**SSN College of Engineering**

UCS1617- Mini Project Lab

Permission and Request Online System

****

**Members**:

Abhishek Narayan - 195001006

Advaith N Narayanan - 195001011

A Anirudh - 195001015

Digant Mehul Gandhi – 195001029

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No.** | **Date of Expt.** | **Title of the Experiment** | **Page No.** |
| 1 | 15-03-2022 | Problem Statement | 3 |
| 2 | 06-04-2022 | Software Requirement Specification | 5 |
| 3 | 11-04-2022 | Use Case Diagram | 12 |
| 4 | 13-04-2022 | Domain Model and Class Diagram | 17 |
| 5 | 23-04-2022 | Interaction Diagram | 24 |
| 6 | 30-04-2022 | State Machine and Activity Diagram | 28 |
| 7 | 08-05-2022 | Implementation | 35 |
| 8 | 08-05-2022 | Test Cases - Permission and Request Online System | 75 |

**Problem Statement**

**Ex.No:** 1

**Date:** 15-03-2022

**Problem Statement:**  
The PROS helps students and professors with issues involving permission and other requests.

The students face issues in the following areas:

* Acquiring permission to leave the college using hostel leave forms for hostellers and for day scholars to leave early. Female students need to apply leave forms one day prior.
* Acquiring OD from professors during college hours especially during events/competitions involves many students queueing, resulting in long wait times and inconvenience.
* Acquiring permission for leave of absence from professors without interrupting classes in case of an emergency situations requiring immediate leave.
* Acquiring information on where to collect documents such as 10th,12th certificates for passports, etc. and Transcripts for intern and other off campus opportunities
* Acquiring information on general aspects such as where to go for a particular class, when to come for getting record signed, whom to contact for in case of any events among others.

The professors and hostel wardens face issues in the following areas:

* Inconvenience as they may not be available and/or free when the student requests OD/Leave-form.
* Inconvenience to keep track of the students who are taking OD/leave in case of events/competitions involving multiple students.
* Occasions like public holidays see many hostellers waiting in line for hostel wardens and this long-drawn process in inconvenient for wardens.
* Staffs/faculties may not be free when student arrives to ask for documents, etc. They must reschedule quickly considering their hectic schedule
* Professors face issues when students have genuine queries, but they don’t have the time or answer at that moment to answer them. Sometimes they might get the answer/time and find it hard to contact the student who asked the question.

**Background:**

SSN College of Engineering comprises over 4000 students and several hundred faculty, staff, and admin. SSN offers students an option to be a day scholar or a hosteller. With many students opting to be hostellers, the process of getting Leave-form from hostel wardens becomes tedious, specifically during festivals, long weekends and during exam holidays.

Likewise, during events or competitions, faculty members need to give permission to several students for OD and this process is tedious and time consuming for faculty and students. In case of emergency, if the faculty is unavailable, it is very difficult for the student to get permission to leave the college.

The process for collecting documents such as 10th and/or 12th certificates is lengthy where students do not have a clear idea on who to meet or where to go. This makes the process of getting the required documents cumbersome. Also, there is the issue of the faculty being free and the documents being available to hand over to the student, else the student is asked to visit some other day to collect the same. Thus, it makes it hard for both the faculty and the student.

Students find it hard to contact any faculty/warden when they have some general queries or doubts. Either they do not know who to contact or they do not know when they would be free or even where they would be. Mails sent using the official SSN email id often go into spam, making the process even challenging.

We believe that the extensive time and effort that goes into requesting for hostel leave-forms, OD permission, Official documents among others, can be simplified and made more convenient with the help of a well-designed and effective web application that provides an interface to students for requesting permission and documents from hostel wardens, professors, and other office members. Similarly, the hostel wardens and professors can choose to accept/reject the students requests on a case-by-case basis without any paperwork, just by a single click.

**Our Methodology:**

We propose to create a log-in portal for students and faculty/hostel admins entering their information; each having their own interfaces.

* **Student Interface:**

Students have a list of options to choose from a fixed set of request/permission that they can ask for. They can upload necessary information(medical certificate, participation proof, reasons to leave hostel) and send it to the appropriate authority for approval. If the student request is approved, he/she receives a message with necessary details and it can be verified for authenticity while leaving the college by simply showing the id card

* **Professor/Hostel Warden Interface:**

The professor/warden will receive a notification of the request raised from student and they can go through the uploaded information and chat with the student before accepting/rejecting the request.

* **Requesting of Original Documents:**

The student selects the request for the document through the student interface as mentioned above. Depending on the document type, the receivers address is automatically added(CSE department/ Exam cell for transcripts and CDC/SSN office for 10th and 12th documents).Once the students describe the reason and sends the request, he/she waits for reply from the faculty. Meanwhile in the faculty interface, they can accept/deny the request and once they accept, they can notify the time and place for the student to collect the necessary documents, thus making it easier for both the parties.

* **Other general queries:**

Apart from the above mentioned, there are a variety of queries that the student may have. The students once clicking on the other queries, selects the receivers address and types his query in the box allocated to it. After clicking send, the request goes to the pending list. Once the request has been answered to, the student gets a notification, and he/she can find their answers there.

* **Authenticity Rating:**

There is a authentication rating for the professors and faculties where they can give ratings of the students behaviour and conduct in the particular request that he/she asked. This will be made public and displayed along with the student’s name the next time he/she asks for a permission.

**Software Requirements Specification**

**Ex.No:** 2

**Date:** 06-04-2022

## Introduction

The process of requesting and receiving permission from Professors/Hostel wardens is currently performed manually where the student needs to personally find the authority in-charge and request for permission. For instance, during long weekends, hundreds of hostellers need to personally seek out the Hostel warden to request for hostel leave form. Due to a large number of students requesting permission in person, the process is very tedious and inefficient. Similarly, in order to get OD permission, students need to request Professors by visiting the staff room. In case the Professor is taking class or not in the staff room, or in case there are many students participating in an event requiring OD, the process yet again becomes tedious for both the teachers and the students. Moreover, the Professors/Hostel wardens need to verify the students' reason for requesting permission to identify legitimate requests and give permission to only those students with legitimate reasons. This takes time and further, if the number of requests is more, becomes harder to track which students received permission.

We have decided to investigate the use of a Permission and Request Online System. This system would be used by students, professors and hostel wardens of SSN College of Engineering to check requests raised by students with any support documents, and decide to accept or reject the request.. The purpose of this document is to analyze and elaborate on the high-level needs and features of the Permission and Request Online System**.** The details of what all are the needs of the Permission and Request Online Systemand if it fulfills these needs are detailed in the use-case and supplementary specifications.

#### Purpose

SSN College of Engineering comprises over 4000 students and several hundred faculty, staff and admin. SSN offers students an option to be a day scholar or a hosteller. With many students opting to be hostellers, the process of getting Leave-form from hostel wardens becomes tedious, specifically during festivals, long weekends and during exam holidays.

Likewise, during events or competitions, faculty members need to give permission to several students for OD and this process is tedious and time consuming for faculty and students. In case of emergency, if the faculty is unavailable, it is very difficult for the student to get permission to leave the college.

We believe that the extensive time and effort that goes into requesting for hostel leave-forms, OD permission among others, can be simplified and made more convenient with the help of a well designed and effective web application that provides an interface to students for requesting permission from hostel wardens and professors. Similarly, the hostel wardens and professors can choose to accept/reject the students' requests on a case by case basis without any paperwork, just by a single click.

#### Scope

The Software Requirements Specification captures all the requirements in a single document. The Permission and Request Online System that is to be developed is supposed to have the following features.

* + - The system provides the students with the ability to raise requests and professors and hostel wardens to view and choose to accept or decline the requests.
    - The system provides logon facility to the users- students, professors and hostel wardens..
    - The system provides the users with the option to check their account and/or change their options like password of the account whenever needed all through the day.
    - The system allows the users to raise requests 24 hours a day and all through the semester.
    - The system lets the professors and hostel wardens to check which students have raised requests and view supporting documents and then decide to accept or decline requests.
    - The system updates the students' request history as and when their requests get accepted or declined..
    - The system also has an option to add new students to the system as and when they get admitted into the college..

The features that are described in this document are used in the future phases of the software development cycle. The features described here meet the needs of all the users. The success criteria for the system is based on the level up to which the features described in this document are implemented in the system.

#### Definitions, Acronyms and Abbreviations

* + - PROS - Permission and Request Online System

#### Overview

The SRS will provide a detailed description of the Permission and Request System. This document will provide the outline of the requirements, overview of the characteristics and constraints of the system.

* + 1. ***Section 2:*** This section of the SRS will provide the general factors that affect the product and its requirements. It provides the background for those requirements. The items such as product perspective, product function, user characteristics, constraints, assumptions and dependencies and requirements subsets are described in this section.
    2. ***Section 3:*** This section of SRS contains all the software requirements mentioned in section 2 in detail sufficient enough to enable designers to design the system to satisfy the requirements and testers to test if the system satisfies those requirements.

## Overall Description

#### Product Perspective

The Permission and Request Online System(PROS) is a web application that will be used by students, faculty members, hostel warden and other staff members of SSN College of Engineering to increase the efficiency of the current approach used in acquiring permissions and requests. The Permission and Request Online System benefits students and staff greatly by decreasing the time and effort needed to request and grant permission. The PROS provides multiple options for the students to choose their requests ranging from leave forms for hostellers, On Duty permissions, Early leave permissions for day scholars and other general requests. The professors/warden/staffs can choose to accept or reject the request with just a single click.

The application to be developed has interactions with Students, faculty members and hostel wardens.

The product has to interact with the Internet.

**Product Functions**

The Permission and Request Online System(PROS) provides various permissions and requests. The Product functions are more or less the same as described in the product perspective. The functions of the system include the system providing different type of services based on the type of users [Student/Faculty].

* + Provisions for the students to ask for permission if all the other required rules hold good.
  + The member is given a provision to check his account information and change the account information any time in the given valid period.
  + The members are provided with the various permissions as a list and can choose any one among them
  + The faculty is prompted with the request along with details of the student's permission history.
  + The faculty is provided with interfaces to accept/reject and also chat with the user before giving permission.
  + The students when receives the permission, a QR code is generated, which can be used by the students
  + The system uses the University information security requirements to provide the login facility to the users.
* **User characteristics**

The users of the system are students, faculties of the university and the administrators who maintain the system. The students and the faculties are assumed to have basic knowledge of the computers and Internet browsing. The administrators of the system should have more knowledge of the internals of the system and are able to rectify the small problems that may arise due to disk crashes, power failures and other catastrophes to maintain the system. The proper user interface, users manual, online help and the guide to install and maintain the system must be sufficient to educate the users on how to use the system without any problems.

### Constraints

* + The information of all the users must be stored in a database that is accessible by the system.
  + The university information security system must be compatible with Internet applications.
  + The Permission and Request system runs all 24 hours a day.
  + The users will be able access the Permission and Request System from any computer that has Internet browsing capabilities and an Internet connection.
  + The users must have their correct usernames and passwords to enter into the System

### Assumptions and dependencies

* + The users have sufficient knowledge of computers.
  + The College/University computer should have Internet connection and Internet server capabilities.
  + The users know the English language, as the user interface will be provided in English
  + The product can access the university student database

## Specific Requirements

This section describes in detail all the functional requirements.

#### Functionality

* + 1. *Logon Capabilities*

The system shall provide the students and the faculties with logon capabilities.

* + 1. *Mobile Devices*

The Permission and Requests System is also supported on mobile devices such as cell phones.

#### Usability

* + - The system shall allow the users to access the system from the Internet using HTML or its derivative technologies. The system uses a web browser as an interface.
    - Since all users are familiar with the general usage of browsers, no specific training is required.
    - The system is user friendly and self-explanatory.
    - The system is secured to avoid forging

#### Reliability

* + 1. The system has to be very reliable due to the importance of data and the damages incorrect or incomplete data can do *Availability*

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

* + 1. *Mean Time Between Failures (MTBF)*

The system will be developed in such a way that it ***may*** fail once in three months.

* + 1. *Mean Time to Repair (MTTR)*

Even if the system fails, the system will be recovered back up within an hour or less.

* + 1. *Accuracy*

The accuracy of the system is limited by the accuracy of the speed at which the students and the faculties use the system.

* + 1. *Maximum Bugs or Defect Rate*

Not specified.

* + 1. *Access Reliability*

The system shall provide 100% access reliability.

#### Performance

* + 1. *Response Time*

The Splash Page or Information page should be able to be downloaded within a minute using a 56K modem. The information is refreshed every two minutes. The access time for a mobile device should be less than a minute. The system shall respond to the member in not less than two seconds from the time of the request submission. The system shall be allowed to take more time when doing large processing jobs.

* + 1. *Administrator/Faculty Response*

The system shall take as less time as possible to provide service to the administrator or the faculty.

* + 1. *Throughput*

The number of transactions is directly dependent on the number of users, the users may be the Administrator, Faculties and also the students of the college for asking permissions, making requests, and requesting transcripts.

* + 1. *Capacity*

The system is capable of handling 100 users at a time.

#### Supportability

The system designers shall take into consideration the following supportability and technical limitations.

* + 1. *Internet Protocols*

The system shall comply with the TCP/IP protocol standards and shall be designed accordingly.

* + 1. *Information Security Requirement*
    2. The system shall support the UHCL information security requirements and use the same standard as the UHCL information security requirements. *Billing System Data Compatibility*

The member balance amount that will be calculated and sent to the billing system shall be compatible with the data types and design constraints of the billing system.

* + 1. *Maintenance*

The maintenance of the system shall be done as per the maintenance contract.

* + 1. *Standards*

The coding standards and naming conventions will be as per the American standards.

#### Design Constraints

* + 1. *Software Language Used*

The languages that shall be used for coding the Permission and Request System are Active Server Pages (ASP), Java Servlets, Python, Java Server Pages (JSP), HTML and JavaScript. For working on the coding phase of the Online Library System, the Internet Information Services (IIS) Server needs to be installed.

* + 1. *Development Tools*

Will make use of the available Java Development Tool kits for working with Java Beans and Java Server Pages. Also will make use of the online references available for developing programs in ASP, HTML and the one scripting languages, JavaScript.

* + 1. *Class Libraries*

Will make use of the existing Java libraries available for JSP and Servlets. Also we need to develop some new libraries for the web-based application. Also will develop new programs using ASP and scripting languages.

#### On-line User Documentation and Help System Requirements

Online help is provided for each of the features available with the Permissions and Requests System. All the applications provide an on-line help system to assist the user. The nature of these systems is unique to application development as they combine aspects of programming (hyperlinks, etc) with aspects of technical writing (organization, presentation). Online help is provided for each and every feature provided by the system.

The User Manual describes the use of the system to Faculties and Students. It describes the use of the system on mobile systems. The user manual should be available as online help.

A ReadMe file is typically included as a standard component. The Read Me includes a “What’s New With This Release” section, and a discussion of compatibility issues with earlier releases. Most users also appreciate documentation defining any known bugs and workarounds in the ReadMe file.

