpanel.repaint();
```

```
                                        outpanel.remove(panel);
                                        outpanel.revalidate();
                                        outpanel.repaint();
                                   }
                              }
                         });
                    }
                    panel.add(button);

                    return panel;
               }
          class MenuButton implements ActionListener
               {

                    @Override
                    public void actionPerformed(ActionEvent e)
                    {
                         String name = e.getActionCommand();
                         CardLayout obj = (CardLayout) mainpanel.getLayout();
                         if(name.equals("In-Time"))
                         {
                              obj.show(mainpanel, "inmenu");
                         }
                         else
                         {
                              obj.show(mainpanel,"outmenu");
                         }
                    }

               }
          public static void main(String agrs[])
               {
                    new SecurityMenu(new JFrame());
               }
     }
```

Login Page:

```
package se.menu;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;
import se.Student.Student;
import se.Student.StudentData;
import se.mentor.Mentor;
import se.mentor.MentorData;
import se.warden.Warden;
import se.warden.WardenData;

import java.awt.event.*;
import java.util.regex.Pattern;
import java.awt.*;

public class Login extends JFrame
{
	Login()
	{
		this.setSize(400,300);
		this.setLayout(null);
		loadscreen();
		this.setVisible(true);
		this.setResizable(false);
		this.getContentPane().setBackground(new Color(45, 62, 80));
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}
	private void loadscreen()
	{
		JLabel top = new JLabel("  Login Form");
		top.setOpaque(true);
		top.setBounds(0,0,500,40);
		top.setBackground(new Color(249, 148, 6));
		top.setForeground(Color.WHITE);
		top.setFont(new Font("Tahoma", 1, 24));
		this.add(top);

		JLabel user = new JLabel("Username");
		user.setBounds(10,95,100,30);
		user.setForeground(Color.WHITE);
		user.setFont(new Font("Tahoma", 0, 14));
		this.add(user);

		JTextField username = new JTextField();
		username.setBounds(120,100,200,20);
```

```
        this.add(username);

        JLabel pass = new JLabel("Password");
        pass.setBounds(10,145,100,30);
        pass.setForeground(Color.WHITE);
        pass.setFont(new Font("Tahoma", 0, 14));
        this.add(pass);

        JPasswordField password = new JPasswordField();
        password.setEchoChar('*');
        password.setBounds(120,150,200,20);
        this.add(password);

        JButton login = new JButton("Login");
        login.setBounds(220,190,100,30);
        login.setForeground(Color.WHITE);
        login.setFont(new Font("Tahoma", 0, 14));
        login.setBackground(new Color(57, 185, 255));
        login.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e)
            {
                String user = username.getText();
                String passw = password.~~getText()~~;
                JFrame frame = (JFrame)SwingUtilities.getRoot((Component)
e.getSource());

                boolean student = Pattern.matches("\\d{2}[a-zA-
Z]{3}\\d{4}", user);
                if(student)
                    if(checker(user , passw , 1))
                        MenuLoader(user,1);
                    else
                        error();

                boolean mentor = Pattern.matches("[E][M]\\d*",user);
                if(mentor)
                    if(checker(user , passw ,2))
                        MenuLoader(user ,2);
                    else
                        error();

                boolean warden = Pattern.matches("[E][H][W]\\d*",user);
                if(warden)
                    if(checker(user , passw , 3))
                        MenuLoader(user , 3);
                    else
                        error();

                boolean security = Pattern.matches("[E][S]\\d*",user);
                if(security)
                    if(checker(user , passw , 4))
                        MenuLoader(user,4);
                    else
                        error();
                    boolean admin = user.equals("admin");
                    if(admin)
                          if(passw.equals("password"))
                                MenuLoader(user,5);
                          else
                                error();
              }
        });
        this.add(login);
        JLabel register = new JLabel("Don't have a account?");
        register.setBounds(140,225,200,30);
        register.setForeground(Color.WHITE);
        register.setFont(new Font("Tahoma", 0, 10));
        register.addMouseListener(new MouseListener()
        {

              @Override
              public void mouseClicked(MouseEvent arg0)
              {
                    new RegistrationPage();
                    JFrame frame = (JFrame)SwingUtilities.getRoot((Component)
arg0.getSource());
                    frame.dispose();
              }
              @Override
              public void mouseEntered(MouseEvent arg0) {
                    // TODO Auto-generated method stub

              }
              @Override
              public void mouseExited(MouseEvent arg0) {
                    // TODO Auto-generated method stub

              }
              @Override
              public void mousePressed(MouseEvent arg0) {
                    // TODO Auto-generated method stub

              }
              @Override
              public void mouseReleased(MouseEvent arg0) {
                    // TODO Auto-generated method stub

              }
        });
        this.add(register);

  }
  private void MenuLoader(String id, int n)
  {
        switch(n)
        {
        case 1: Student S = new Student();
                    StudentData data = S.getStudentById(id);
                    new StudentMenu(data,this);
```

```
                        return;

                case 2: Mentor M = new Mentor();
                        MentorData Mdata =M.getMentorById(id);
                        new MentorMenu(Mdata,this);
                        return;

                case 3:         Warden W = new Warden();
                        WardenData Wdata = W.getWardenById(id);
                        new WardenMenu(Wdata,this);
                        return;

                case 4: new SecurityMenu(this);
                        return;

                case 5: new Admin(this);
                        return;
                }
        }
        private boolean checker(String username , String password , int n)
        {
                switch(n)
                {
                case 1: Student S = new Student();
                        String getps = S.getPassword(username);
                        return  getps != null && getps.equals(password);

                case 2: Mentor M = new Mentor();
                        String Wgetps = M.getPassword(username);
                        return  Wgetps != null && Wgetps.equals(password);

                case 3:         Warden W = new Warden();
                        String Mgetps = W.getPassword(username);
                        return  Mgetps != null && Mgetps.equals(password);

                case 4: return username.equals("ES1001") && password.equals("login");

                default : return false;
                }
        }
        private void error()
        {
                JOptionPane.showMessageDialog(this,("Invalid Username/Password"));
        }
        public static void main(String agrs[])
        {
                new Login();
        }
}
```

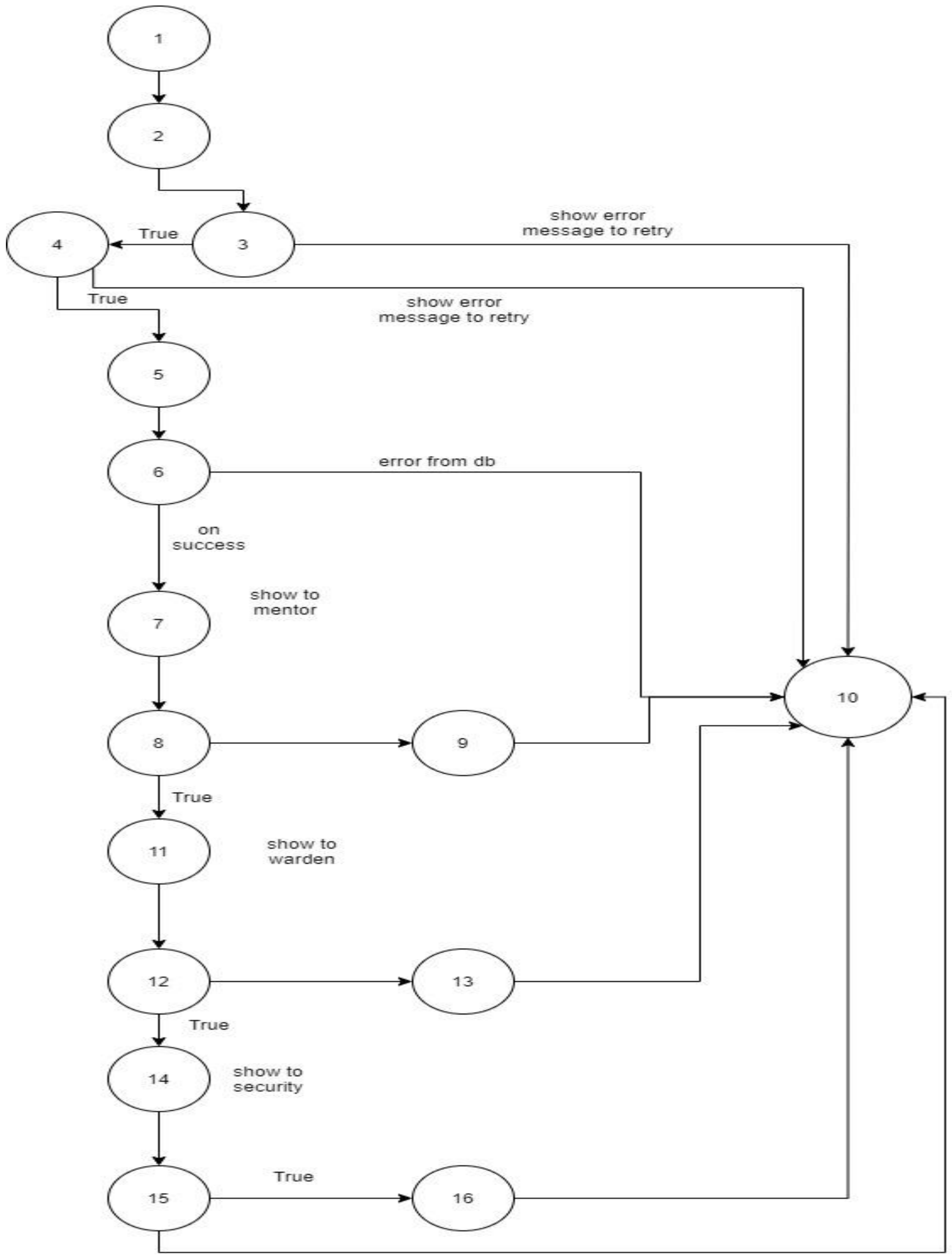
Registration Page:

```
package se.menu;

import java.awt.Color;
import java.awt.Component;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.regex.Pattern;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;
import se.Student.Student;
import se.Student.StudentData;
import se.mentor.Mentor;
import se.mentor.MentorData;
import se.warden.WardenData;

public class RegistrationPage extends JFrame
{

	RegistrationPage()
	{
		this.setLayout(null);
		this.setSize(500,600);
		loadscreen();
		this.setResizable(false);
		this.setVisible(true);
		this.getContentPane().setBackground(new Color(45, 62, 80));
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	private void loadscreen()
	{
		JLabel heading  = new JLabel("  Registration Form");
		heading.setBounds(0,0,400,40);
		heading.setBackground(new Color(249, 148, 6));
		heading.setForeground(Color.WHITE);
		heading.setFont(new Font("Tahoma", 1, 24));
		heading.setOpaque(true);
		this.add(heading);

		String[] names = new String[] {"Name","ID","Email","Gender","Room 
Number","Hostel","Mentor Id","Password"};
		JLabel[] label = new JLabel[names.length];
		JTextField[] text = new JTextField[names.length];
		int counter = 0;
		int y = 70;
		for(String x : names)
		{
			label[counter] = new JLabel(x);
			label[counter].setBounds(10,y,200,30);
			label[counter].setForeground(Color.WHITE);
			label[counter].setFont(new Font("Tahoma", 0, 14));
			this.add(label[counter]);

			text[counter] = new JTextField();
			text[counter].setBounds(220,y,200,20);
			this.add(text[counter]);
			y = y + 50;
			counter++;
		}

		JButton check = new JButton("Check");
		check.setBounds(10,470,100,25);
		this.add(check);

		JButton save = new JButton("Register");
		save.setBounds(300,470,100,25);
		save.setEnabled(false);
		this.add(save);

		JLabel login = new JLabel("Back to Login");
		login.setBackground(new Color(249, 148, 6));
		login.setBounds(400,0,100,40);
		login.setForeground(Color.WHITE);
		login.setFont(new Font("Tahoma", 0, 14));
		login.setOpaque(true);
		login.addMouseListener(new MouseListener()
		{

			@Override
			public void mouseClicked(MouseEvent arg0)
			{
				new Login();
				JFrame frame = (JFrame)SwingUtilities.getRoot((Component)
arg0.getSource());
				frame.dispose();
			}
			@Override
			public void mouseEntered(MouseEvent arg0) {
				// TODO Auto-generated method stub

			}
			@Override
			public void mouseExited(MouseEvent arg0) {
				// TODO Auto-generated method stub

			}
			@Override
			public void mousePressed(MouseEvent arg0) {
				// TODO Auto-generated method stub

			}
			@Override
			public void mouseReleased(MouseEvent arg0) {
				// TODO Auto-generated method stub

			}
		});
		this.add(login);


		Mentor mobj = new Mentor();
		check.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e)
			{
				for(JTextField x : text)
					if(x.getText().length() == 0)
					{
	JOptionPane.showMessageDialog(check.getParent(),"Fill complete form");
						return;
					}

				String passwordString = text[7].getText();
				if(passwordString.length() > 10)
				{
	JOptionPane.showMessageDialog(check.getParent(),"Password length should be 
less than 10");
					return;
				}
				if(!(Pattern.matches("\\d{2}[a-zA-Z]{3}\\d{4}", 
text[1].getText()) && Pattern.matches("\\d{3}", text[4].getText())))
				{
	JOptionPane.showMessageDialog(check.getParent(),"Student ID not proper");
					return;
				}
				String id = text[6].getText();
				String hostel = text[5].getText();
				String gender = ""+text[3].getText().charAt(0);
				String hg = ""+text[5].getText().charAt(0);
				gender = gender.toUpperCase();
				if(!hg.equals(gender))
				{
	JOptionPane.showMessageDialog(check.getParent(),("Please choose proper 
hostel"));
					return;
				}
				boolean pass = Pattern.matches("[E][M]\\d*",id);
				if(pass && mobj.checkId(id))
					if(hostel.equals("MH1") || hostel.equals("MH2") || 
hostel.equals("MH3"))
```

```
                                        save.setEnabled(true);
                                else

        JOptionPane.showMessageDialog(check.getParent(),("Invalid HostelId"));
                        else

        JOptionPane.showMessageDialog(check.getParent(),"Mentor ID not proper");
                }
        });

        save.addActionListener(new ActionListener()
        {
                public void actionPerformed(ActionEvent e)
                {
                        StudentData data = new StudentData();
                        data.Name =  text[0].getText();
                        data.Id =    text[1].getText();
                        data.Email = text[2].getText();
                        data.gender = ""+text[3].getText().charAt(0);
                        data.RoomNumber = Integer.parseInt(text[4].getText());
                        data.HostelName = text[5].getText();
                        data.MentorDetails =
mobj.getMentorById(text[6].getText());
                        String password = text[7].getText();

                        Student obj = new Student();
                        if(obj.addStudent(data, password))
                        {

        JOptionPane.showMessageDialog(check.getParent(),"Saved Data");
                                for(JTextField x : text)
                                        x.setText("");
                        }
                        else
                                JOptionPane.showMessageDialog(check.getParent(),"Try
Again");
                }
        });
    }