#### Purchased Components

#### The System Administrator will need to purchase the license for IIS Server. Mostly it is available with Windows Environment. So the system need not purchase any licensing products. Interfaces

#### Interfaces

* + 1. *User Interfaces*

Will make use of the existing Web Browsers such as Microsoft Internet Explorer, Mozilla Firefox or Google Chrome.

**Sample Interface:**

For sample interface, click on the below links and explore different tabs present

<https://abhisheknarayan190.wixsite.com/pros>  **->Student Interface**

<https://abhisheknarayan190.wixsite.com/my-site-2> **->Faculty Interface**

* + 1. *Hardware Interfaces*

The existing Local Area Network (LAN) will be used for collecting data from the users and also for updating the database of the system.

* + 1. *Software Interfaces*

A Scanner will be used with the server to prevent duplicacy of the system.

* + 1. *Communications Interfaces*

The Permission and Requests System will be connected to the World Wide Web.

#### Licensing Requirements

The usage is restricted to only Sri Siva Subramaniya Nadar College of Engineering and who is purchasing the Permission and Request System from the college and signs the maintenance contract.

#### Legal, Copyright, and Other Notices

Permission and Requests System is a trademark of Abhishek Narayanan, Adavith N Narayan, A Anirudh and Digant Mehul Gandhi and cannot be used without its consent.

#### Applicable Standards

The ISO/IEC 6592 guidelines for the documentation of computer based application systems will be followed.

## Supporting Information

The use-case storyboards or the user-interface prototypes are not available. The appendices are not to be considered as part of the requirements.

**Use Case Description and Diagram**

**Ex.No:** 3

**Date:** 11-04-2022

**Scope:** Permission and Request Online System

**Level:** user goal

**Primary Actor:** Student

**Stakeholders and Interests:**

- Student: Wants fast and easy grant of permissions, as current procedure is too tedious. Wants an easy to use User Interface to enter their reason and other information for asking permission. Wants to be able to upload supporting documents. Wants to be able to download granted permission for any verification.

- Faculty: Wants easily visible information of students requesting for permissions. Wants supporting proof of request. Wants to be able to check students' request history and rating. Wants to be able to Accept or Decline requests of one or more than one student. Wants to be able to give rating to the student based on the request. Wants to be able to chat with student incase more information is needed.

- Database: Wants to authenticate and remove redundancy in student and faculty login information. Wants to accurately record all incoming requests along with the supporting documents. Wants to track status of Request. Wants to maintain request history and student rating based on request history. Wants automatic and fast update of request information

- College: Wants to collect details of each student and assess student behavior

**Preconditions:** The student is identified and authenticated

**Success Guarantee (or Postconditions):** Requests are saved. Rating is calculated. Database is Updated. Permissions are recorded along with the priority ones. QR Code is generated.

**Main Success Scenario (or Basic Flow):**

1. Student visits the site to request for permissions

2. Student logs in or register into the system

3. Database identifies the student

4. Students choose a permission out of a list of permissions.

5. System asks for student to enter the reason and upload supporting document

6. Faculty gets notification of request

7. Faculty views the request and supporting information and chats with student for validation

8. Faculty accepts or declines the request and gives rating to the student

9. Student receives the notification of request decision along with the QR code

10. Student logs off the system

**Extensions (or Alternate Flows):**

1.At any time, system fails:

To support recovery, and ensure all details and events can be recovered from any step of the scenario.

2.Student enter invalid/forgot login credentials mode:

* System provides the option of entering a registered email address for password reset.
* Student or Faculty reset their old password to a new password by answering the security question
* Student or faculty can search for the mail address and send login link to it

1. If mail Id does not exist:

* System should tell student or faculty to contact administrator

3.Invalid/Not Existing Username is database:

* System signals error and asks student to register and add necessary credentials

4.The permission entered is not wanted

* Student can delete the permission before it gets approved

5. Supporting document exceeds the size limit:

* System restarts the upload service again

6. Timeout is achieved

* The request is turned into a priority request and the notification is sent to the faculty

**Special Requirements:**

1. Touch Screen UI on mobile. Text visible from 1 meter

2. Laptop with touch pad or mouse and keyboard on laptop.

3. Language internationalization on the text displayed

4. QR scanner for verification

**Technology and Data Variations List:**

1. QR verification can be done by using a QR code scanner or or by entering the associated unique ID.
2. The documents uploaded by the students may be in the .docx, .pdf, .jpeg, .png, .tiff, .avi or .mp3 format.
3. The chat between the staff and professor will be encoded in the Unicode character representation standard only.
4. The permission history of each student can be stored in a database. However, it may become necessary to transfer the less recent history of the student to a secondary storage as the number of users on the system increase.

**Frequency of Occurrence:**

The frequency of use of this software could be nearly continuous.

**Miscellaneous/Open Issues:**

1. What are the variations in the storage of different data formats?
2. Explore the database recovery issue.
3. What customization is needed for different colleges?
4. Can users log in on more than one device?
5. Explore the document storage techniques.

**All Actors present and their type:**

|  |  |
| --- | --- |
| **Actor** | **Type** |
| Student | Primary |
| Faculty/Warden | Secondary |
| Database | Secondary |

**Use Case Diagrams:**

There are a total of 3 use case diagrams. One denotes the major flow of the System, the other 2 denotes the details of the sub use cases present.

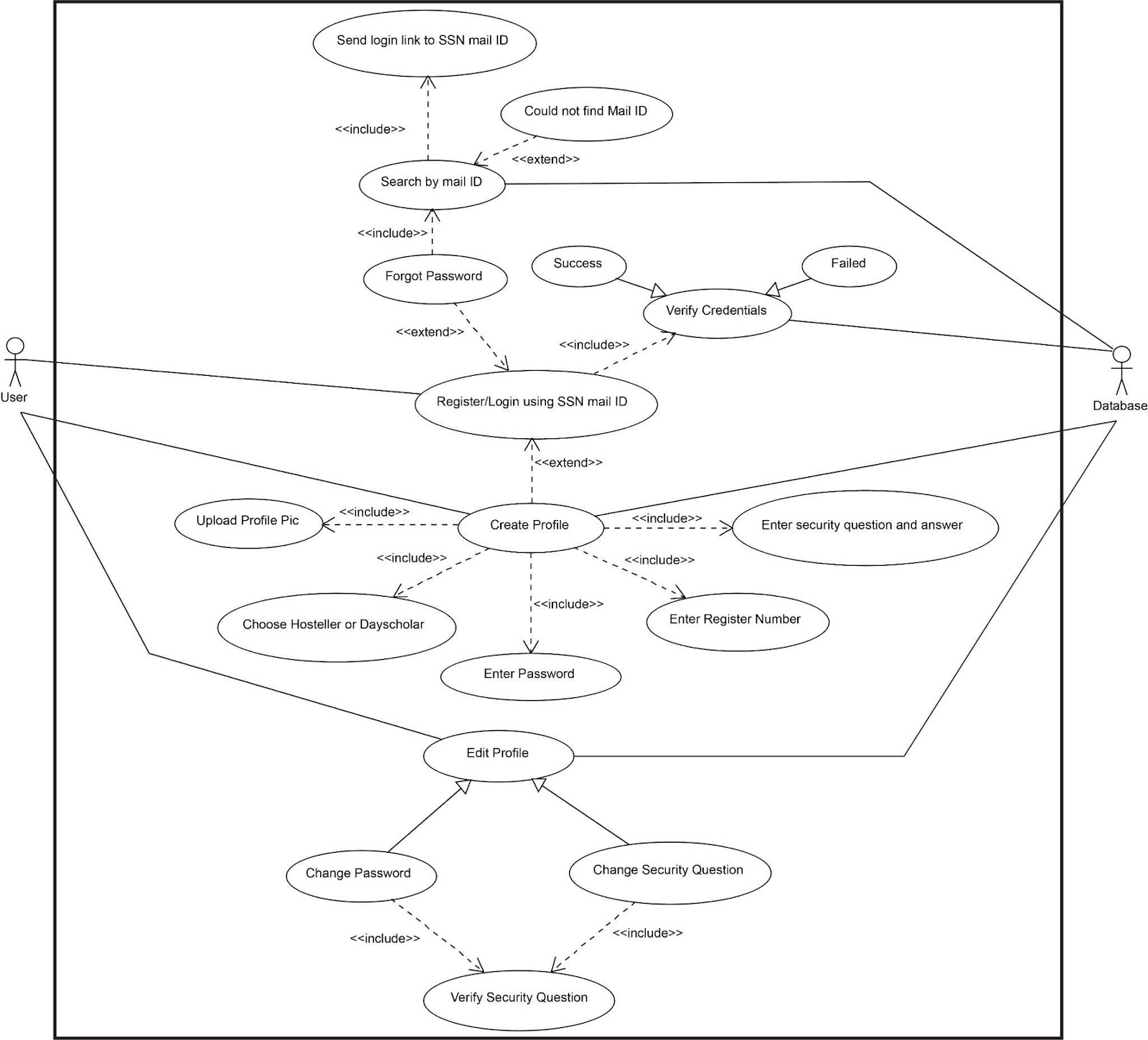
**Use Case Diagram 1:**



**Use Case Diagram 2:**



**Use Case Diagram 3:**



**Domain Model and Class Diagram**

**Ex.No:** 4

**Date:** 13-04-2022

**Aim:**

To identify the classes and their associations and draw domain model and class diagram for Permission and Requests Online System.

**Notations for Domain model:**

|  |  |  |
| --- | --- | --- |
| **Notation** | **Type** | **Explanation** |
|  | Conceptual Class | An object in the problem domain |
|  | Association | Relationship between objects or the classes |
| 1..\* 1 | Multiplicity of the Role | Defines how many instances of a class A can be associated with one instance of a class B |
|  | Generalization | The child class inherits the characteristics of base class |

**Notation for Class diagram:**

|  |  |  |
| --- | --- | --- |
| **Notation** | **Type** | **Explanation** |
|  | Conceptual Class | An object in the problem domain along with its attributes |
|  | Aggregation | It is a relationship between two classes where one class is a part of another where one class is meaningful even without the aggregate |
|  | Composition | It is a relationship between two classes where one class is a part of another where one class is not meaningful without the other |
|  | Generalization | The child class inherits the characteristics of base class |
|  | Association | Relationship between objects or the classes |
| 1..\* 1 | Multiplicity of the Role | Defines how many instances of a class A can be associated with one instance of a class B |

**Identification of Classes:**

1. **Identification of Conceptual Classes:**

|  |  |
| --- | --- |
| **Conceptual class category** | **Conceptual Class** |
| Physical or Tangible object | - |
| Specifications | 1. Student Description 2. Staff Description |
| Catalogs | Student History |
| Roles of People | 1. Student 2. Staff |
| Computer systems | Permission and Request System |
| Events | Requesting |

1. **Identification of Noun Phrases:**

**Initial List of Noun Phrases:**

* Permission and Request Online System
* Request system
* Student
* Student ID
* Student description
* Rating
* Accepted requests
* Pending request
* Document
* Messages
* Staff
* Admin
* Faculty
* Staff ID
* Staff description
* Accept request
* Deny request
* Login
* Logout
* Signup
* Credentials
* QR code
* Document Type

**Final List of Noun Phrases:**

1. Permission and Request Online System
2. Student
3. Student Description
4. Rating
5. Accepted Requests
6. Pending Request
7. Documents
8. Messages
9. Staff
10. Staff Description

**Identification of Associations:**

1. **Identification of Association:**

|  |  |  |
| --- | --- | --- |
| **Category** | **Association** | **Between** |
| *A* uses *B* | Used by | System and Student |
| *A* is a member of *B* | Has | System and Staff |
| *A* is known/logged/reported/captured in *B* | Given to | Rating and Student |
| *A* is a description for *B* | Described by | Student and Student Description,  Staff and Staff Description |
| *A* is event related to *B* | Granted | Student and Accepted Requests |
| *A* is event related to *B* | Creates | Student and Pending Request |
| *A* manages *B* | Approves | Staff and Pending Request |
| *A* is a logically contained in B | Contains | Pending Request and Documents |
| *A* is a transaction related to another transaction *B* | Incorporates | Pending Request and Messages |

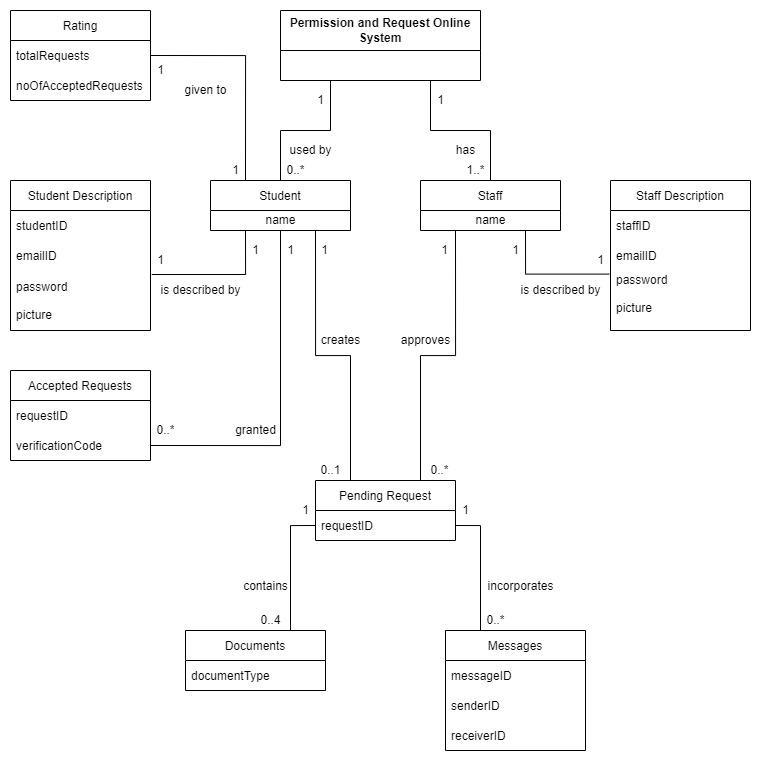
1. **Definition of Association:**

|  |  |
| --- | --- |
| **Association** | **Explanation** |
| Generalization | The Generalization association ("is a") is the relationship between the base class that is named as “superclass” or “parent” and the specific class that is named as “subclass” or “child” |
| Aggregation | An aggregation is a relationship between classes which says one class ‘is a part of’ another class. In aggregation, the part (constituent) is meaningful without the whole (aggregate) |
| Composition | An aggregation is a relationship between classes which says one class ‘is a part of’ another class. In composition, the part (component) is not meaningful without the whole (container) |

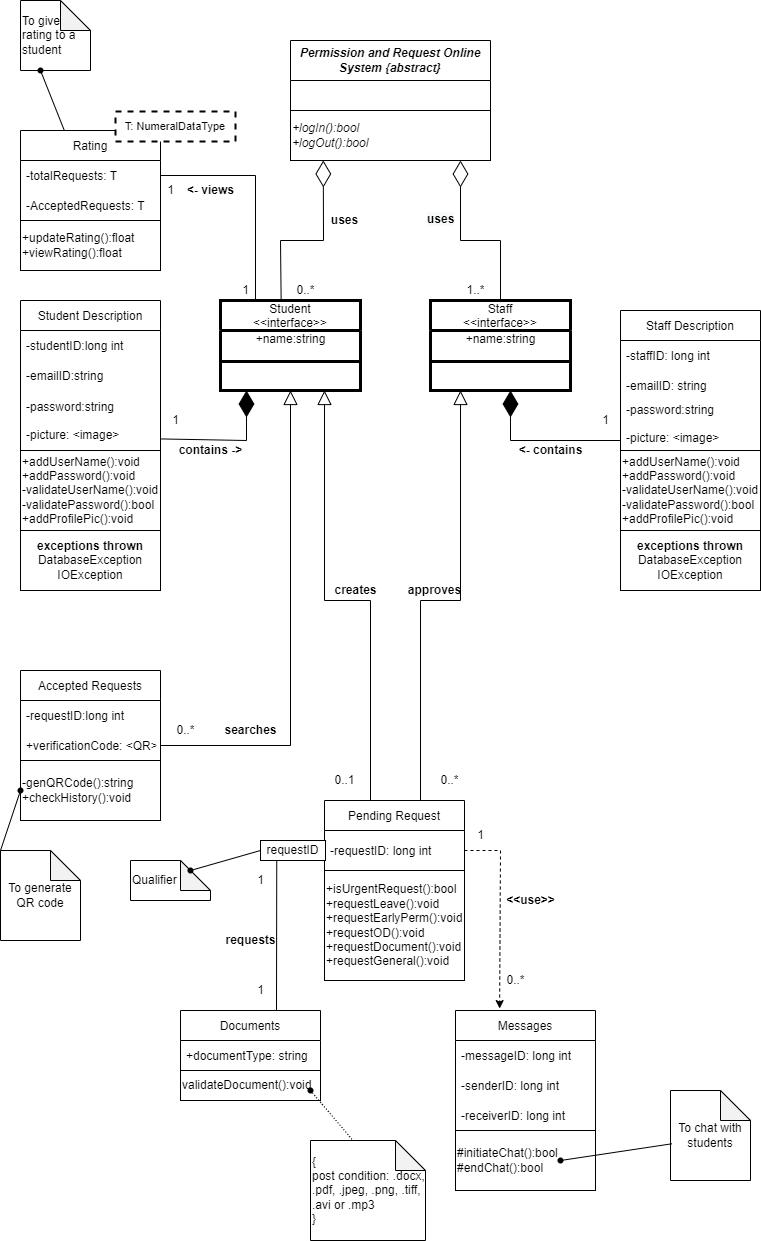
1. **Multiplicity:**

|  |  |
| --- | --- |
| **Relationship** | **Multiplicity** |
| System-Student | One (1) permission and request system is used by zero or more (0..\*) students |
| System-Staff | One (1) permission and request system has one or more (1..\*) staff |
| Rating-Student | One (1) rating is given to one (1) student |
| Student-Student Description | One (1) student is described by one (1) student description |
| Staff-Staff Description | One (1) staff is described by one (1) staff description |
| Student-Accepted Requests | One (1) student is granted permission for zero or more (0..\*) requests |
| Student-Pending Request | One (1) student creates zero or one (0..1) pending request |
| Staff-Pending Request | One (1) staff approves zero or more (0..\*) pending requests |
| Pending Request-Documents | One (1) pending request contains zero to four (0..4) associated documents |
| Pending Request-Messages | One (1) pending request incorporates zero or more (0..\*) messages between staff and students |

**Domain model:**

****

**Class Diagram:**

****

**Documentation:**

Requests can be made by one or more users. Users include students. Users have to enter their details and verify them before making any requests. Users can check their history of requests. Users are also provided with functionality to check their rating as well as make priority requests. Users can also cancel their requests. Each user and their request have a unique identifier. Users must upload a document as proof for making a request. Faculty can approve requests of one or more students. Faculty can initiate chat with the students before approving the request. Accepted requests contain the request ID and the QR code for verification.

**UML Sequence Diagram**

**Ex.No:** 5

**Date:** 23-04-2022

**Aim:**

Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams.

**Prerequisites:** Some basic understanding of UML interaction diagrams

**Input:** Problem Statement

Students of SSN College of Engineering residing in Hostels as well as students who travel a long way to reach college (Day scholars) face various issues due to an overload of manual paperwork, getting authentication from concerned authorities, and hassles involved in getting various documents.

**Output:** UML sequence Diagram

**Apply the following concepts in sequence diagram**

**1. Lifeline Boxes and Lifelines**

Each classifier role has a lifeline as illustrated in the diagram.

**2. Messages:**

Each solid arrow depicting an action has its corresponding meaning mentioned through the message.

**3. Focus of Control and Execution Specification Bars**

The Application has an infinite focus of control as it will always be running, while the other classifier roles have their focus of control using the execution specification bars restricted to the timeline during which they interact with the application.

**4. Illustrating Reply or Returns**

A response message is also present wherever needed to acknowledge the incoming action. This concept illustrates the replies involved in the interactions. There are no returns in our sequence diagram.

**5. Messages to "self" or "this"**

A response to the application is sent and during login and registration. A response to the database is sent when a request is made.

**6. Creation of Instances**

An instance is created depending on whether the type of the user logging in is student or faculty.

**7. Object Lifelines and Object Destruction**

Object lifelines and destruction is implemented for the Student during the “Log In” and “Log Out” process. Similarly, Object lifelines and destruction is implemented for the Faculty during the “Log In” and “Log Out” process. The lifelines are defined by the beginning and end of the two actions.

**8. Diagram Frames in UML Sequence Diagrams**

Our sequence diagram constitutes of various scenarios that are a result of alternate scenarios, Eg: Results of request for permission i.e. accepted or rejected.

**9. Looping**

There is a loop in our diagram that the login process is looped till a successful login is performed or registration is performed.

**10. Conditional Messages**

There are conditional messages as responses to different scenarios involved in validation of user credentials, requests, support documents, etc.

**11. Conditional Messages in UML**

There are conditional messages as responses to different scenarios involved in validation of login process, QR Code if request is accepted, etc.

**12. Mutually Exclusive Conditional Messages**

There are conditional messages as responses to different scenarios involved in validation of Requests using QR Code. These conditional messages are NOT mutually exclusive as only one scenario is possible in our implementation.

**13. Iteration Over a Collection**

NILL (Our sequence diagram doesn’t necessitate the above concept)

**14. Nesting of Frames**

This concept is visible in sequence diagram 1, where Login() and Register() are two scenarios which have an alternative and loop frame inside each scenario

**15. Messages to Classes to Invoke Static (or Class) Methods**

NILL (Our sequence diagram doesn’t necessitate the above concept)

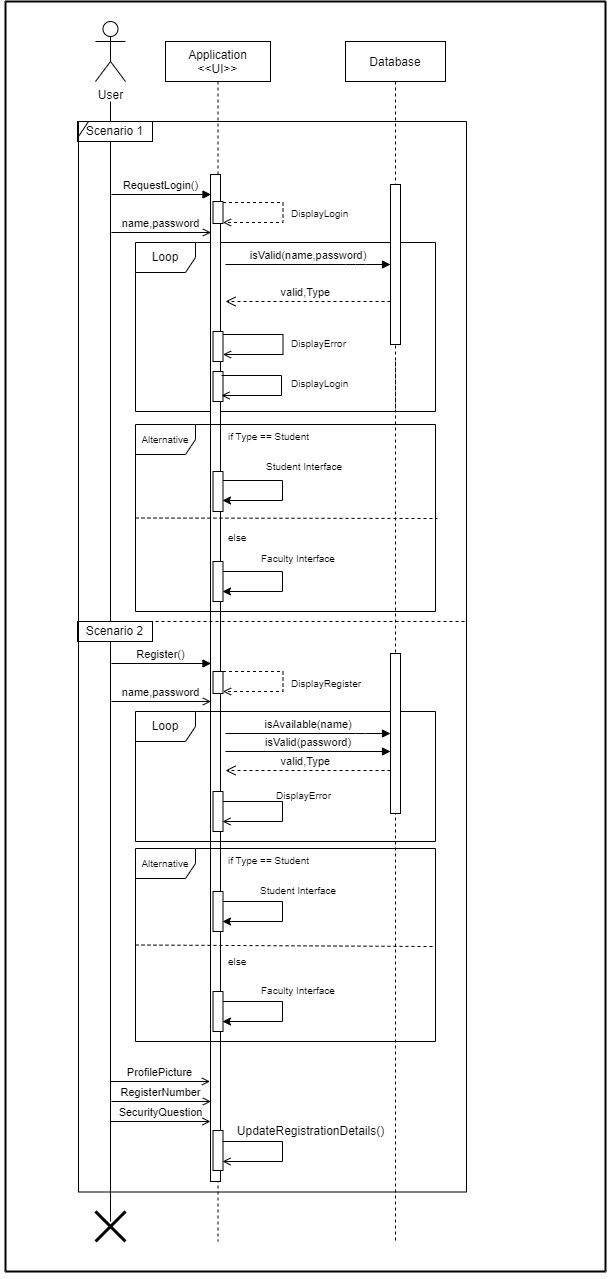
**16. Polymorphic Messages and Cases**

NILL (Our sequence diagram doesn’t necessitate the above concept)

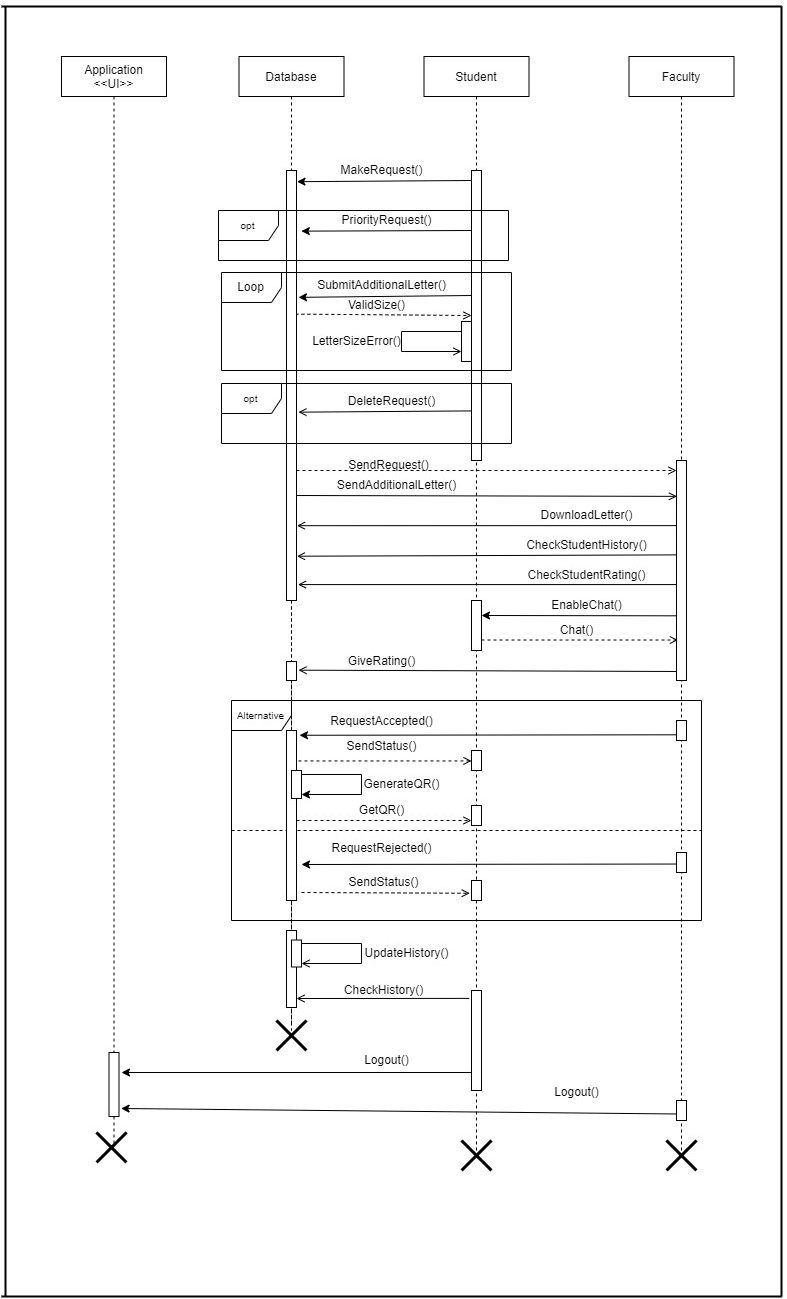
**17. Asynchronous and Synchronous Calls**

Various requests and login/register details include synchronous calls whereas checking student history and student rating require asynchronous calls

**Sequence Diagram 1:**



**Sequence Diagram 2:**



**State Machine Diagram**

**Ex.No:** 6

**Date:** 30-04-2022

**Aim:** Use the identified states, activities, guard conditions etc. to construct a state diagram.

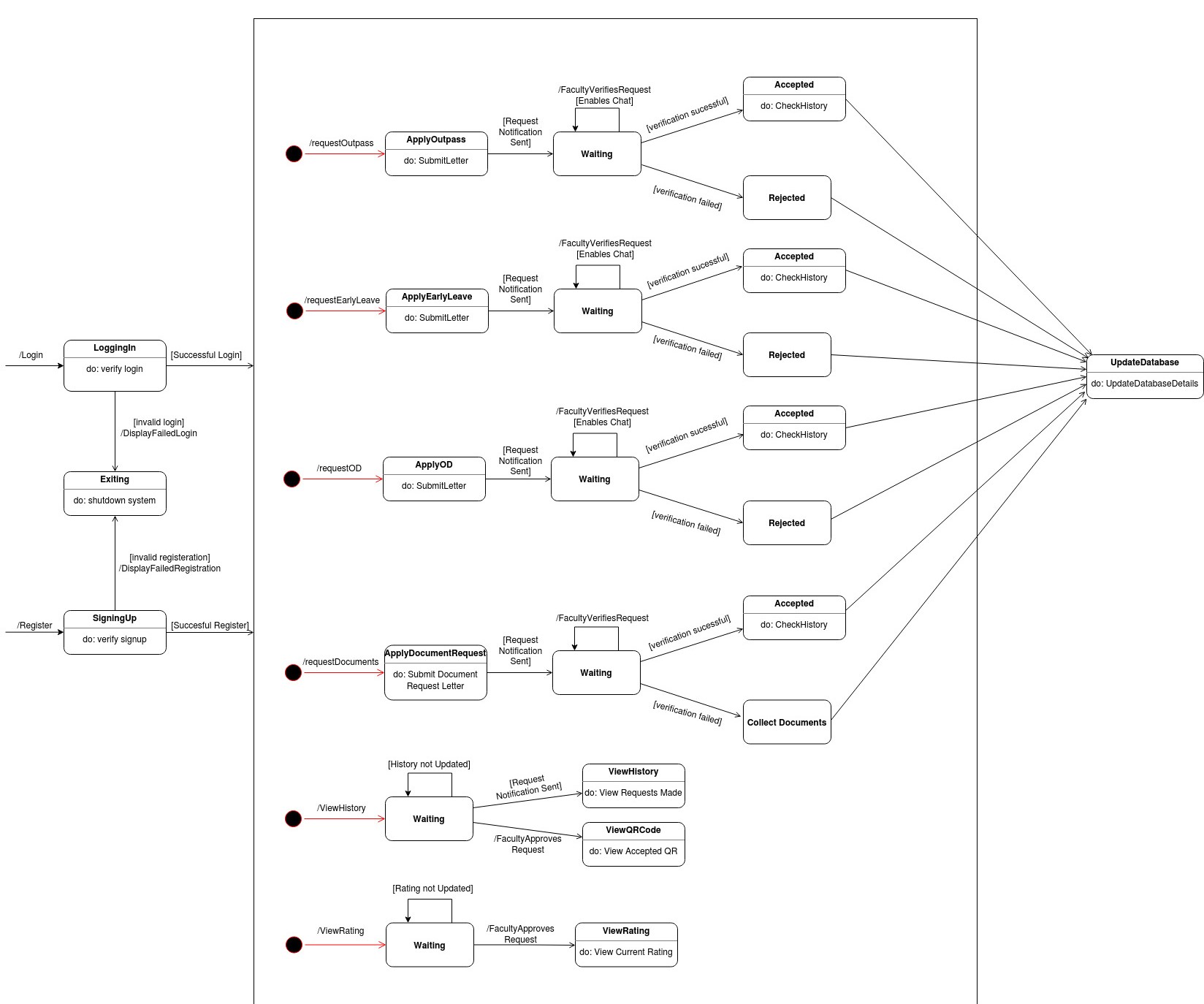
**UML Notations for state machine diagram:**