    public static void main(String agrs[])
    {
        new RegistrationPage();
    }
}
```

TESTING –WHITE BOX

Diagram:



All node details

1. Load request form
2. Fill all details
3. if destination < 10 char
4. if from date < to date
5. create request
6. if got RequestID
7. set status=1
8. if mentor accept
9. set status=2
10. Close request
11. set status=3
12. if warden accept
13. set status=4
14. set status=5 , Enter Exit and Entry date.
15. if Entry date > Request Last date
16. set Late entry as true

Complexity calculation

V(G) = E - N + 2 => 21 -16 +2 = 7

V(G) = P + 1 => 6 + 1 = 7

(Conditional nodes are 3,4,6,8,12,15)

All possible path:

1. 1 , 2 , 3 , 10
2. 1 , 2 , 3 , 4 , 10
3. 1 , 2 , 3 , 4 , 5 , 6 , 10
4. 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10
5. 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 11 , 12 ,13 , 10
6. 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 11 , 12 ,14 , 15 , 10
7. 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 11 , 12 ,14 , 15 , 16 ,10

Testcases:

1. Load -> Entered destination greater than 10 char or less than 1 char -> Request Denied

2. Load -> Entered illogical from and to date (from date < current date or from date < to date) -> Request Denied

3. Load -> Filled all request details properly -> Failed to make proper request to DB to make request -> Request Denied

4. Load -> Filled all request details properly -> status =1 ->Mentor Declined -> status = 2 -> Request Denied and closed.

5. Load -> Filled all request details properly -> status =1 ->Mentor Accepted-> status = 3 -> Warden Declined -> status = 4 -> Request Denied and closed.

6. Load -> Filled all request details properly -> status =1 ->Mentor Accepted-> status = 3 -> Warden Approved-> status = 5 -> Request Approved and student can leave campus -> Guard enters Entry and Exit date -> Entry date < Request Entry date -> Request closed.