****

**Identification of States:**

1. **LoggingIn -** The software will be in the state of Login where the credentials are verified through the mentioned activity.
2. **Exiting -** The software will be in the state of Exit where the software will be shutdown through the activity mentioned.
3. **SigningUp -** The software will be in a state of sign up,where the credentials are entered and verified for the first time using the mentioned activity.
4. **ApplyOutpass –** The software will be in a state of ApplyOutpass where a form is filled for hostellers getting outpass and submitted.
5. **ApplyEarlyLeave –** The software will be in a state of ApplyEarlyLeave where a form is filled for day scholars getting outpass and submitted.
6. **ApplyOD –** The software will be in a state of ApplyOD where a form is filled for hostellers and day scholars attending an event for getting attendance and submitted.
7. **ApplyDocumentRequest–** The software will be in a state of ApplyDocumentRequest where a form is filled for students to get their original Documents from college and submitted.
8. **Waiting -** The software will be in the wait state as the warden/Faculties will have to verify the request, chat if necessary and then accept or reject the request for the transition to occur.
9. **Accepted:** The software goes into the state of accept where the request is granted and it is reflected in the student's history.
10. **Rejected:** The software goes into the state of accept where the request is denied and it is reflected in the student's history.
11. **ViewHistory:** The software is in this state till the student/faculty views the student’s previous request histories.
12. **ViewQRCode:** The software is in this state when the student views his qrcode after the request has been accepted.
13. **ViewRating:** The software is in this state when the student/faculty wants to view the rating of the student.
14. **CollectDocuments:** The software is in this state when the request is not sufficient enough to verify and more documents are needed for proof.
15. **UpdateDatabase:** The software is in this state when, the request has been accepted or denied and the rating has been given. It contains the necessary documents and does the activity updatedatabasedetails after every request.

**State Machine Diagram:**



**Explanation:**

The above state machine diagram comprises of various states that the system would be in based on the actions and transitions. The user can first access the system by logging in, which thus takes the system into the LoggingIn system. On validation, the user is allowed to carry out the various use cases that he or she is eligible to utilise. But, incase the user is new, he/she can use the SigningUp state and register his/her account. Once validation is complete, the user is free to do a wide range of activities like the user (student) can get an Outpass, OD, Early Leave Permission, Request Original Documents and also view his ratings, previous requests and QR Code for each request by carrying out the appropriate action. Similarly the faculty/warden have their own interface where they can do a specific set of use cases that they are eligible to use. Each action would take the system to another state that is further changed based on the actions of the user. There are some waiting states present as the transition to the next state is dependent on actions from other users such as the warden/faculty needs to approve/deny the request for student to get confirmation and QR Code. Hence, the above state machine diagram depicts all the possible states at which the system can exist, based on the various actions and activities that are carried out. Finally, there is the database which connects to this super class which contains the data of all the students and faculties.

|  |  |  |
| --- | --- | --- |
| **Notation** | **Type** | **Explanation** |
|  | Start | It initiates the transition |
|  | End | It marks the end of the transitions |
| State | State | Each state in the state machine is represented using this rounded rectangle |
| Decision | Decision | It represents the transitions based on the conditions |
| Swimlane | Swim Lane | It is used for grouping same user’s activities in one column |
|  | Fork | It is used for concurrent executions |

**Activity Diagram**

**Ex.No:** 6

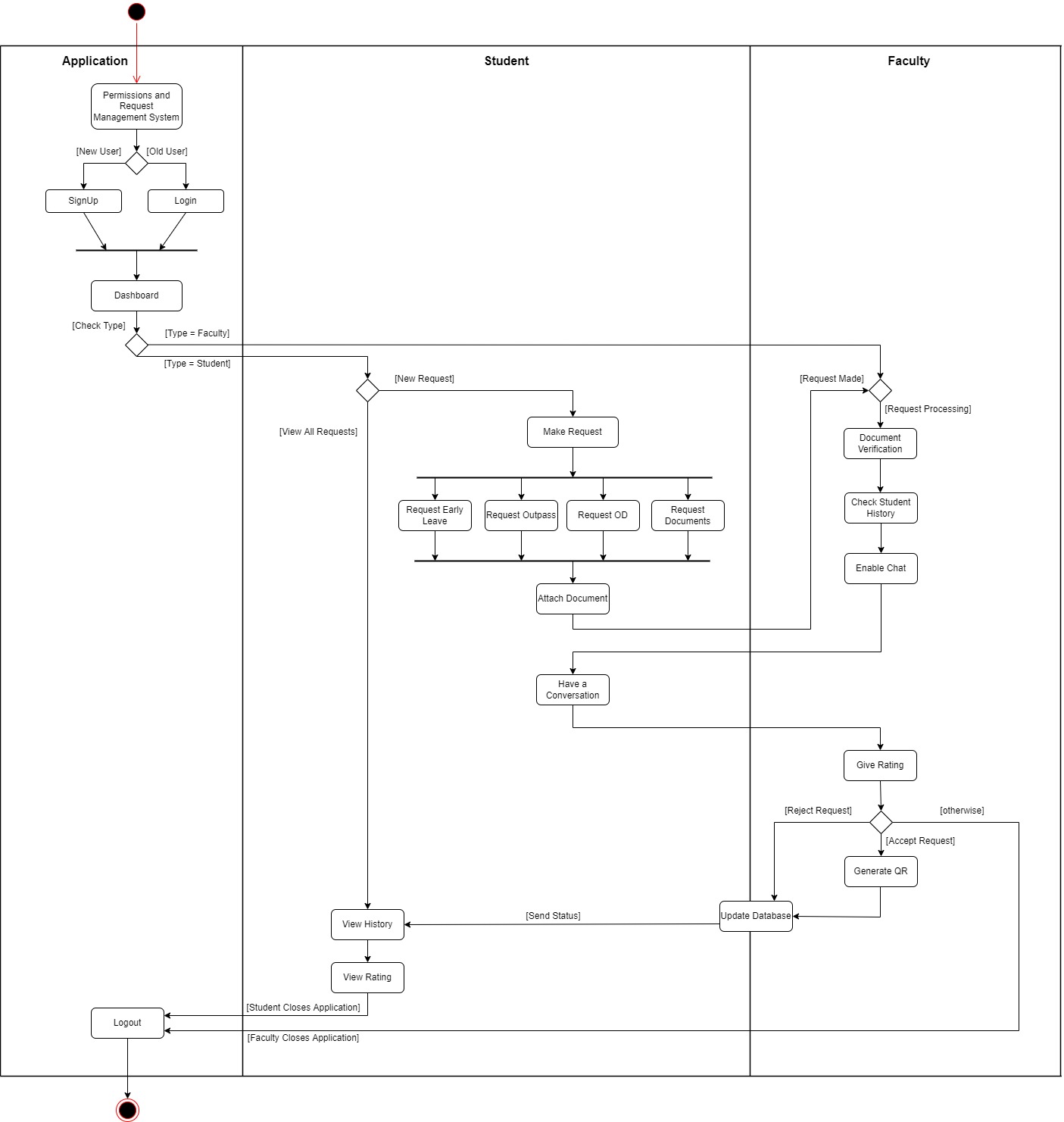
**Date:** 30-04-2022

**Aim:**

To construct an Activity Diagram for the given use case by identifying actors and activities.

**Notations used for Activity Diagram:**

**Activity Diagram:**

****

**Explanation:**

The above Activity diagram shows the flow of the process that takes place from login till logout. It contains 3 swimlanes depicting the 3 main players responsible for in the software. The application is the base where login/registration and verification from the database happens. The student swimlane consists of all the operations that the student performs which includes making a request, attaching necessary documents, view their requests history, rating and QR Code. The Faculty swimlane is where the faculty can choose to accept or reject the student’s request. If the facuty wants more clarifications, then they can enable chat and chat with the student. After that, the faculty has to give the rating based on the student’s request and behaviour. The student then can check whether their request got approved or not. If yes, they can show the QR code wherever necessary for authentication. The final step is, the student can Log out of the application. The above diagram consists of all possible scenarios that can take place to start from the starting stage to go the ending stage.

**Implementation**

**Ex.No:** 7

**Date:** 08-05-2022

**Server Side Framwork:**

**index.js:**

const express = require("express");

const mysql = require("mysql2");

const cors = require("cors");

const app = express();

//parse the inputs from the front-end

app.use(express.json());

app.use(cors());

const db = mysql.createConnection({

user: "root",

host: "localhost",

password: "123Washington!!",

database: "pros\_db",

});

app.post('/register', (req, res) => {

const username = req.body.username;

const password = req.body.password;

const userAppName = req.body.userAppName;

const profilePhoto = req.body.profilePhoto;

const userType = req.body.userType;

db.query(

"INSERT INTO users (username, password, userAppName, profilePhoto, userType) VALUES (?,?,?,?,?)",

[username, password, userAppName, profilePhoto, userType],

(err, result) => {

if(err) {

//console.log(err);

res.send({message:err.code});

}

else {

res.send({message:"Success! Enter EMAIL ID and password to login!"});

}

});

});

app.post("/login", (req, res) => {

const username = req.body.username;

const password = req.body.password;

db.query(

"SELECT \* FROM users where username = ? AND password = ?",

[username, password],

(err, result) => {

if(err) {

//console.log(err);

res.send({err:err});

}

if (result.length > 0) {

res.send(result);

}

else {

res.send({message:"Wrong Username & Password Combination"});

}

});

});

app.post('/req1', (req, res) => {

const name = req.body.uname;

const phno = req.body.uphno;

const emailID = req.body.uemailID;

const year = req.body.uyear;

const reason = req.body.ureason;

const fdate = req.body.ufdate;

const tdate = req.body.utdate;

const letter = req.body.letter;

const dep = req.body.udep;

db.query(

"INSERT INTO hostelleave (name, phno, emailID, year, reason, dep, fdate, tdate, letter) VALUES (?,?,?,?,?,?,?,?,?)",

[name, phno, emailID, year, reason, dep, fdate, tdate, letter],

(err, result) => {

if(err) {

res.send({message: "Please make sure you have filled in the correct details! You have either have an outstanding request or have not filled out the fields properly!"});

}

else {

res.send({message: "Success!"});

}

});

});

app.post('/get1', (req,res) => {

const name = req.body.username;

db.query(

"SELECT \* FROM hostelleave WHERE name = ?",

[name],

(err, result) => {

console.log(res);

if(result.length>0) {

res.send({message:"Pending!"});

}

else {

res.send({message:"You have not made any recent request!"});

}

}

)

});

app.post('/res1', (req, res) => {

db.query(

"SELECT \* FROM hostelleave",

(err, result) => {

if(err) {

res.send({err:err});

}

else{

res.send(result);

}

});

});

app.post("/clean1", (req,res) => {

db.query(

"DELETE FROM hostelleave where decision is NULL",

(err,result) => {

console.log(err);

res.send({message:"Complete!"})

});

});

app.post('/req2', (req, res) => {

const name = req.body.uname;

const phno = req.body.uphno;

const emailID = req.body.uemailID;

const year = req.body.uyear;

const reason = req.body.ureason;

const dep = req.body.udep;

const tdate = req.body.utdate;

const time = req.body.utime;

const letter = req.body.letter;

db.query(

"INSERT INTO earlyleave (name, phno, emailID, year, reason, dep, tdate, time, letter) VALUES (?,?,?,?,?,?,?,?,?)",

[name, phno, emailID, year, reason, dep, tdate, time, letter],

(err, result) => {

if(err) {

res.send({message: "Please make sure you have filled in the correct details! You have either have an outstanding request or have not filled out the fields properly!"});

}

else {

res.send({message: "Success!"});

}

});

});

app.post('/get2', (req,res) => {

const name = req.body.username;

db.query(

"SELECT \* FROM earlyleave WHERE name = ?",

[name],

(err, result) => {

console.log(res);

if(result.length>0) {

res.send({message:"Pending!"});

}

else {

res.send({message:"You have not made any recent request!"});

}

}

)

});

app.post('/res2', (req, res) => {

db.query(

"SELECT \* FROM earlyleave",

(err, result) => {

if(err) {

res.send({err:err});

}

else{

res.send(result);

}

});

});

app.post("/clean2", (req,res) => {

db.query(

"DELETE FROM earlyleave where decision is NULL",

(err,result) => {

console.log(err);

res.send({message:"Complete!"})

});

});

app.post('/req3', (req, res) => {

const name = req.body.uname;

const phno = req.body.uphno;

const emailID = req.body.uemailID;

const year = req.body.uyear;

const reason = req.body.ureason;

const dep = req.body.udep;

const tdate = req.body.utdate;

const ftime = req.body.uftime;

const ttime = req.body.uttime;

const letter = req.body.letter;

db.query(

"INSERT INTO onduty (name, phno, emailID, year, reason, dep, tdate, ftime, ttime, letter) VALUES (?,?,?,?,?,?,?,?,?,?)",

[name, phno, emailID, year, reason, dep, tdate, ftime, ttime, letter],

(err, result) => {

if(err) {

res.send({message: "Please make sure you have filled in the correct details! You have either have an outstanding request or have not filled out the fields properly!"});

}

else {

res.send({message: "Success!"});

}

});

});

app.post('/get3', (req,res) => {

const name = req.body.username;

db.query(

"SELECT \* FROM onduty WHERE name = ?",

[name],

(err, result) => {

console.log(res);

if(result.length>0) {

res.send({message:"Pending!"});

}

else {

res.send({message:"You have not made any recent request!"});

}

}

)

});

app.post('/res3', (req, res) => {

db.query(

"SELECT \* FROM onduty",

(err, result) => {

if(err) {

res.send({err:err});

}

else{

res.send(result);

}

});