7. Load -> Filled all request details properly -> status =1 ->Mentor Accepted-> status = 3 -> Warden Approved-> status = 5 -> Request Approved and student can leave campus -> Guard enters Entry and Exit date -> Entry date > Request Entry date -> Request marked as late -> Request closed.

PROJECT ESTIMATION (LOC & COCOMO)

LOC based Estimation:

Function	LOC
Summary Generator	300
Database Connection	500
Mentor Interface	600
Warden Interface	610
Student Interface	650
Login	203
Registration	183
Security Interface	210
Total	3256

Estimated LOC: 3256 LOC
Burdened Labor Rate: 24300 rupees/month
Average productivity for systems: 1000 LOC/pm

a.) Cost per line of code = 24300/1000 = 24.3 rupees

b.) Total estimated project cost = 3256*24.3 = 79,120.8 rupees

c.) Estimated effort (person) = 79120.8/24300 = 3.256 = 4 people in a month

(approx.)

Basic COCOMO model:

The estimated LOC is 3256 or 3.256K LOC. So the project is of Organic type.

Organic mode

a_b = 2.4
b_b = 1.05
c_b = 2.5
d_b = 0.38

E = a_b (KLOC)^b = 2.4 * (3.256) ^ (1.05) = 8.28952

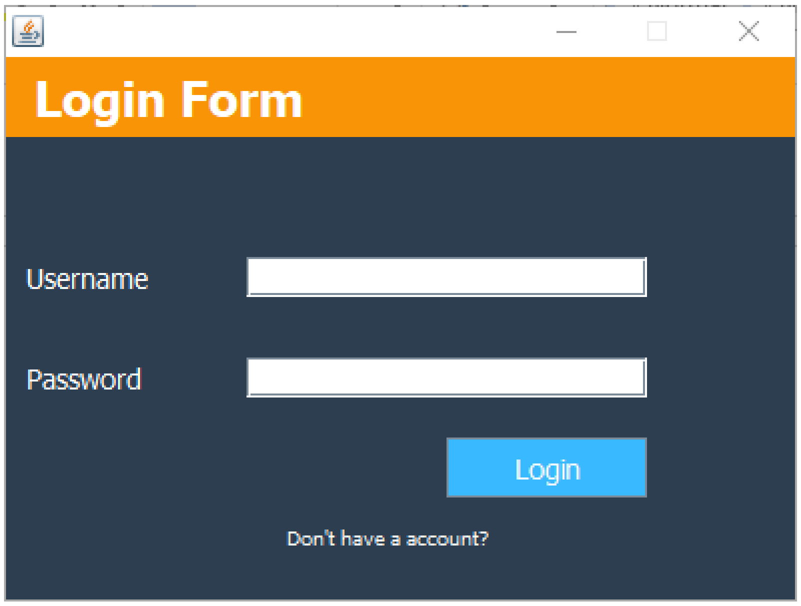
D = c_b (E)^d_b = 2.5 * (8.28952) ^ (0.38) = 5.5844

SS = E/D persons = 1.4844 persons = 2 persons(approx.)

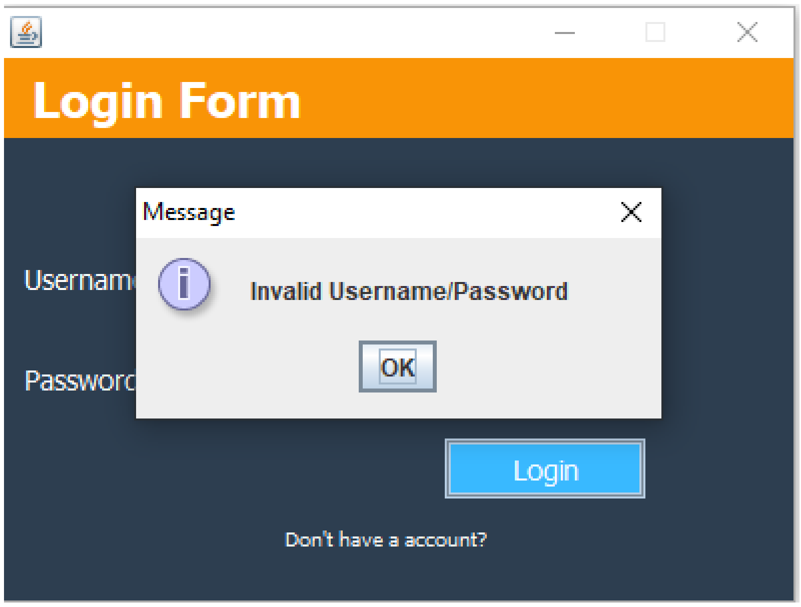
P = KLOC/E = 0.3929

RESULTS

Login Form:

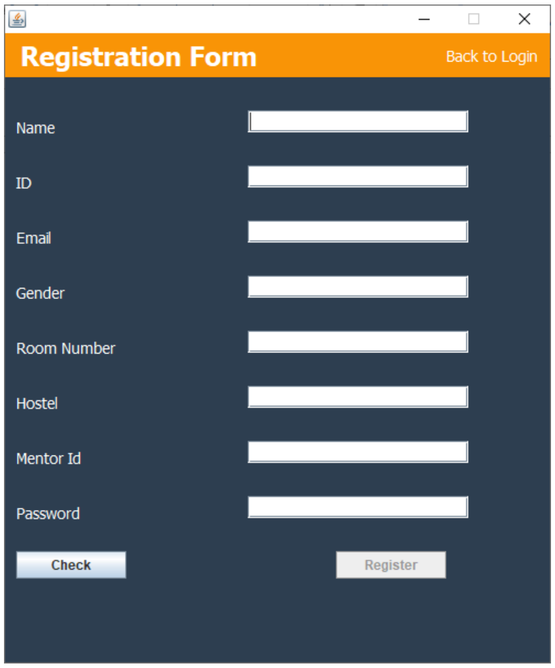


Login Form (Wrong Username/Password):

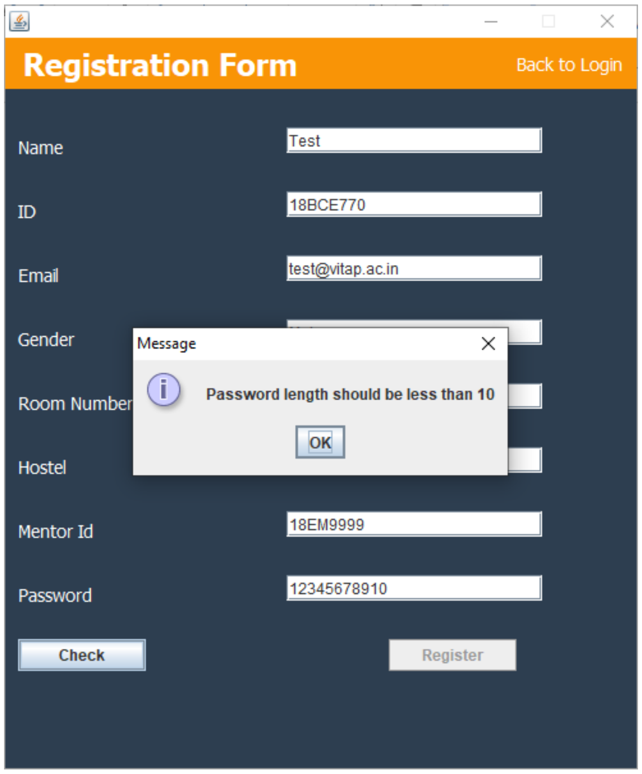


Invalid Username/Password

Registration Form:



Registration Form – Wrong Length of Password



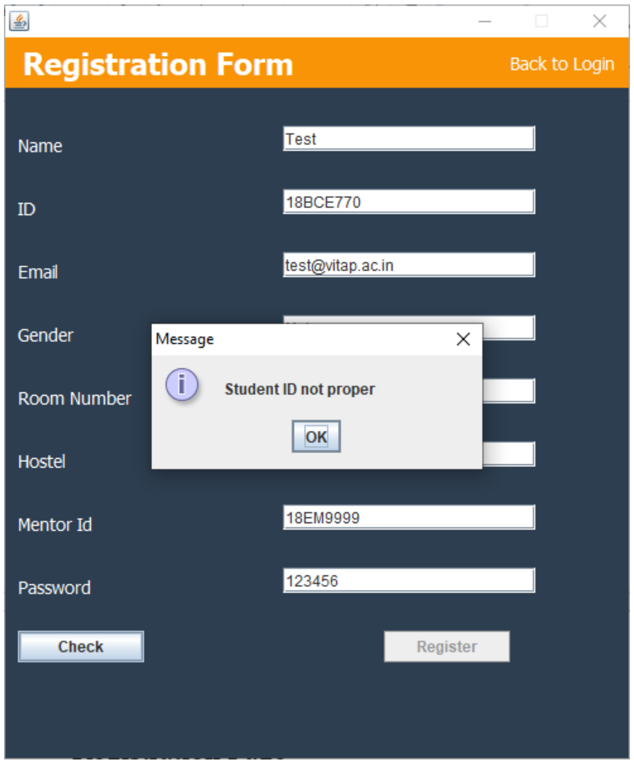
Registration Form
Back to Login

Name	Test
ID	18BCE770
Email	test@vitap.ac.in
Gender	
Room Number	
Hostel	
Mentor Id	18EM9999
Password	12345678910