});

app.post("/clean3", (req,res) => {

db.query(

"DELETE FROM onduty where decision is NULL",

(err,result) => {

console.log(err);

res.send({message:"Complete!"})

});

});

app.post('/req4', (req, res) => {

const name = req.body.uname;

const phno = req.body.uphno;

const emailID = req.body.uemailID;

const year = req.body.uyear;

const reason = req.body.ureason;

const dep = req.body.udep;

const tdate = req.body.utdate;

const time = req.body.utime;

const letter = req.body.udocuments;

db.query(

"INSERT INTO reqdoc (name, phno, emailID, year, reason, dep, tdate, time, letter) VALUES (?,?,?,?,?,?,?,?,?)",

[name, phno, emailID, year, reason, dep, tdate, time, letter],

(err, result) => {

if(err) {

res.send({message: "Please make sure you have filled in the correct details! You have either have an outstanding request or have not filled out the fields properly!"});

}

else {

res.send({message: "Success!"});

}

});

});

app.post('/get4', (req,res) => {

const name = req.body.username;

db.query(

"SELECT \* FROM reqdoc WHERE name = ?",

[name],

(err, result) => {

console.log(res);

if(result.length>0) {

res.send({message:"Pending!"});

}

else {

res.send({message:"You have not made any recent request!"});

}

}

)

});

app.post('/res4', (req, res) => {

db.query(

"SELECT \* FROM reqdoc",

(err, result) => {

if(err) {

res.send({err:err});

}

else{

res.send(result);

}

});

});

app.post("/clean4", (req,res) => {

db.query(

"DELETE FROM reqdoc where decision is NULL",

(err,result) => {

console.log(err);

res.send({message:"Complete!"})

});

});

app.post('/req5', (req, res) => {

const name = req.body.uname;

const phno = req.body.uphno;

const emailID = req.body.uemailID;

const year = req.body.uyear;

const reason = req.body.ureason;

const dep = req.body.udep;

const femail = req.body.ufemailID;

const letter = req.body.letter;

db.query(

"INSERT INTO otherreq (name, phno, emailID, year, reason, dep, femail, letter) VALUES (?,?,?,?,?,?,?,?)",

[name, phno, emailID, year, reason, dep, femail, letter],

(err, result) => {

if(err) {

res.send({message: "Please make sure you have filled in the correct details! You have either have an outstanding request or have not filled out the fields properly!"});

}

else {

res.send({message: "Success!"});

}

});

});

app.post('/get5', (req,res) => {

const name = req.body.username;

db.query(

"SELECT \* FROM otherreq WHERE name = ?",

[name],

(err, result) => {

console.log(res);

if(result.length>0) {

res.send({message:"Pending!"});

}

else {

res.send({message:"You have not made any recent request!"});

}

}

)

});

app.post('/res5', (req, res) => {

db.query(

"SELECT \* FROM otherreq",

(err, result) => {

if(err) {

res.send({err:err});

}

else{

res.send(result);

}

});

});

app.post("/clean5", (req,res) => {

db.query(

"DELETE FROM otherreq where decision is NULL",

(err,result) => {

console.log(err);

res.send({message:"Complete!"})

});

});

app.listen(3001, () => {

console.log("server running on port 3001");

});

**pros.sql:**

CREATE DATABASE pros\_db;

USE pros\_db;

CREATE TABLE users(

username VARCHAR(100) NOT NULL,

password VARCHAR(100) NOT NULL,

userAppName VARCHAR(100) NOT NULL,

profilePhoto VARCHAR(100) NOT NULL,

userType VARCHAR(45) NOT NULL,

PRIMARY KEY(username)

);

CREATE TABLE hostelleave(

name VARCHAR(100) NOT NULL,

phno INT(11) NOT NULL,

emailID VARCHAR(200) NOT NULL,

year INT(4) NOT NULL,

reason VARCHAR(400) NOT NULL,

dep VARCHAR(100) NOT NULL,

fdate VARCHAR(12) NOT NULL,

tdate VARCHAR(12) NOT NULL,

letter VARCHAR(500) NOT NULL,

decision VARCHAR(5),

PRIMARY KEY(name)

);

CREATE TABLE earlyleave(

name VARCHAR(100) NOT NULL,

phno INT(11) NOT NULL,

emailID VARCHAR(200) NOT NULL,

year INT(4) NOT NULL,

reason VARCHAR(400) NOT NULL,

dep VARCHAR(100) NOT NULL,

tdate VARCHAR(12) NOT NULL,

time VARCHAR(12) NOT NULL,

letter VARCHAR(500) NOT NULL,

decision VARCHAR(5),

PRIMARY KEY(name)

);

CREATE TABLE onduty(

name VARCHAR(100) NOT NULL,

phno INT(11) NOT NULL,

emailID VARCHAR(200) NOT NULL,

year INT(4) NOT NULL,

reason VARCHAR(400) NOT NULL,

dep VARCHAR(100) NOT NULL,

tdate VARCHAR(12) NOT NULL,

ftime VARCHAR(12) NOT NULL,

ttime VARCHAR(12) NOT NULL,

letter VARCHAR(500) NOT NULL,

decision VARCHAR(5),

PRIMARY KEY(name)

);

CREATE TABLE reqdoc(

name VARCHAR(100) NOT NULL,

phno INT(11) NOT NULL,

emailID VARCHAR(200) NOT NULL,

year INT(4) NOT NULL,

reason VARCHAR(400) NOT NULL,

dep VARCHAR(100) NOT NULL,

tdate VARCHAR(12) NOT NULL,

time VARCHAR(12) NOT NULL,

letter VARCHAR(500) NOT NULL,

decision VARCHAR(5),

PRIMARY KEY(name)

);

CREATE TABLE otherreq(

name VARCHAR(100) NOT NULL,

phno INT(11) NOT NULL,

emailID VARCHAR(200) NOT NULL,

year INT(4) NOT NULL,

reason VARCHAR(400) NOT NULL,

dep VARCHAR(100) NOT NULL,

femail VARCHAR(200) NOT NULL,

letter VARCHAR(500) NOT NULL,

decision VARCHAR(5),

PRIMARY KEY(name)

);

**Client Side Framework:**

**App.js:**

import React from "react";

import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

import LoginRegister from "./Pages/LoginRegister";

import StudentLandingPage from "./Pages/StudentLandingPage";

import FacultyLandingPage from "./Pages/FacultyLandingPage";

import ErrorPage from "./Pages/ErrorPage";

import Form1 from "./Pages/Form1";

import Form2 from "./Pages/Form2";

import Form3 from "./Pages/Form3";

import Form4 from "./Pages/Form4";

import Form5 from "./Pages/Form5";

import StudentViewReq from "./Pages/StudentViewReq";

import Aform1 from "./Pages/Aform1";

import Aform2 from "./Pages/Aform2";

import Aform3 from "./Pages/Aform3";

import Aform4 from "./Pages/Aform4";

import Aform5 from "./Pages/Aform5";

function App() {

return (

<Router>

<Routes>

<Route path="/" element={<LoginRegister />} />

<Route path="/studentProfile" element={<StudentLandingPage />} />

<Route path="/studentView" element={<StudentViewReq />} />

<Route path="/facultyProfile" element={<FacultyLandingPage />} />

<Route path="/form1" element={<Form1 />}></Route>

<Route path="/form2" element={<Form2 />}></Route>

<Route path="/form3" element={<Form3 />}></Route>

<Route path="/form4" element={<Form4 />}></Route>

<Route path="/form5" element={<Form5 />}></Route>

<Route path="/aform1" element={<Aform1 />}></Route>

<Route path="/aform2" element={<Aform2 />}></Route>

<Route path="/aform3" element={<Aform3 />}></Route>

<Route path="/aform4" element={<Aform4 />}></Route>

<Route path="/aform5" element={<Aform5 />}></Route>

<Route path="\*" element={<ErrorPage />} />

</Routes>

</Router>

);

}

export default App;

**index.js:**

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);

**LoginRegister.js:**

import React, { useState } from "react";

import Axios from "axios";

import "./LoginRegister.css";

import { useNavigate } from "react-router-dom";

function LoginRegister() {

let navigate = useNavigate();

//registration details

const [emailIDReg, setEmailIDReg] = useState('');

const [passwordReg, setPasswordReg] = useState('');

const [usernameReg, setUsernameReg] = useState('');

const [picReg, setPicReg] = useState('');

const [typeReg, setTypeReg] = useState('');

//login details

const [emailID, setEmailID] = useState('');

const [password, setPassword] = useState('');

//login and register status

const [loginStatus, setLoginStatus] = useState('');

const [regStatus, setRegStatus] = useState('');

//pass register data to backend

const register = () => {

Axios.post("http://localhost:3001/register", {

username: emailIDReg,

password: passwordReg,

userAppName: usernameReg,

profilePhoto: picReg,

userType: typeReg,

}).then((response) => {

console.log(response);

setRegStatus(response.data.message);

});

};

//pass login data to backend

const login = () => {

Axios.post("http://localhost:3001/login",{

username: emailID,

password: password,

}).then((response) => {

//console.log(response);

if (response.data.message) {

setLoginStatus(response.data.message);

}

else{

if(response.data[0].userType=="Student") {

navigate("/studentProfile",

{ state: { userAppName: response.data[0].userAppName, profilePhoto: response.data[0].profilePhoto, username: response.data[0].username } });

}

else {

navigate("/facultyProfile",

{ state: { userAppName: response.data[0].userAppName, profilePhoto: response.data[0].profilePhoto, username: response.data[0].username } });

}

}

});

};

return (

<div className="LoginRegister">

<div className="login">

<h1>Login to your Account</h1>

<label>Email-ID: </label>

<input id="uname" type="text" name="email" placeholder="abcde123@gmail.com" onChange={(e) => {

setEmailID(e.target.value);

}}/>

<br />

<br />

<label>Password: </label>

<input type="text" placeholder="\*\*\*\*" name="password" onChange={(e) => {

setPassword(e.target.value);

}}/>

<br />

<br />

<button onClick={login}>Login</button>

</div>

<h1 style={{color: "red"}}>{loginStatus}</h1>

<br />

<br />

<br />

<br />

<div className="registration">

<h1>Create an Account</h1>

<div>

<label>User Type: </label>

<br />

Student<input type="radio" checked={typeReg=="Student"} onChange={() => setTypeReg("Student")} />

<br />

Faculty<input type="radio" checked={typeReg=="Faculty"} onChange={() => setTypeReg("Faculty")} />

</div>

<br />

<label>Email-ID: </label>

<input type="text" onChange={(e) => {

setEmailIDReg(e.target.value);

}}/>

<br />

<br />

<label>Password: </label>

<input type="text" onChange={(e) => {

setPasswordReg(e.target.value);

}}/>

<br />

<br />

<label>Username: </label>

<input type="text" onChange={(e) => {

setUsernameReg(e.target.value);

}}/>

<br />

<br />

<label>Profile Pic: </label>

<input type="text" onChange={(e) => {

setPicReg(e.target.value);

}}/>

<br />

<br />

<button onClick={register}>Register</button>

</div>

<h1>{regStatus}</h1>

</div>

);

}

export default LoginRegister;

**LoginRegister.css:**

.LoginRegister {

text-align: center;

}

.LoginRegister-logo {

height: 40vmin;

pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

.LoginRegister-logo {

animation: LoginRegister-logo-spin infinite 20s linear;

}

}

.LoginRegister-header {

background-color: #282c34;

min-height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: calc(10px + 2vmin);

color: white;

}

.LoginRegister-link {

color: #61dafb;

}

@keyframes LoginRegister-logo-spin {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

**FacultyLandingPage.js:**

import React from "react";

import { useLocation } from "react-router-dom";

import "./FacultyLandingPage.css";

import { useNavigate } from "react-router-dom";

function facultyLandingPage() {

let location = useLocation()

const { state } = useLocation();

const { userAppName, profilePhoto } = state;

let navigate = useNavigate();

const logout = () => {

navigate("/");

}

const goto1 = () => {

navigate("/aform1",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

const goto2 = () => {

navigate("/aform2",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

const goto3 = () => {

navigate("/aform3",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

const goto4 = () => {

navigate("/aform4",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

const goto5 = () => {

navigate("/aform5",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

return (

<div className="FacultyLandingPage">

<h1>Faculty Landing Page</h1>

<h2>You are logged in</h2>

<div>

<p>

<h3>{userAppName}</h3>

<img src={profilePhoto} alt="user profile photo" />

</p>

</div>

<button onClick={goto1}>View Hostel Leave Requests</button>

<button style={{margin:10}} onClick={goto2}>View Early Leave Requests</button>

<button style={{margin:10}} onClick={goto3}>View On-Duty Requests</button>

<button style={{margin:10}} onClick={goto4}>View Document Requests</button>

<button style={{margin:10}} onClick={goto5}>View Other General Requests</button>

<br />

<br />

<button onClick={logout}>Logout</button>

</div>

);

}

export default facultyLandingPage;

**FacultyLandingPage.css:**

.FacultyLandingPage {

text-align: center;

}

**StudentLandingPage.js:**

import React from "react";

import { useLocation } from "react-router-dom";

import { useNavigate } from "react-router-dom";

import "./StudentLandingPage.css";

function StudentLandingPage() {

let navigate = useNavigate();

const logout = () => {

navigate("/");

}

const goto1 = () => {

navigate("/form1",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

const goto2 = () => {

navigate("/form2",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

const goto3 = () => {

navigate("/form3",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

const goto4 = () => {

navigate("/form4",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

const goto5 = () => {

navigate("/form5",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

const viewReq = () => {

navigate("/studentView",{ state: { userAppName: userAppName, profilePhoto: profilePhoto, username: username}})

}

let location = useLocation()

const { state } = useLocation();

const { userAppName, profilePhoto, username } = state;

return (

<div className="StudentLandingPage">

<h1>Student Landing Page</h1>

<h2>You are logged in</h2>

<div>

<p>

<h2>{userAppName}</h2>

<img src={profilePhoto} alt="user profile photo" />

</p>

</div>

<br />

<br />

<button onClick={goto1}>Hostel Leave Form [Hostelers]</button>

<button style={{margin:10}} onClick={goto2}>Early Leave Form [Day Scholars]</button>

<button style={{margin:10}} onClick={goto3}>On-Duty Form</button>

<button style={{margin:10}} onClick={goto4}>Documents Request Form</button>

<button style={{margin:10}} onClick={goto5}>Other Permissions</button>

<br />

<br />

<button onClick={viewReq}>View Requests</button>

<br />

<br />

<button onClick={logout}>Logout</button>

</div>

);

}

export default StudentLandingPage;

**StudentLandingPage.css:**

.StudentLandingPage-id {

text-align: center;

}

**StudentViewReq.js:**

import React from "react";

import Axios from "axios";

import { useState } from "react";

import { useNavigate, useLocation } from "react-router-dom";

function StudentViewReq() {

let navigate = useNavigate();

const { state } = useLocation();

const { userAppName, profilePhoto, username } = state;

const [form1Status,setForm1Status] = useState('');

const [form2Status,setForm2Status] = useState('');

const [form3Status,setForm3Status] = useState('');

const [form4Status,setForm4Status] = useState('');

const [form5Status,setForm5Status] = useState('');

const ans1 = () => {

Axios.post("http://localhost:3001/get1", {

username: userAppName

}).then((response) => {

console.log(response);

setForm1Status(response.data.message);

});

};

const ans2 = () => {

Axios.post("http://localhost:3001/get2", {

username: userAppName

}).then((response) => {

setForm2Status(response.data.message);

});

};

const ans3 = () => {

Axios.post("http://localhost:3001/get3", {

username: userAppName

}).then((response) => {

setForm3Status(response.data.message);

});

};

const ans4 = () => {

Axios.post("http://localhost:3001/get4", {

username: userAppName

}).then((response) => {

setForm4Status(response.data.message);

});

};

const ans5 = () => {

Axios.post("http://localhost:3001/get5", {

username: userAppName

}).then((response) => {

setForm5Status(response.data.message);

});

};

return (

<div>

<h1>View Pending Requests</h1>

<div>

<button onClick={ans1}>Hostel Leave Form:</button>

<span> {form1Status}</span>

</div>

<br />

<div>

<button onClick={ans2}>Early Leave Form:</button>

<span> {form2Status}</span>

</div>

<br />

<div>

<button onClick={ans3}>On Duty Form:</button>

<span> {form3Status}</span>

</div>

<br />

<div>

<button onClick={ans4}>Original Document Form:</button>

<span> {form4Status}</span>

</div>

<br />

<div>

<button onClick={ans5}>Other Permission Requests Form:</button>

<span> {form5Status}</span>

</div>

</div>

);

}

export default StudentViewReq;

**form.css:**

body {

text-align: center;

}

input[type=text],

select,

textarea {

width: 100%;

padding: 12px;

color: azure;

background-color: rgba(65, 55, 55, 0.253);

border: 3px solid rgb(0, 0, 0);

border-radius: 4px;

resize: vertical;

}

input[type=email],

select,

textarea {

width: 100%;

padding: 12px;

color: azure;

background-color: rgba(65, 55, 55, 0.253);

border: 3px solid rgb(0, 0, 0);

border-radius: 4px;

resize: vertical;

}

input[type=number],

select,

textarea {

width: 100%;

padding: 12px;

color: azure;

background-color: rgba(65, 55, 55, 0.253);

border: 3px solid rgb(0, 0, 0);

border-radius: 4px;

resize: vertical;

}

label {

padding: 12px 12px 12px 0;

display: inline-block;

}

input[type=submit] {

background-color: rgb(0, 0, 0);

color: beige;

padding: 12px 20px;

border: 1px solid #45a049;

border-radius: 4px;

cursor: pointer;

float: right;

}

input[type=submit]:hover {

background-color: #45a049;

border: none;

}

input[type=reset] {

background-color: rgb(0, 0, 0);

color: beige;

padding: 12px 20px;

border: 1px solid #e48204;

border-radius: 4px;

cursor: pointer;

float: left;

}

input[type=reset]:hover {

background-color: #e45700;

border: none;

}

input[type=date] {

background-color: azure;

width: 15%;

border-radius: 10%;

color: black;

border: none;

padding: 12px 20px;

}

select,

textarea {

background-color: transparent;

color: aliceblue;

}

option {

color: aliceblue;

background-color: black;

line-height: 8px;

font-size: medium;

}

.container {

border-radius: 5px;

background-color: rgba(146, 29, 150, 0.5);

padding: 20px;

}

**form1.js:**

import React, { useState } from "react";

import Axios from "axios";

import "./form.css";

import { useNavigate } from "react-router-dom";

import { useLocation } from "react-router-dom";

function Form1() {

let navigate = useNavigate();

const { state } = useLocation();

const { userAppName, profilePhoto } = state;

//registration details

const [name, setname] = useState('');

const [phno, setphno] = useState('');

const [emailID, setEmailID] = useState('');

const [year, setyear] = useState('');

const [reason, setreason] = useState('');

const [fdate, setfdate] = useState('');

const [tdate, settdate] = useState('');

const [reasonletter, setreasonletter] = useState('');

const [dep, setdep] = useState('');

const refresh = () => {

};

const request = () => {

Axios.post("http://localhost:3001/req1", {

uname: name,

uphno: phno,

uemailID: emailID,

uyear: year,

ureason: reason,

ufdate: fdate,

utdate: tdate,

letter: reasonletter,

udep: dep

}).then((response) => {

if(response.data.message == "Success!") {

window.alert(response.data.message);

navigate("/studentProfile",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

else{

window.alert(response.data.message);

}

});

};

return (

<div>

<h1>Hostel Leave Form</h1>

<br />

<br />

<div className="container">

<label>Name \*</label>

<input type="text" id="name" name="name" placeholder="Your name.." pattern="' '|[a-zA-Z ]\*" required onChange={(e) => {

setname(e.target.value);}}/>

<label>Phone number</label>

<input name="contact number" type="number" placeholder="Your mobile no.." pattern="[0-9]{1,10}" required onChange={(e) => {

setphno(e.target.value);}}/>

<label>Mail Id</label>

<input type="email" id="email" name="mailid" placeholder="Your mail id.." required onChange={(e) => {

setEmailID(e.target.value);}} />

<br />

<label>Department</label>

<input type="text" id="dept" name="dept" placeholder="Your Department.." pattern="' '|[a-zA-Z ]\*"

required onChange={(e) => {setdep(e.target.value);}}/>

<br />

<label>Year</label>

<input type="number" id="year" name="year" pattern="[1-4]" placeholder="Enter Year" required onChange={(e) => {

setyear(e.target.value);}} />

<br />

<br />

<label>Date Leaving:</label><br />

<input name="from" id="from" type="date" required onChange={(e) => {setfdate(e.target.value);}}/>

<br />

<label>Date Returning:</label><br />

<input name="to" id="to" type="date" required onChange={(e) => {settdate(e.target.value);}}/>

<br /><br /><br />

<label>Type the reason </label>

<input type="text" id="reason" name="reason" placeholder="Your Reason here..." required onChange={(e) => {

setreason(e.target.value);}} />

<label>Request letter:</label>

<input type="text" name="proof" onChange={(e) => {

setreasonletter(e.target.value);}}/>

<br />

<br />

<input type="submit" value="Submit" onClick={request}/>

<input type="reset" />

</div>

</div>

);

}

export default Form1;

**form2.js:**

import React, { useState } from "react";

import Axios from "axios";

import "./form.css";

import { useNavigate } from "react-router-dom";

import { useLocation } from "react-router-dom";

function Form2() {

let navigate = useNavigate();

const { state } = useLocation();

const {userAppName, profilePhoto } = state;

//registration details

const [name, setname] = useState('');

const [phno, setphno] = useState('');

const [emailID, setEmailID] = useState('');

const [year, setyear] = useState('');

const [reason, setreason] = useState('');

const [dep, setdep] = useState('');

const [tdate, settdate] = useState('');

const [time, settime] = useState('');

const [reasonletter, setreasonletter] = useState('');

const request = () => {

Axios.post("http://localhost:3001/req2", {

uname: name,

uphno: phno,

uemailID: emailID,

uyear: year,

ureason: reason,

udep: dep,

utdate: tdate,

utime: time,

letter: reasonletter

}).then((response) => {

if(response.data.message == "Success!") {

window.alert(response.data.message);

navigate("/studentProfile",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

else{

window.alert(response.data.message);

}

});

};

return (

<div>

<h1>Early Leave Form</h1>

<br /><br />

<div className="container">

<label for="fname">Name</label>

<input type="text" id="name" name="name" placeholder="Your name.." pattern="' '|[a-zA-Z ]\*"

required onChange={(e) => {setname(e.target.value);}}/>

<label for="phoneno">Phone number</label>

<input name="contact number" type="number" placeholder="Your mobile no.."

pattern="[0-9]{1,10}" required onChange={(e) => {setphno(e.target.value);}}/>

<label for="email">Mail Id</label>

<input type="email" id="email" name="mailid" placeholder="Your mail id.." required onChange={(e) => {setEmailID(e.target.value);}}/>

<label for="year">Year</label>

<input type="number" id="year" name="year" pattern="[1-4]" placeholder="Enter Year" required onChange={(e) => {setyear(e.target.value);}}/>

<br /><br />

<label for="dept">Department</label>

<input type="text" id="dept" name="dept" placeholder="Your Department.." pattern="' '|[a-zA-Z ]\*"

required onChange={(e) => {setdep(e.target.value);}}/>

<br /><br />

<label>Early Leave Form(Dayscholars only)</label>

<input type="checkbox" name="type" value="Early Leave Form" checked required />

<br /><br />

<label>Date Leaving:</label><br />

<input name="from" id="from" type="date" required onChange={(e) => {settdate(e.target.value);}}/><br />

<label for="time">Time Leaving</label>

<input type="text" id="time" name="time" placeholder="Enter time.." required onChange={(e) => {settime(e.target.value);}}/>

<label for="reason">Type the reason </label>

<input type="text" id="reason" name="reason" placeholder="Your Reason here..." required onChange={(e) => {setreason(e.target.value);}}/>

<label>Request letter:</label>

<input type="text" name="proof" onChange={(e) => {setreasonletter(e.target.value);}}/>

<br /><br />

<input type="submit" value="Submit" onClick={request} />

<input type="reset" />

</div>

</div>

);

}

export default Form2;

**form3.js:**

import React, { useState } from "react";

import Axios from "axios";

import "./form.css";

import { useNavigate } from "react-router-dom";

import { useLocation } from "react-router-dom";

function Form3() {

let navigate = useNavigate();

const { state } = useLocation();

const {userAppName, profilePhoto } = state;

//registration details

const [name, setname] = useState('');

const [phno, setphno] = useState('');

const [emailID, setEmailID] = useState('');

const [year, setyear] = useState('');

const [reason, setreason] = useState('');

const [dep, setdep] = useState('');

const [tdate, settdate] = useState('');

const [ftime, setftime] = useState('');

const [ttime, setttime] = useState('');

const [reasonletter, setreasonletter] = useState('');

const request = () => {

Axios.post("http://localhost:3001/req3", {

uname: name,

uphno: phno,

uemailID: emailID,

uyear: year,

ureason: reason,

udep: dep,

utdate: tdate,

uftime: ftime,

uttime: ttime,

letter: reasonletter

}).then((response) => {

if(response.data.message == "Success!") {

window.alert(response.data.message);

navigate("/studentProfile",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

else{

window.alert(response.data.message);

}

});

};

return (

<div>

<h1>On-Duty Form</h1>

<br /><br />

<div className="container">

<label for="fname">Name</label>

<input type="text" id="name" name="name" placeholder="Your name.." pattern="' '|[a-zA-Z ]\*"

required onChange={(e) => {setname(e.target.value);}}/>

<label for="phoneno">Phone number</label>

<input name="contact number" type="number" placeholder="Your mobile no.."

pattern="[0-9]{1,10}" required onChange={(e) => {setphno(e.target.value);}}/>

<label for="email">Mail Id</label>

<input type="email" id="email" name="mailid" placeholder="Your mail id.." required onChange={(e) => {setEmailID(e.target.value);}}/>

<label for="year">Year</label>

<input type="number" id="year" name="year" pattern="[1-4]" placeholder="Enter Year" required onChange={(e) => {setyear(e.target.value);}}/><br /><br />

<label for="dept">Department</label>

<input type="text" id="dept" name="dept" placeholder="Your Department.." pattern="' '|[a-zA-Z ]\*"

required onChange={(e) => {setdep(e.target.value);}}/><br /><br />

<label>On Duty Request Form</label>

<input type="checkbox" name="type" value="Early Leave Form" checked required />

<br /><br />

<label>Date On Duty:</label><br />

<input name="date" id="date" type="date" required onChange={(e) => {settdate(e.target.value);}}/><br />

<label for="time">Starting Time</label>

<input type="text" id="from" name="from" placeholder="Enter from time.." required onChange={(e) => {setftime(e.target.value);}}/>

<label for="time">Ending Time</label>

<input type="text" id="to" name="to" placeholder="Enter to time.." required onChange={(e) => {setttime(e.target.value);}}/>

<label for="reason">Type the reason </label>

<input type="text" id="reason" name="reason" placeholder="Your Reason here..." required onChange={(e) => {setreason(e.target.value);}}/>

<label>Proof Document(mandatory):</label>

<input type="text" name = "proof" value="Proof letter" required onChange={(e) => {setreasonletter(e.target.value);}}/>

<br /><br />

<input type="submit" value="Submit" onClick={request} />

<input type="reset" />

</div>

</div>

);

}

export default Form3;

**form4.js:**

import React, { useState } from "react";

import Axios from "axios";

import "./form.css";

import { useLocation, useNavigate } from "react-router-dom";

function Form4() {

let navigate = useNavigate();

const { state } = useLocation();

const { userAppName, profilePhoto } = state;

//registration details

const [name, setname] = useState('');

const [phno, setphno] = useState('');

const [emailID, setEmailID] = useState('');

const [year, setyear] = useState('');

const [reason, setreason] = useState('');

const [dep, setdep] = useState('');

const [tdate, settdate] = useState('');

const [time, settime] = useState('');

const [documents, setdocuments] = useState('');

const request = () => {

Axios.post("http://localhost:3001/req4", {

uname: name,

uphno: phno,

uemailID: emailID,

uyear: year,

ureason: reason,

udep: dep,

utdate: tdate,

utime: time,

udocuments: documents

}).then((response) => {

if(response.data.message == "Success!") {

window.alert(response.data.message);

navigate("/studentProfile",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

else{

window.alert(response.data.message);

}

});

};

return (

<div>

<h1>Original Documents Request Form</h1>

<br /><br />

<div className="container">

<label for="fname">Name</label>

<input type="text" id="name" name="name" placeholder="Your name.." pattern="' '|[a-zA-Z ]\*" required onChange={(e) => {setname(e.target.value);}}/>

<label for="phoneno">Phone number</label>

<input name="contact number" type="number" placeholder="Your mobile no.." pattern="[0-9]{1,10}" required onChange={(e) => {setphno(e.target.value);}}/>

<label for="email">Mail Id</label>

<input type="email" id="email" name="mailid" placeholder="Your mail id.." required onChange={(e) => {setEmailID(e.target.value);}}/>

<label for="year">Year</label>

<input type="number" id="year" name="year" pattern="[1-4]" placeholder="Enter Year" required onChange={(e) => {setyear(e.target.value);}}/><br /><br />

<label for="dept">Department</label>

<input type="text" id="dept" name="dept" placeholder="Your Department.." pattern="' '|[a-zA-Z ]\*" required onChange={(e) => {setdep(e.target.value);}}/><br /><br />

<label>Original Documents Request Form</label>

<input type="checkbox" name="type" value="Early Leave Form" checked required />

<br /><br />

<label>Date</label><br />

<input name="from" id="from" type="date" required onChange={(e) => {settdate(e.target.value);}}/><br />

<label for="time">Time</label>

<input type="text" id="time" name="time" placeholder="Enter time.." required onChange={(e) => {settime(e.target.value);}}/>

<label for="request">Documents Name</label>

<input type="text" id="request" name="request" placeholder="Documents name here..." onChange={(e) => {setdocuments(e.target.value);}}/>