Message

Password length should be less than 10

Registration Page – Wrong ID format



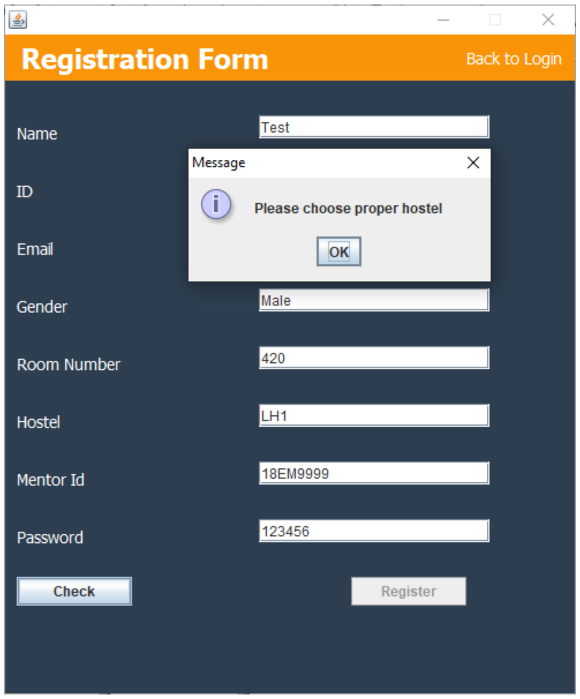
Registration Form
Back to Login

Name	Test
ID	18BCE770
Email	test@vitap.ac.in
Gender	
Room Number	
Hostel	
Mentor Id	18EM9999
Password	123456

Message

Student ID not proper

Registration Page – When Hostel Type and gender do not match



Registration Form

Back to Login

Name: Test

ID

Email

Gender: Male

Room Number: 420

Hostel: LH1

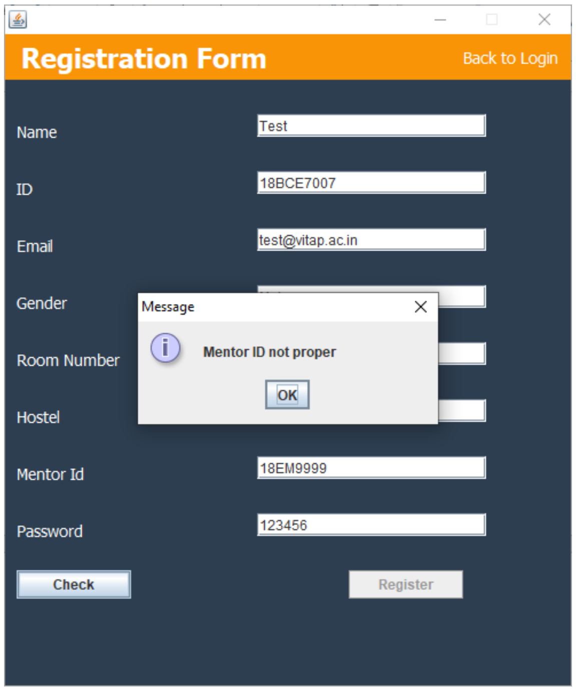
Mentor Id: 18EM9999

Password: 123456

Message

Please choose proper hostel

Registration Page – Mentor Id does not exists



Registration Form

Back to Login

Name: Test

ID: 18BCE7007

Email: test@vitap.ac.in

Gender

Room Number

Hostel

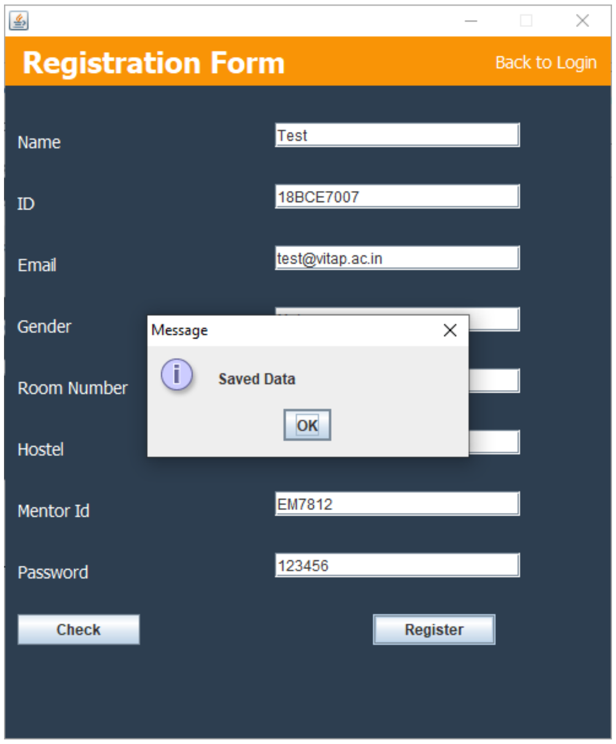
Mentor Id: 18EM9999

Password: 123456

Message

Mentor ID not proper

Registration Page – Accepted



Registration Form

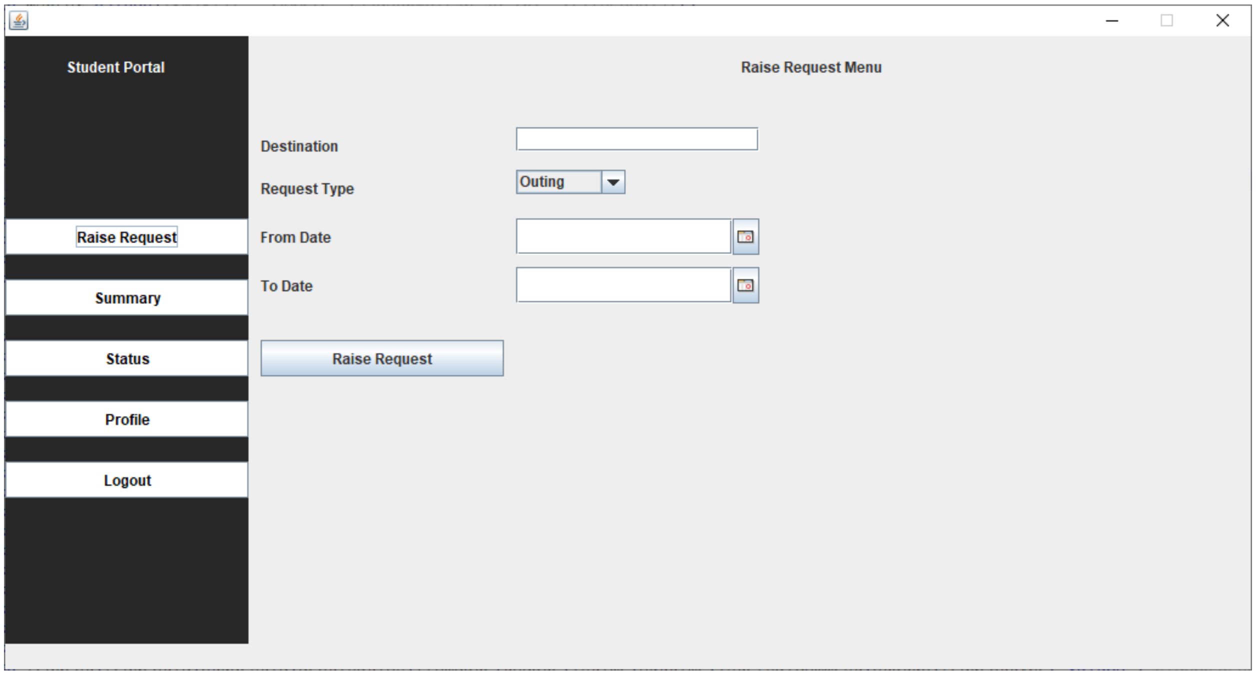
Back to Login

Name	Test
ID	18BCE7007
Email	test@vitap.ac.in
Gender	
Room Number	
Hostel	
Mentor Id	EM7812
Password	123456

Message

Saved Data

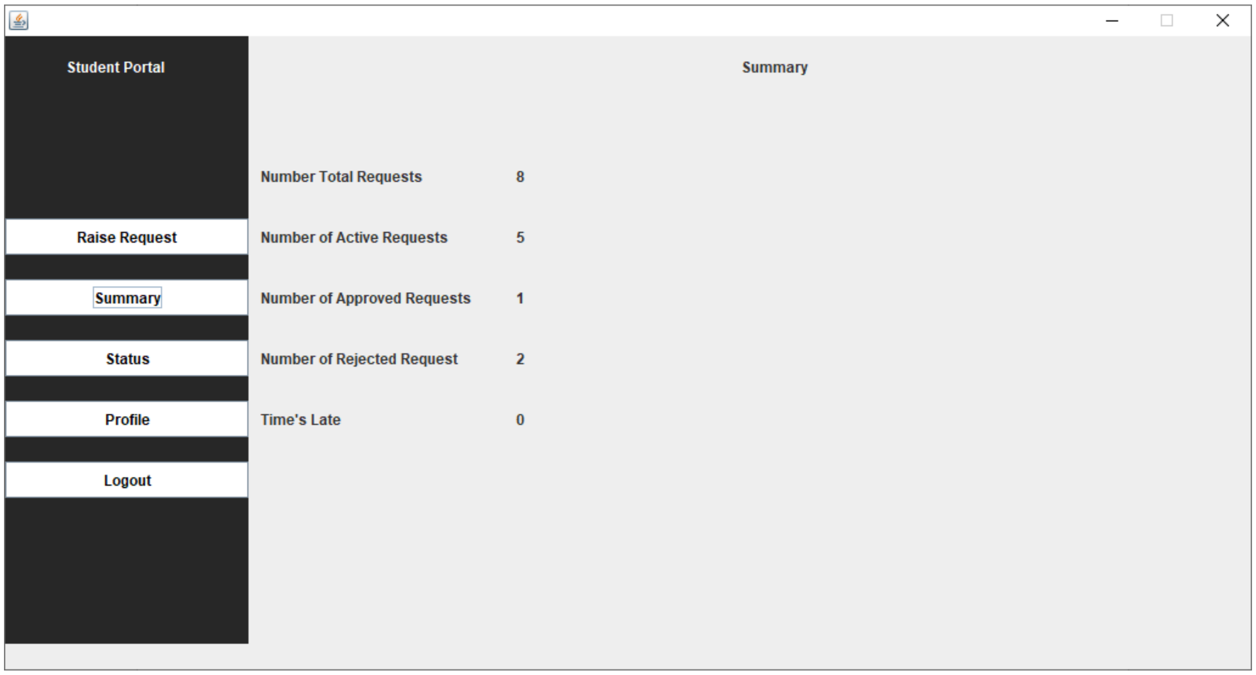
Student Portal - Raise Request Menu



Raise Request Menu

Destination	
Request Type	Outing
From Date	
To Date	

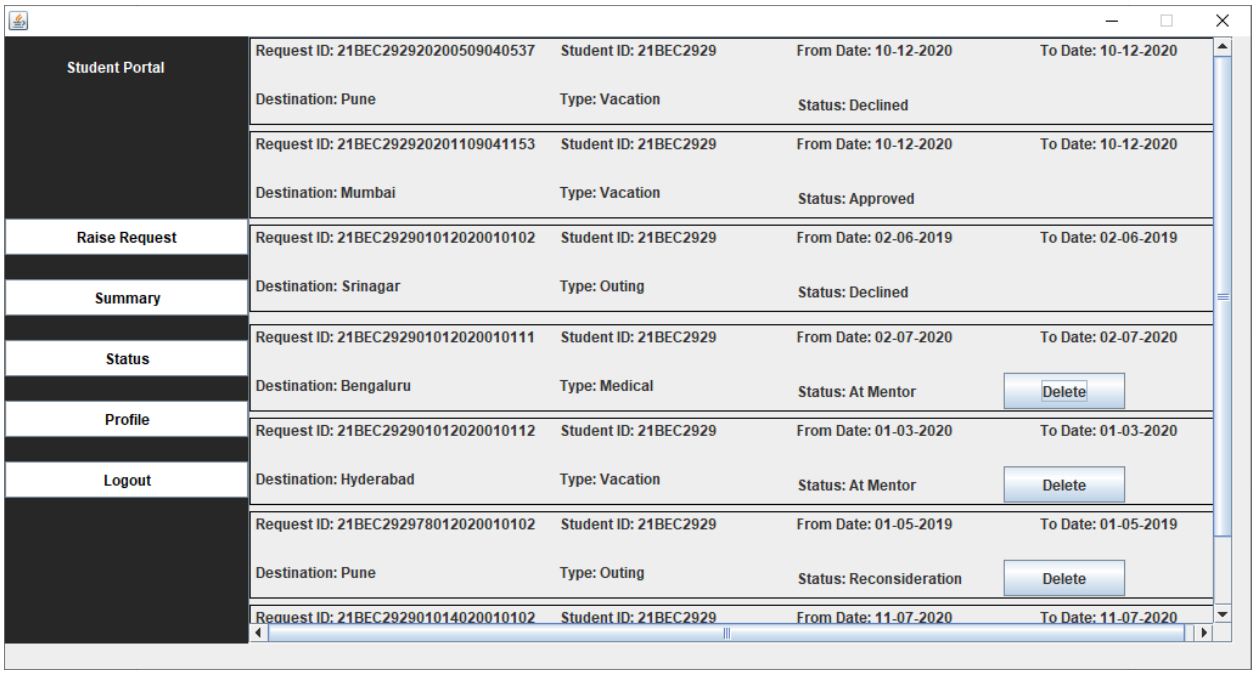
Student Portal - Summary Menu



Summary

Number Total Requests	8
Number of Active Requests	5
Number of Approved Requests	1
Number of Rejected Request	2
Time's Late	0

Student Portal – Status Menu



Request ID: 21BEC292920200509040537 Student ID: 21BEC2929 From Date: 10-12-2020 To Date: 10-12-2020
Destination: Pune Type: Vacation Status: Declined

Request ID: 21BEC292920201109041153 Student ID: 21BEC2929 From Date: 10-12-2020 To Date: 10-12-2020
Destination: Mumbai Type: Vacation Status: Approved