<label for="reason">Type the reason </label>

<input type="text" id="reason" name="reason" placeholder="Your Reason here..." required onChange={(e) => {setreason(e.target.value);}}/>

<br /><br />

<br /><br />

<input type="submit" value="Submit" onClick={request}/>

<input type="reset" />

</div>

</div>

);

}

export default Form4;

**form5.js:**

import React, { useState } from "react";

import Axios from "axios";

import "./form.css";

import { useLocation, useNavigate } from "react-router-dom";

function Form5() {

let navigate = useNavigate();

const { state } = useLocation();

const { userAppName, profilePhoto } = state;

//registration details

const [name, setname] = useState('');

const [phno, setphno] = useState('');

const [emailID, setEmailID] = useState('');

const [year, setyear] = useState('');

const [reason, setreason] = useState('');

const [dep, setdep] = useState('');

const [femailID, setfEmailID] = useState('');

const [reasonletter, setreasonletter] = useState('');

const request = () => {

Axios.post("http://localhost:3001/req5", {

uname: name,

uphno: phno,

uemailID: emailID,

uyear: year,

ureason: reason,

udep: dep,

ufemailID: femailID,

letter: reasonletter

}).then((response) => {

if(response.data.message == "Success!") {

window.alert(response.data.message);

navigate("/studentProfile",{ state: { userAppName: userAppName, profilePhoto: profilePhoto } });

}

else{

window.alert(response.data.message);

}

});

};

return (

<div>

<h1>Other Requests Form</h1>

<br /><br />

<div className="container">

<label>Name</label>

<input type="text" id="name" name="name" placeholder="Your name.." pattern="' '|[a-zA-Z ]\*"

required onChange={(e) => {setname(e.target.value);}}/>

<label>Phone number</label>

<input name="contact number" type="number" placeholder="Your mobile no.."

pattern="[0-9]{1,10}" required onChange={(e) => {setphno(e.target.value);}}/>

<label for="email">Mail Id</label>

<input type="email" id="email" name="mailid" placeholder="Your mail id.." required onChange={(e) => {setEmailID(e.target.value);}}/>

<label>Year</label>

<input type="number" id="year" name="year" pattern="[1-4]" placeholder="Enter Year" required onChange={(e) => {setyear(e.target.value);}}/><br /><br />

<label>Department</label>

<input type="text" id="dept" name="dept" placeholder="Your Department.." pattern="' '|[a-zA-Z ]\*"

required onChange={(e) => {setdep(e.target.value);}}/><br /><br />

<label>Other Request Form</label>

<input type="checkbox" name="type" value="Early Leave Form" checked required />

<br /><br />

<label>To faculty Mail ID</label>

<input type="email" id="femail" name="fmailid" placeholder="Faculty's mail id.." required onChange={(e) => {setfEmailID(e.target.value);}}/>

<label>Type the reason </label>

<br /><br />

<input type="text" id="reason" name="reason" placeholder="Your Reason here..." required onChange={(e) => {setreason(e.target.value);}}/>

<br /><br />

<label>Document(if necessarry): </label>

<input type="text" name = "proof" onChange={(e) => {setreasonletter(e.target.value);}}/>

<br /><br />

<input type="submit" value="Submit" onClick={request}/>

<input type="reset" />

</div>

</div>

);

}

export default Form5;

**Aform1.js:**

import React, { useState } from "react";

import { useLocation, useNavigate } from "react-router-dom";

import Axios from "axios";

function Aform1() {

let resString = "<table border=\"2px solid\"><tr><th>Name</th><th>Request</th><th>Accept/Reject</th></tr>";

const [test,setTest] = useState('');

let navigate = useNavigate();

const { state } = useLocation();

const { userAppName, profilePhoto, username } = state;

const getForm1 = () => {

Axios.post("http://localhost:3001/res1").then((response) => {

console.log(response);

for(let i=0;i<response.data.length;i++) {

resString += "<tr><td>" + response.data[i].name + "</td><td>" + response.data[i].reason +

"</td><td>" + "Accept<input type=\"radio\" name=\"decision" + i +"\" id=\"yes\"/> Reject<input type=\"radio\" name=\"decision" + i +"\" id=\"no\" />"

+ "</td></tr>";

}

resString += "</table>"

console.log(resString);

document.getElementById("disp").style.display = "block";

document.getElementById("disp1").style.display = "block";

document.getElementById("disp").innerHTML = resString;

});

};

const clearForm1 = () => {

Axios.post("http://localhost:3001/clean1").then((response) =>{

console.log("Cleaned!");

window.alert("Notification sent to students!");

navigate("/facultyProfile", { state: {userAppName: userAppName, profilePhoto: profilePhoto, username: username}});

})

};

return (

<div>

<h1>Hostel Leave Form Responses</h1>

<button onClick={getForm1}>Get Responses</button>

<br />

<br />

<div align="center">

<div style={{display: "None", margin: "10px", fontSize: "16px"}} id="disp"></div>

<br />

<button style={{display: "None"} } id="disp1" onClick={clearForm1}>Submit Responses</button>

</div>

</div>

);

}

export default Aform1;

**Aform2.js:**

import React, { useState } from "react";

import { useLocation, useNavigate } from "react-router-dom";

import Axios from "axios";

function Aform2() {

let resString = "<table border=\"2px solid\"><tr><th>Name</th><th>Request</th><th>Accept/Reject</th></tr>";

const [test,setTest] = useState('');

let navigate = useNavigate();

const { state } = useLocation();

const { userAppName, profilePhoto, username } = state;

const getForm2 = () => {

Axios.post("http://localhost:3001/res2").then((response) => {

console.log(response);

for(let i=0;i<response.data.length;i++) {

resString += "<tr><td>" + response.data[i].name + "</td><td>" + response.data[i].reason +

"</td><td>" + "Accept<input type=\"radio\" name=\"decision" + i +"\" id=\"yes\"/> Reject<input type=\"radio\" name=\"decision" + i +"\" id=\"no\" />"

+ "</td></tr>";

}

resString += "</table>"

console.log(resString);

document.getElementById("disp").style.display = "block";

document.getElementById("disp1").style.display = "block";

document.getElementById("disp").innerHTML = resString;

});

};

const clearForm2 = () => {

Axios.post("http://localhost:3001/clean2").then((response) =>{

console.log("Cleaned!");

window.alert("Notification sent to students!");

navigate("/facultyProfile", { state: {userAppName: userAppName, profilePhoto: profilePhoto, username: username}});

})

};

return (

<div>

<h1>Early Leave Form Responses</h1>

<button onClick={getForm2}>Get Responses</button>

<br />

<br />

<div align="center">

<div style={{display: "None", margin: "10px", fontSize: "16px"}} id="disp"></div>

<br />

<button style={{display: "None"} } id="disp1" onClick={clearForm2}>Submit Responses</button>

</div>

</div>

);

}

export default Aform2;

**Aform3.js:**

import React, { useState } from "react";

import { useLocation, useNavigate } from "react-router-dom";

import Axios from "axios";

function Aform3() {

let resString = "<table border=\"2px solid\"><tr><th>Name</th><th>Request</th><th>Accept/Reject</th></tr>";

const [test,setTest] = useState('');

let navigate = useNavigate();

const { state } = useLocation();

const { userAppName, profilePhoto, username } = state;

const getForm3 = () => {

Axios.post("http://localhost:3001/res3").then((response) => {

console.log(response);

for(let i=0;i<response.data.length;i++) {

resString += "<tr><td>" + response.data[i].name + "</td><td>" + response.data[i].reason +

"</td><td>" + "Accept<input type=\"radio\" name=\"decision" + i +"\" id=\"yes\"/> Reject<input type=\"radio\" name=\"decision" + i +"\" id=\"no\" />"

+ "</td></tr>";

}

resString += "</table>"

console.log(resString);

document.getElementById("disp").style.display = "block";

document.getElementById("disp1").style.display = "block";

document.getElementById("disp").innerHTML = resString;

});

};

const clearForm3 = () => {

Axios.post("http://localhost:3001/clean3").then((response) =>{

console.log("Cleaned!");

window.alert("Notification sent to students!");

navigate("/facultyProfile", { state: {userAppName: userAppName, profilePhoto: profilePhoto, username: username}});

})

};

return (

<div>

<h1>On Duty Form Responses</h1>

<button onClick={getForm3}>Get Responses</button>

<br />

<br />

<div align="center">

<div style={{display: "None", margin: "10px", fontSize: "16px"}} id="disp"></div>

<br />

<button style={{display: "None"} } id="disp1" onClick={clearForm3}>Submit Responses</button>

</div>

</div>

);

}

export default Aform3;

**Aform4.js:**

import React, { useState } from "react";

import { useLocation, useNavigate } from "react-router-dom";

import Axios from "axios";

function Aform4() {

let resString = "<table border=\"2px solid\"><tr><th>Name</th><th>Request</th><th>Accept/Reject</th></tr>";

const [test,setTest] = useState('');

let navigate = useNavigate();

const { state } = useLocation();

const { userAppName, profilePhoto, username } = state;

const getForm4 = () => {

Axios.post("http://localhost:3001/res4").then((response) => {

console.log(response);

for(let i=0;i<response.data.length;i++) {

resString += "<tr><td>" + response.data[i].name + "</td><td>" + response.data[i].reason +

"</td><td>" + "Accept<input type=\"radio\" name=\"decision" + i +"\" id=\"yes\"/> Reject<input type=\"radio\" name=\"decision" + i +"\" id=\"no\" />"

+ "</td></tr>";

}

resString += "</table>"

console.log(resString);

document.getElementById("disp").style.display = "block";

document.getElementById("disp1").style.display = "block";

document.getElementById("disp").innerHTML = resString;

});

};

const clearForm4 = () => {

Axios.post("http://localhost:3001/clean4").then((response) =>{

console.log("Cleaned!");

window.alert("Notification sent to students!");

navigate("/facultyProfile", { state: {userAppName: userAppName, profilePhoto: profilePhoto, username: username}});

})

};

return (

<div>

<h1>On Duty Form Responses</h1>

<button onClick={getForm4}>Get Responses</button>

<br />

<br />

<div align="center">

<div style={{display: "None", margin: "10px", fontSize: "16px"}} id="disp"></div>

<br />

<button style={{display: "None"} } id="disp1" onClick={clearForm4}>Submit Responses</button>

</div>

</div>

);

}

export default Aform4;

**Aform5.js:**

import React, { useState } from "react";

import { useLocation, useNavigate } from "react-router-dom";

import Axios from "axios";

function Aform5() {

let resString = "<table border=\"2px solid\"><tr><th>Name</th><th>Request</th><th>Accept/Reject</th></tr>";

const [test,setTest] = useState('');

let navigate = useNavigate();

const { state } = useLocation();

const { userAppName, profilePhoto, username } = state;

const getForm5 = () => {

Axios.post("http://localhost:3001/res3").then((response) => {

console.log(response);

for(let i=0;i<response.data.length;i++) {

resString += "<tr><td>" + response.data[i].name + "</td><td>" + response.data[i].reason +

"</td><td>" + "Accept<input type=\"radio\" name=\"decision" + i +"\" id=\"yes\"/> Reject<input type=\"radio\" name=\"decision" + i +"\" id=\"no\" />"

+ "</td></tr>";

}

resString += "</table>"

console.log(resString);

document.getElementById("disp").style.display = "block";

document.getElementById("disp1").style.display = "block";

document.getElementById("disp").innerHTML = resString;

});

};

const clearForm5 = () => {

Axios.post("http://localhost:3001/clean3").then((response) =>{

console.log("Cleaned!");

window.alert("Notification sent to students!");

navigate("/facultyProfile", { state: {userAppName: userAppName, profilePhoto: profilePhoto, username: username}});

})

};

return (

<div>

<h1>On Duty Form Responses</h1>

<button onClick={getForm5}>Get Responses</button>

<br />

<br />

<div align="center">

<div style={{display: "None", margin: "10px", fontSize: "16px"}} id="disp"></div>

<br />

<button style={{display: "None"} } id="disp1" onClick={clearForm5}>Submit Responses</button>

</div>

</div>

);

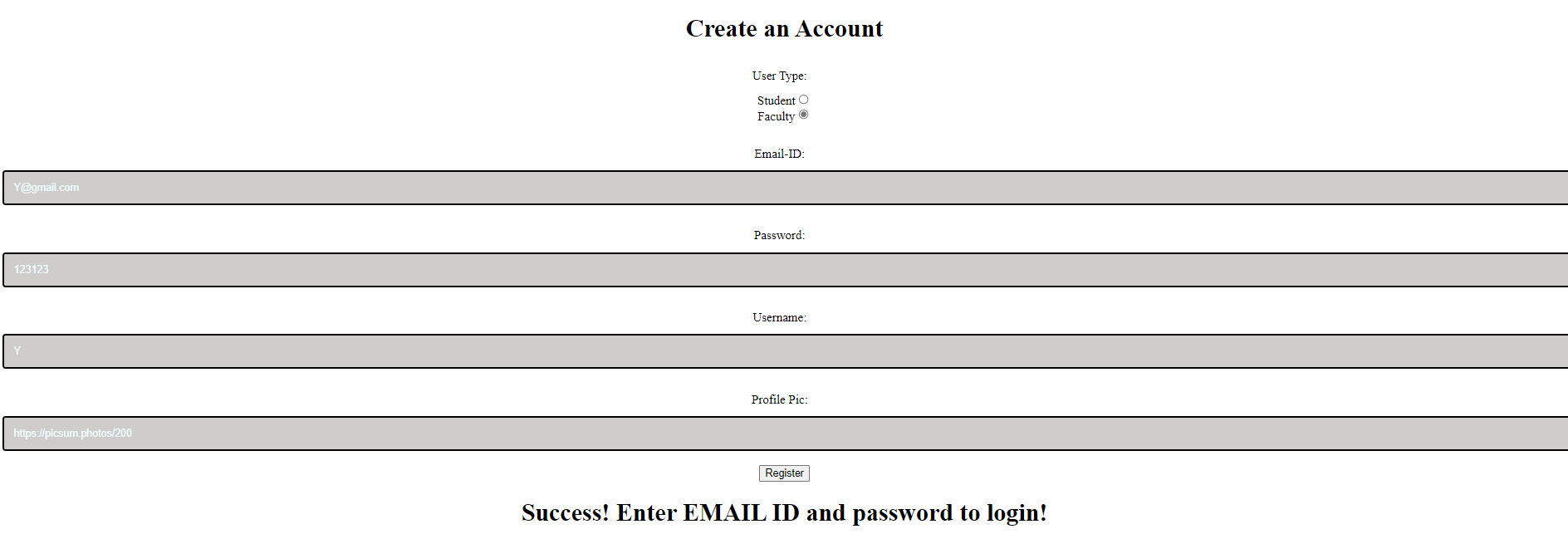
}

export default Aform5;