Request ID: 21BEC292901012020010102 Student ID: 21BEC2929 From Date: 02-06-2019 To Date: 02-06-2019
Destination: Srinagar Type: Outing Status: Declined

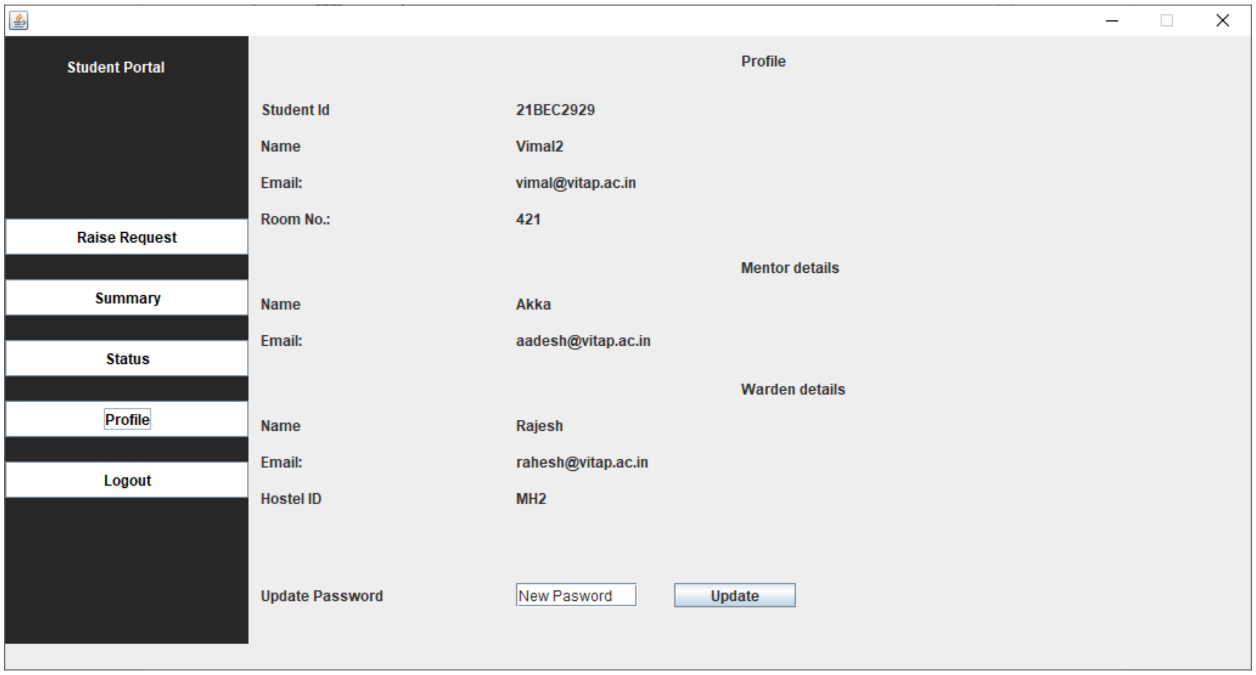
Request ID: 21BEC292901012020010111 Student ID: 21BEC2929 From Date: 02-07-2020 To Date: 02-07-2020
Destination: Bengaluru Type: Medical Status: At Mentor Delete

Request ID: 21BEC292901012020010112 Student ID: 21BEC2929 From Date: 01-03-2020 To Date: 01-03-2020
Destination: Hyderabad Type: Vacation Status: At Mentor Delete

Request ID: 21BEC292978012020010102 Student ID: 21BEC2929 From Date: 01-05-2019 To Date: 01-05-2019
Destination: Pune Type: Outing Status: Reconsideration Delete

Request ID: 21BEC292901014020010102 Student ID: 21BEC2929 From Date: 11-07-2020 To Date: 11-07-2020

Student Portal – Profile



Profile

Student Id	21BEC2929
Name	Vimal2
Email:	vimal@vitap.ac.in
Room No.:	421

Mentor details

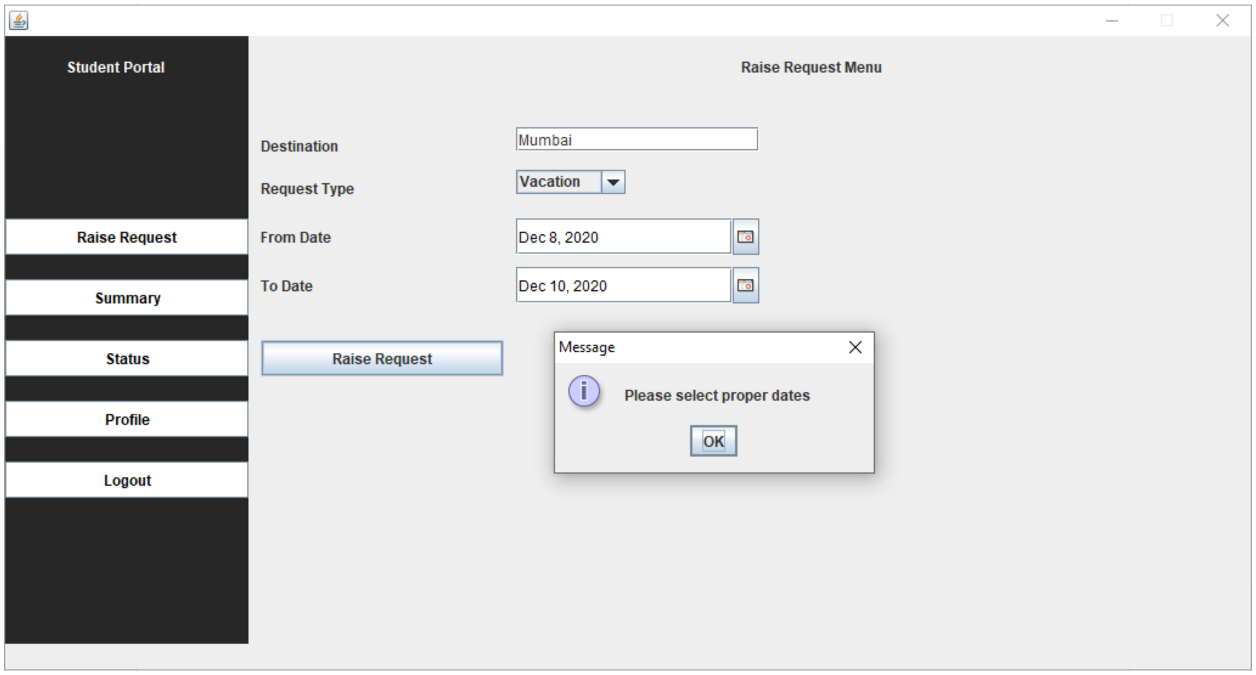
Name	Akka
Email:	aadesh@vitap.ac.in

Warden details

Name	Rajesh
Email:	rahesh@vitap.ac.in
Hostel ID	MH2

Update Password New Pasword Update

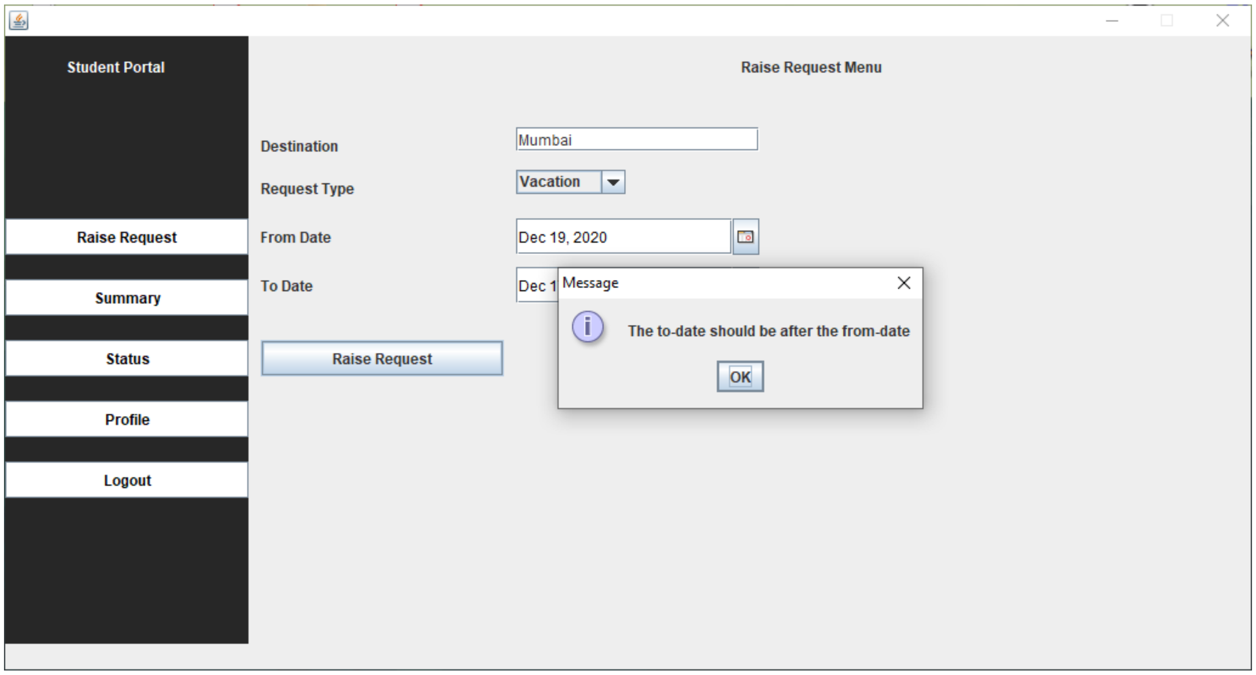
Student Portal – Raise Request Menu – Wrong date error messages



Destination: Mumbai
Request Type: Vacation
From Date: Dec 8, 2020
To Date: Dec 10, 2020

Message

Please select proper dates

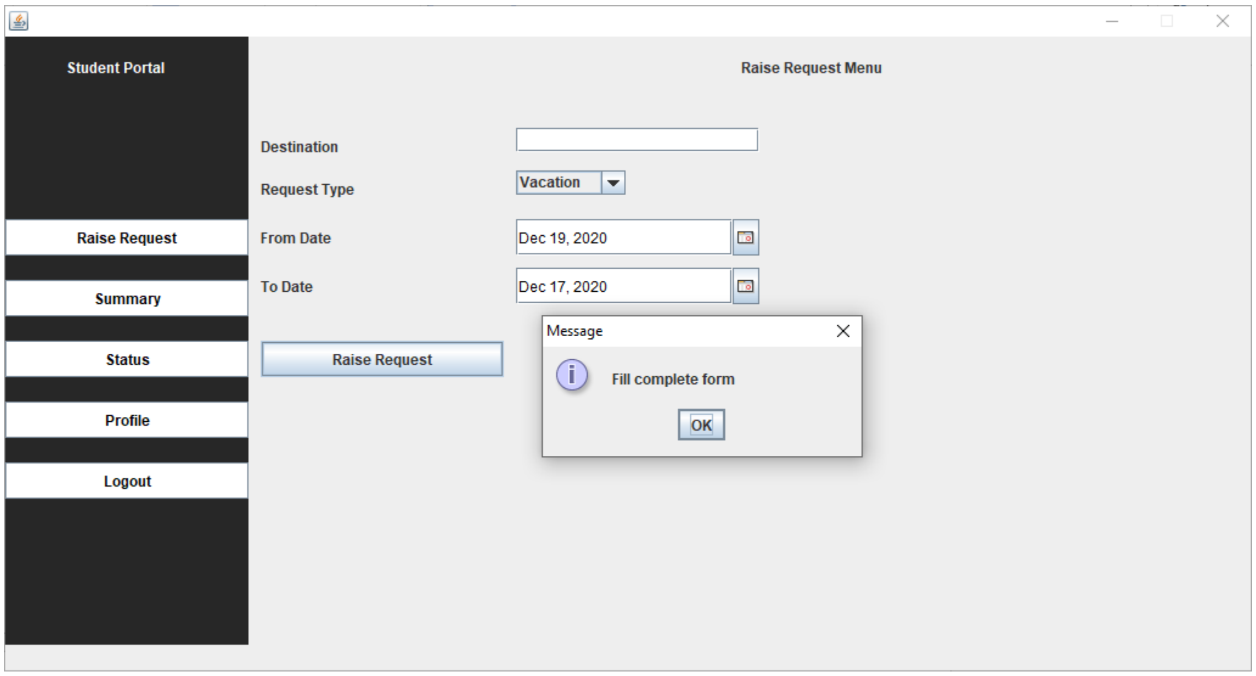


Destination: Mumbai
Request Type: Vacation
From Date: Dec 19, 2020
To Date: Dec 1

Message

The to-date should be after the from-date

Student Portal – Raise Request Menu – Incomplete form



Destination

Request Type: Vacation

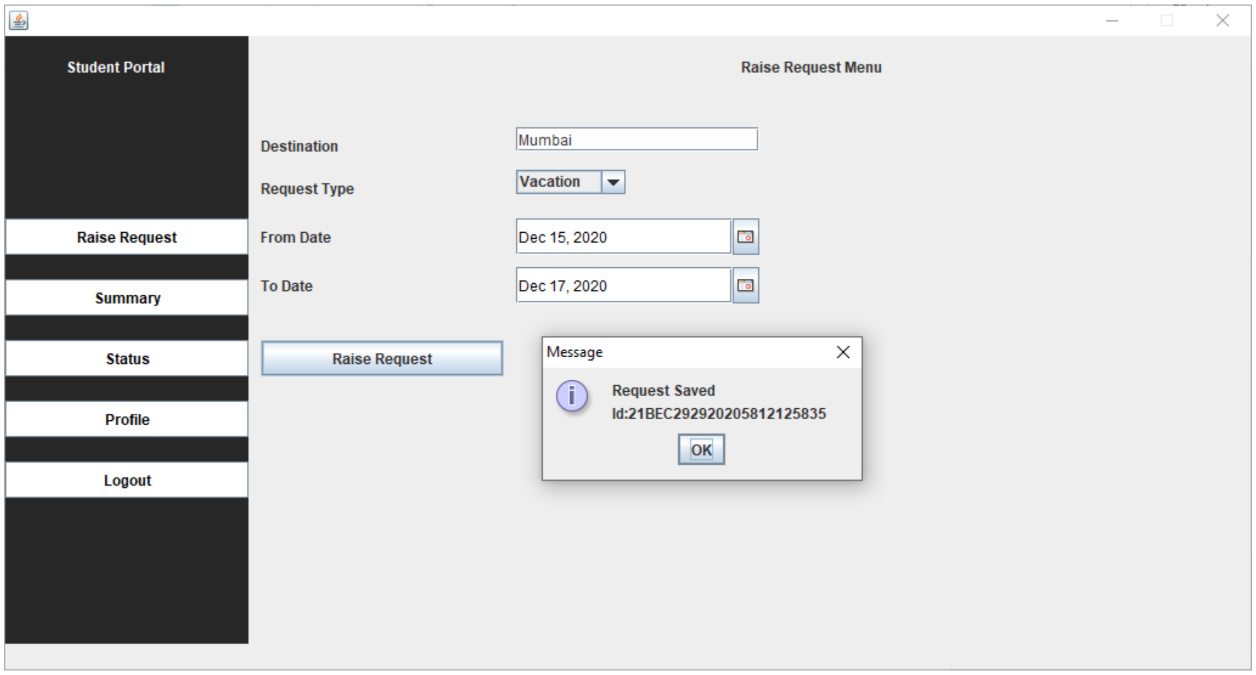
From Date: Dec 19, 2020

To Date: Dec 17, 2020

Message

Fill complete form

Request Raised



Destination: Mumbai

Request Type: Vacation

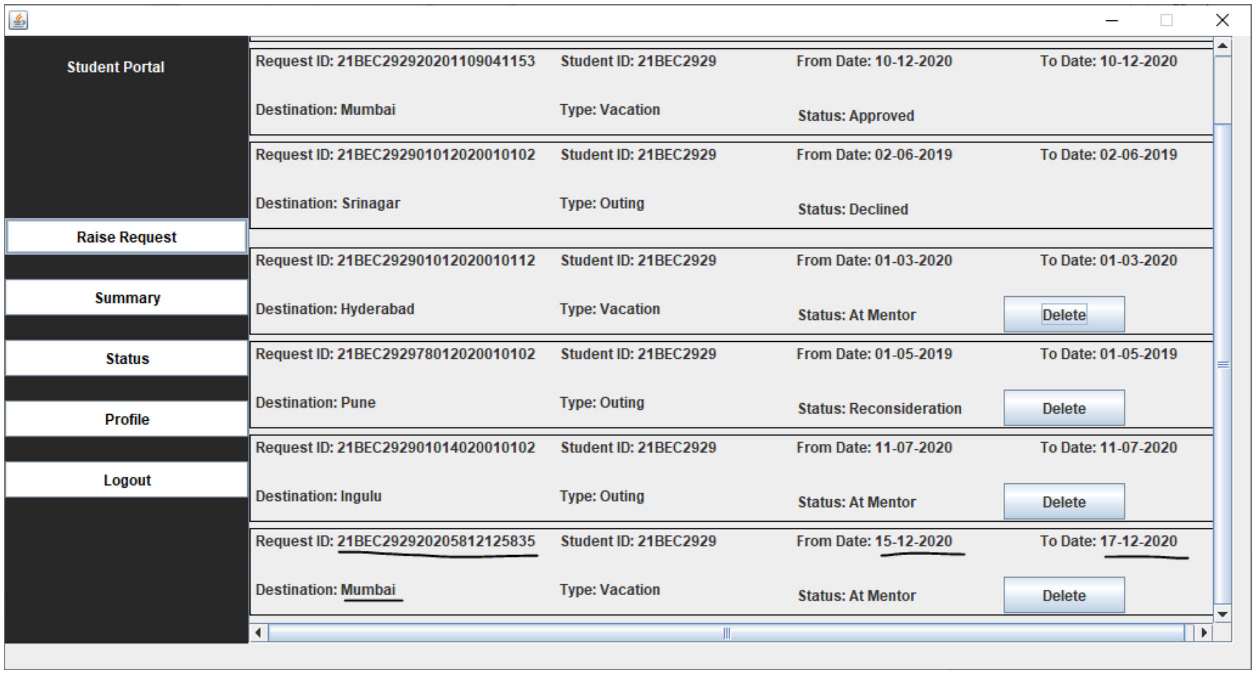
From Date: Dec 15, 2020

To Date: Dec 17, 2020

Message

Request Saved
Id:21BEC29292020581212 5835

We can see the same request in status menu



Student Portal

Raise Request

Summary

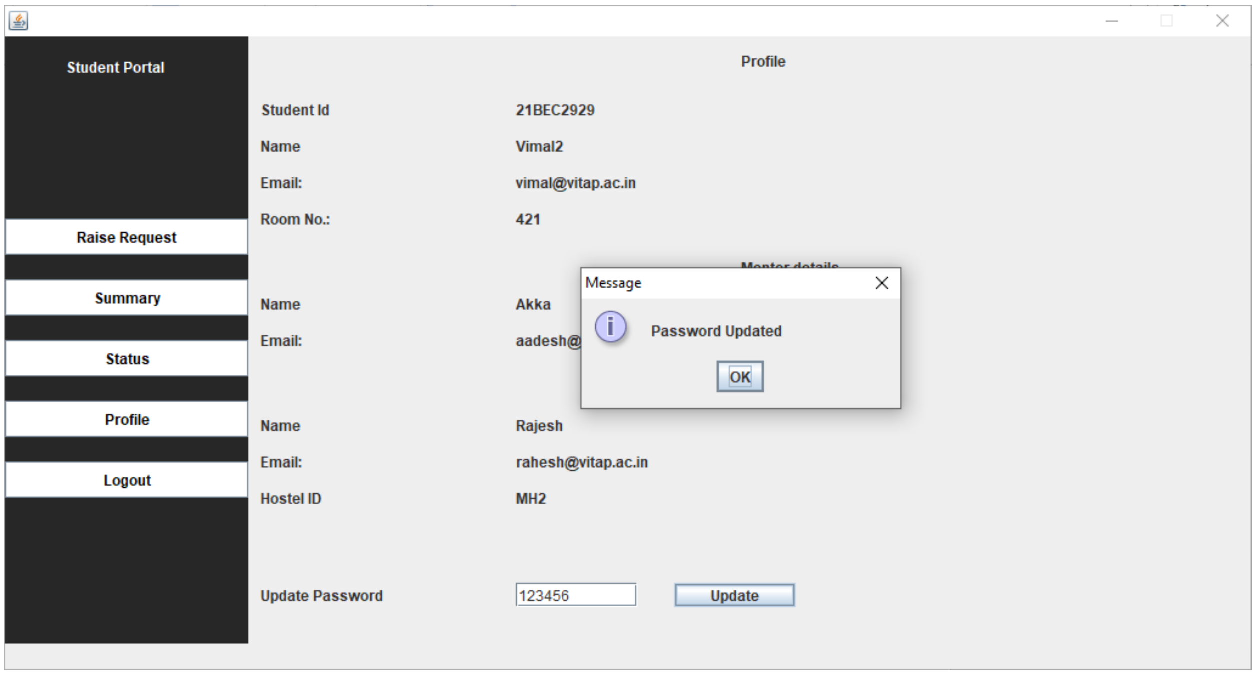
Status

Profile

Logout

Request ID: 21BEC292920201109041153	Student ID: 21BEC2929	From Date: 10-12-2020	To Date: 10-12-2020	
Destination: Mumbai	Type: Vacation	Status: Approved		
Request ID: 21BEC292901012020010102	Student ID: 21BEC2929	From Date: 02-06-2019	To Date: 02-06-2019	
Destination: Srinagar	Type: Outing	Status: Declined		
Request ID: 21BEC292901012020010112	Student ID: 21BEC2929	From Date: 01-03-2020	To Date: 01-03-2020	
Destination: Hyderabad	Type: Vacation	Status: At Mentor	Delete	
Request ID: 21BEC292978012020010102	Student ID: 21BEC2929	From Date: 01-05-2019	To Date: 01-05-2019	
Destination: Pune	Type: Outing	Status: Reconsideration	Delete	
Request ID: 21BEC292901014020010102	Student ID: 21BEC2929	From Date: 11-07-2020	To Date: 11-07-2020	
Destination: Ingulu	Type: Outing	Status: At Mentor	Delete	
Request ID: 21BEC292920205812125835	Student ID: 21BEC2929	From Date: 15-12-2020	To Date: 17-12-2020	
Destination: Mumbai	Type: Vacation	Status: At Mentor	Delete	

Password Update



Student Portal

Raise Request

Summary

Status

Profile

Logout

Profile

Student Id	21BEC2929
Name	Vimal2
Email:	vimal@vitap.ac.in
Room No.:	421

Mentor details

Name	Akka
Email:	aadesh@
Name	Rajesh
Email:	rahesh@vitap.ac.in
Hostel ID	MH2

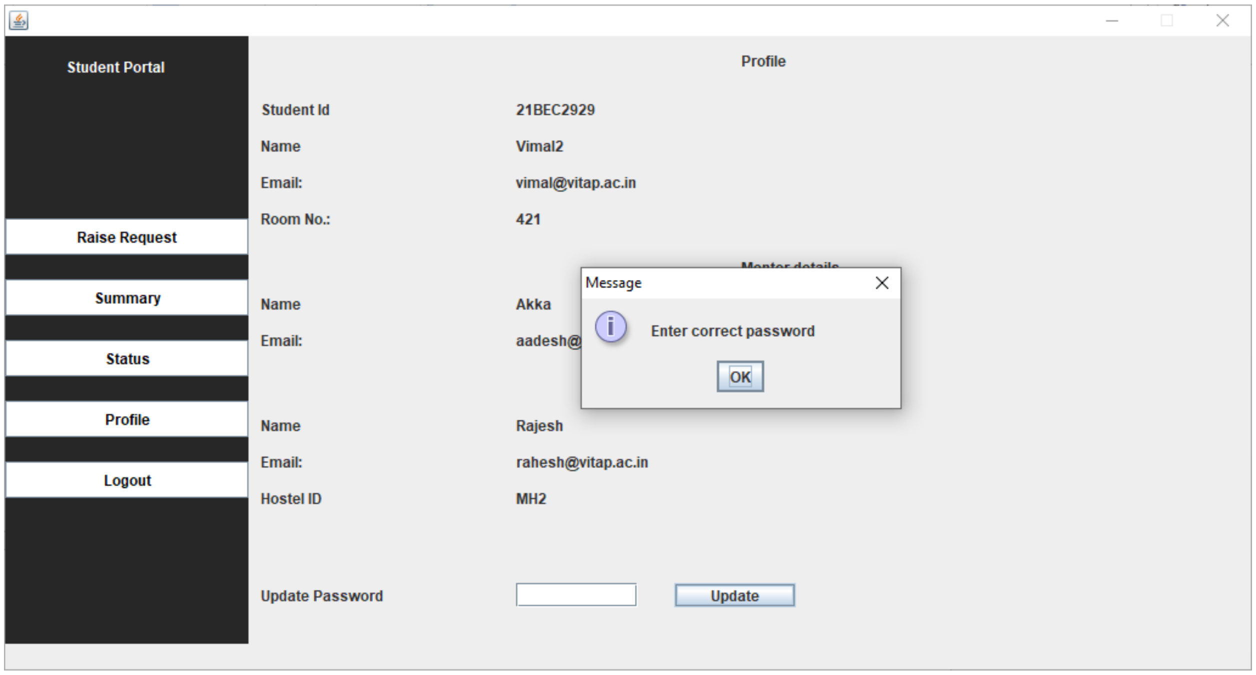
Update Password 123456 Update

Message

Password Updated

OK

When field is blank and update button is clicked


Student Portal

Raise Request

Summary

Status

Profile

Logout

Profile

Student Id 21BEC2929

Name Vimal2

Email: vimal@vitap.ac.in

Room No.: 421

Name Akka

Email: aadesh@

Name Rajesh

Email: rahesh@vitap.ac.in

Hostel ID MH2

Update Password

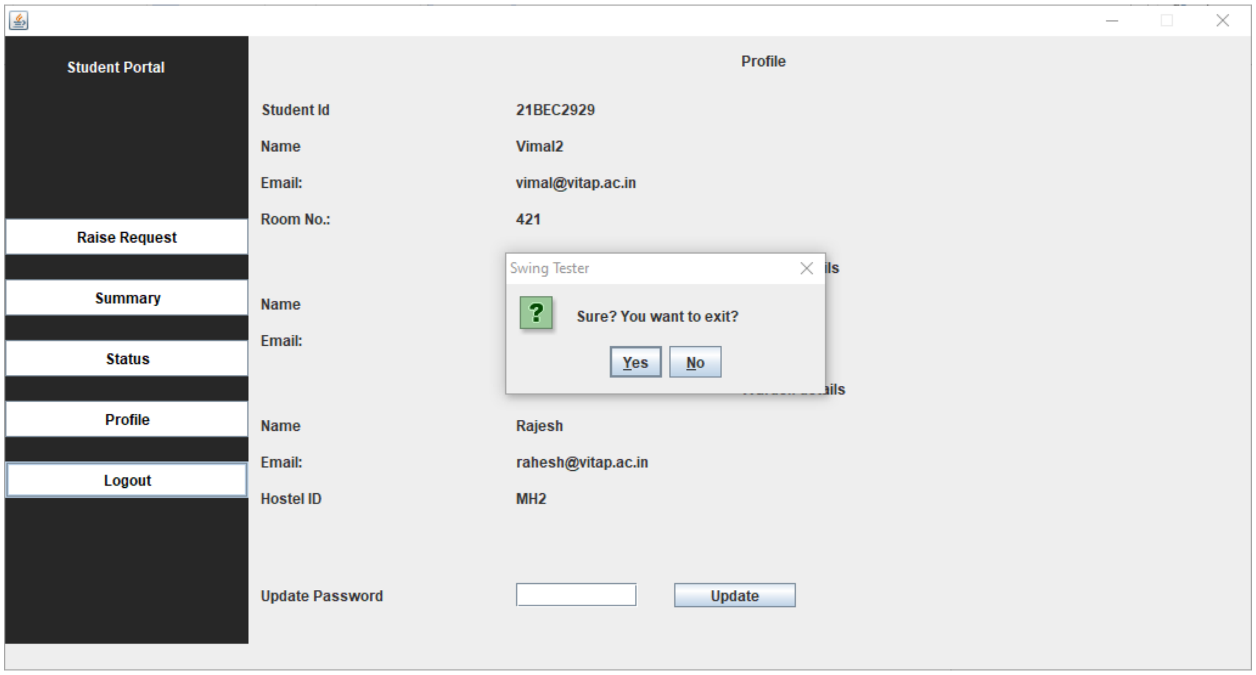
Update

Message

Enter correct password

OK

Logout


Student Portal

Raise Request

Summary

Status

Profile

Logout

Profile

Student Id 21BEC2929

Name Vimal2

Email: vimal@vitap.ac.in

Room No.: 421

Name

Email:

Name Rajesh

Email: rahesh@vitap.ac.in

Hostel ID MH2

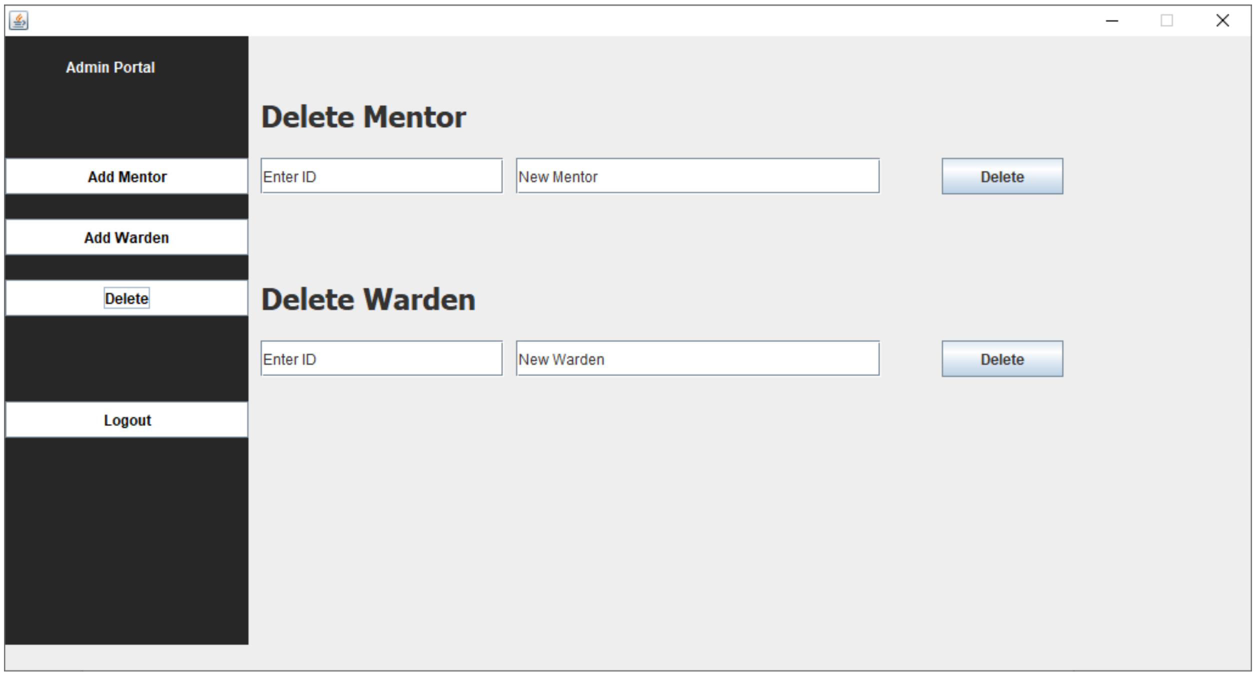
Update Password

Update

Swing Tester

Sure? You want to exit?

Yes No


Admin Portal
Add Mentor
Add Warden
Delete
Logout
Delete Mentor
Enter ID
New Mentor
Delete
Delete Warden
Enter ID
New Warden
Delete

CONCLUSION & FUTURE ENHANCEMENT

- The admin can flexibly interact with all the data flowing through the software and its individual users.
- The student can easily raise and delete active requests along with being able to view all the past request raised by him/her.
- The mentor can view and verify all the request from all his/her mentees and can also view their profiles and other details.
- The warden can view and verify all the request from the students staying in the particular hostel under his/her management while also being able to send the request back to mentor for reconsideration with an internal comment.
- The Guard can view all the request which have been accepted completely and can enter the in-time and out-time.

Future Enhancement

- Addition of the parent module.
- Linking the software to the existing college portal.

Bibliography

Websites:

https://docs.oracle.com/javase/tutorial/getStarted/index.html

https://docs.oracle.com/cd/E11882_01/server.112/e40540/sqllangu.htm#CNCPT015

https://www.oracletutorial.com/oracle-administration/

https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/