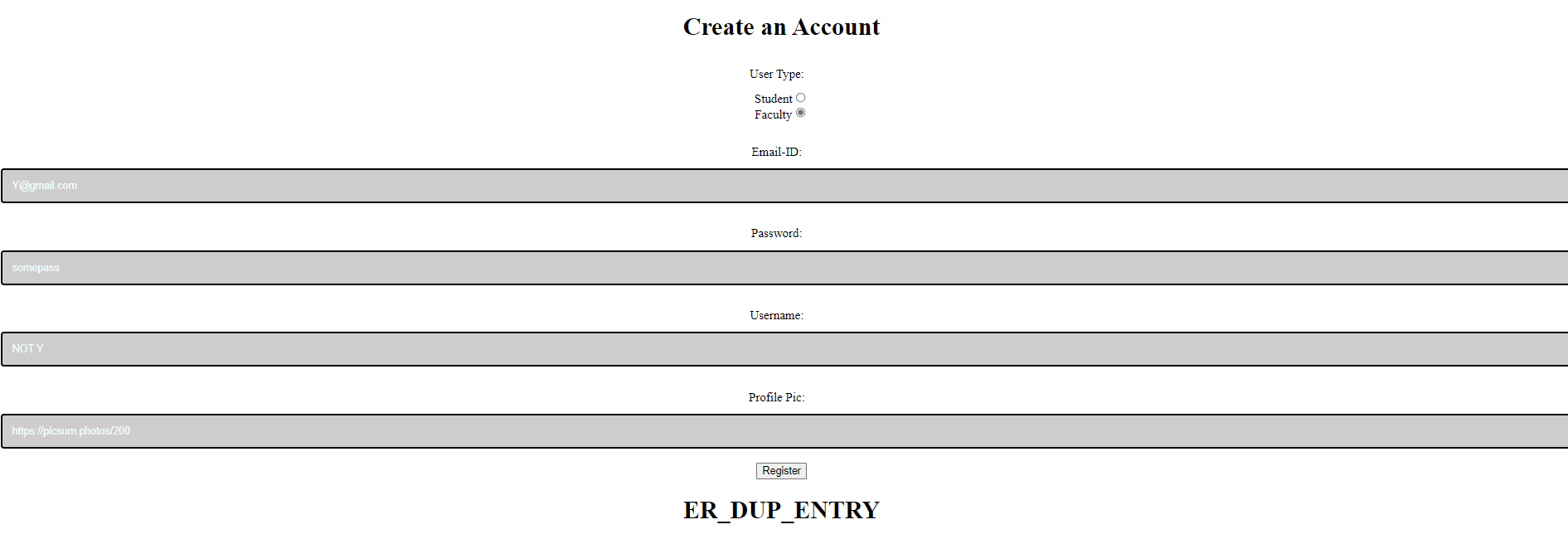
**Output Screenshots:**

**User side functionalities:**

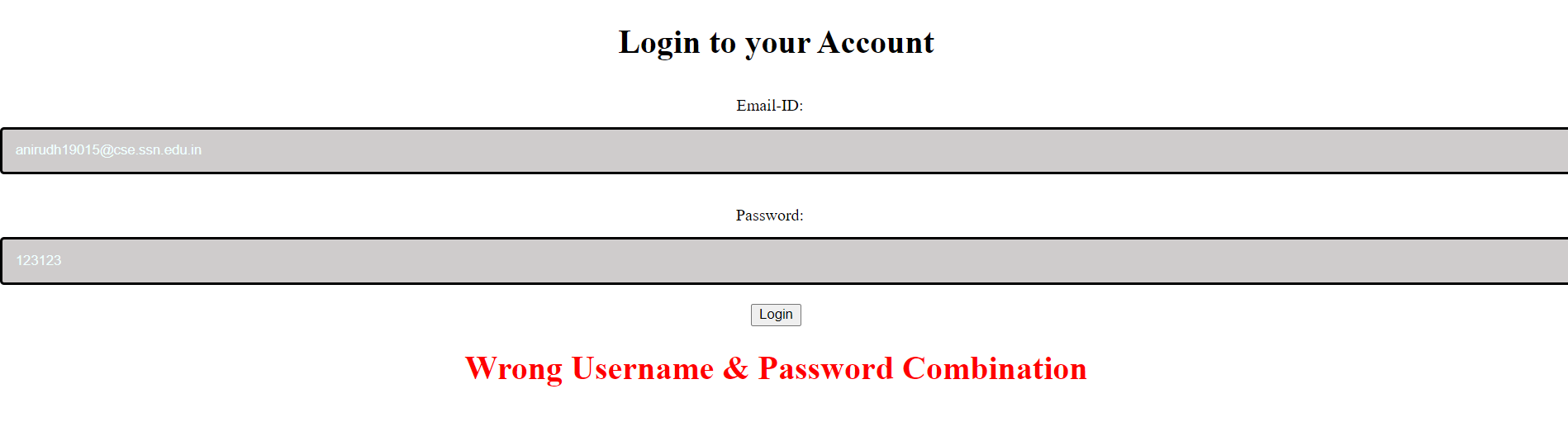
1. VALID REGISTRATION



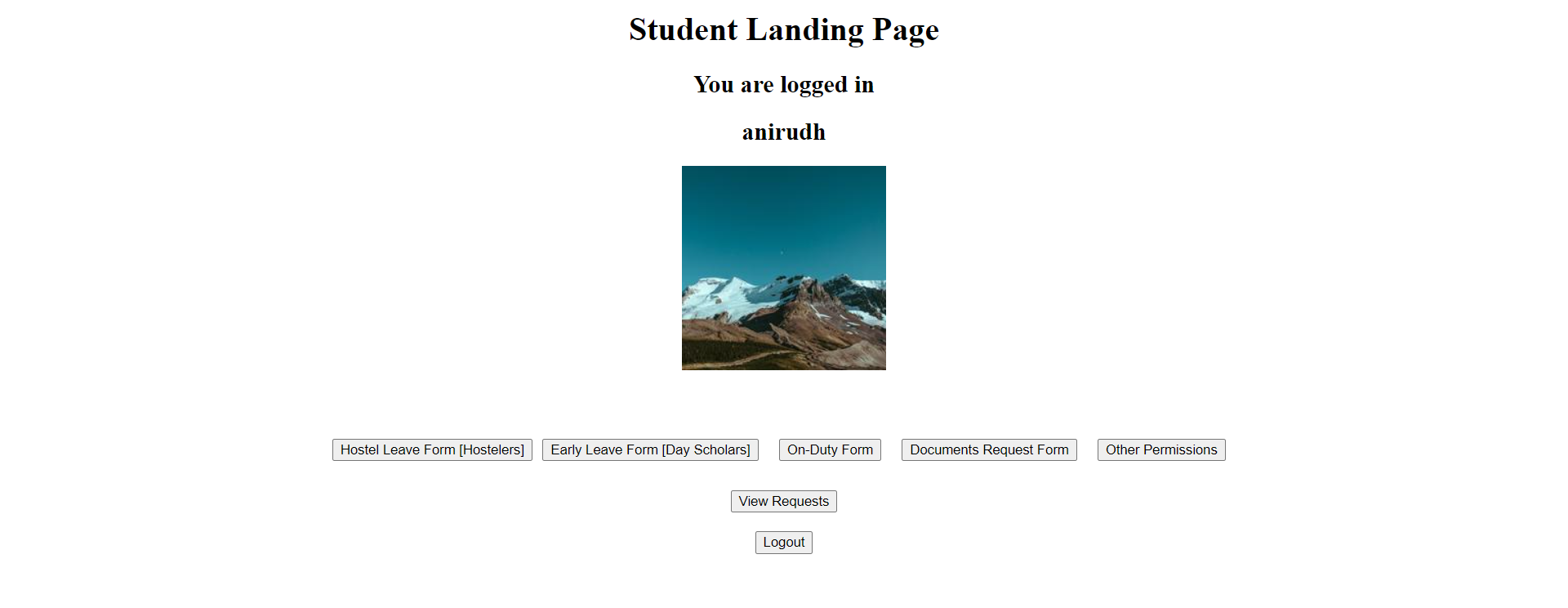
1. INVALID REGISTRATION



1. INVALID LOGIN



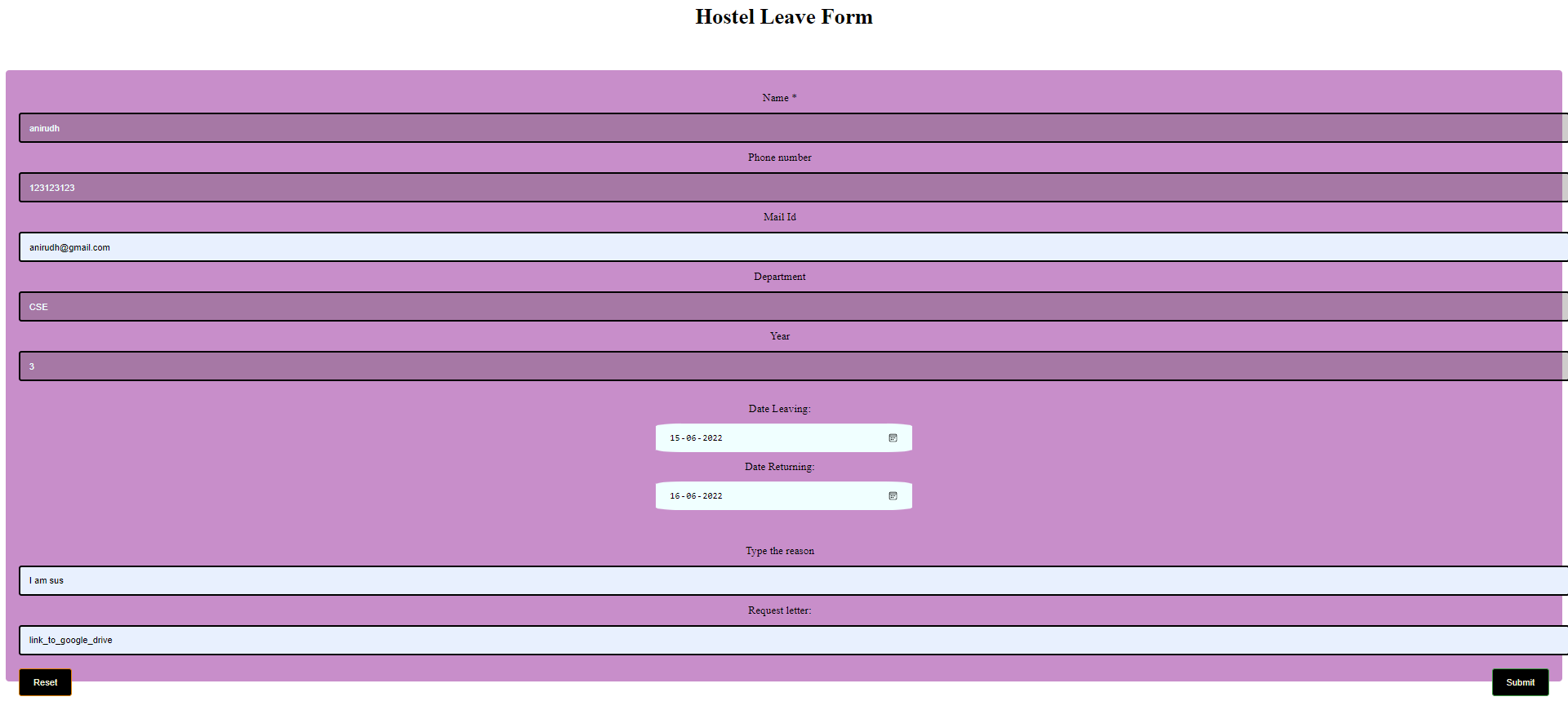
1. VALID LOGIN: STUDENT



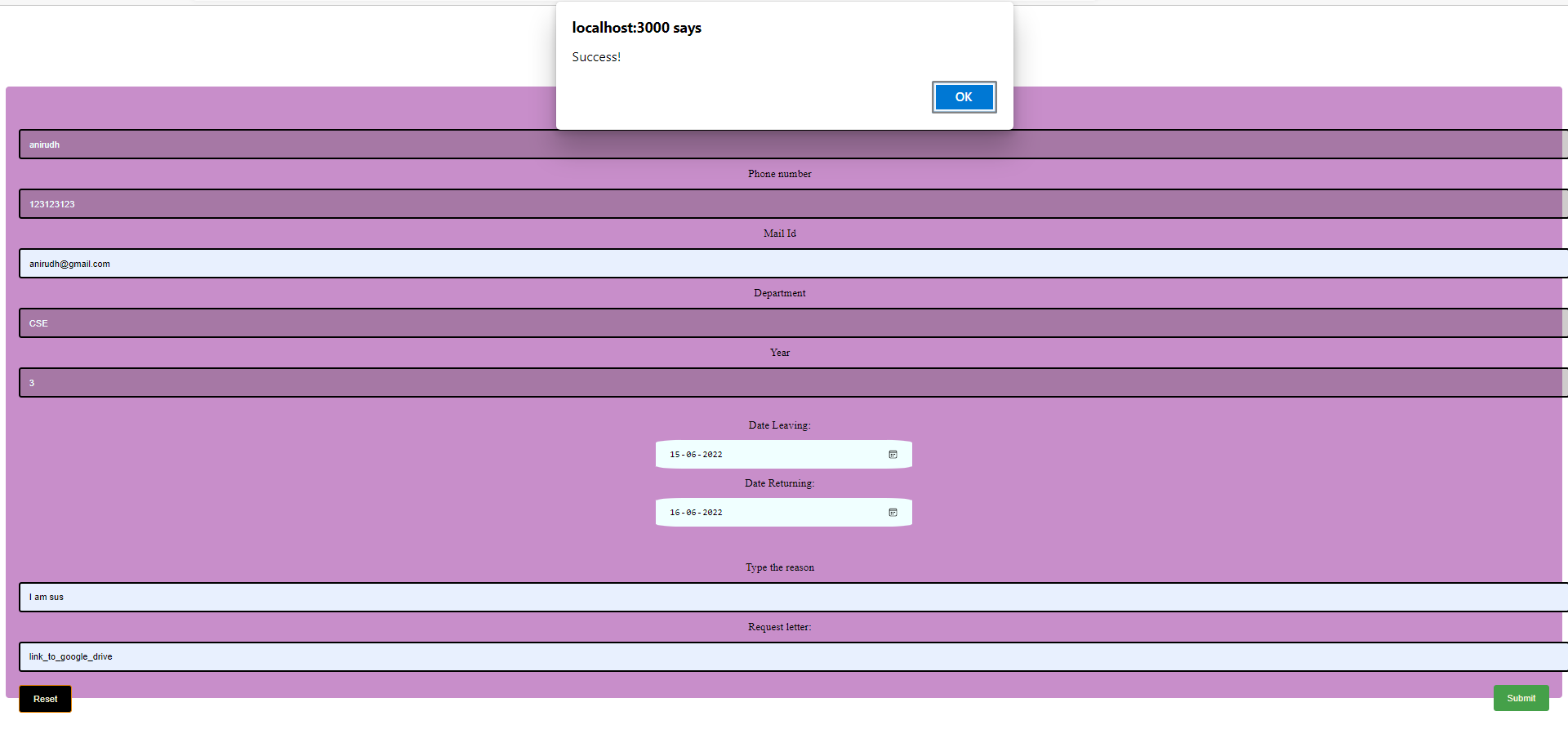
1. VALID LOGIN: FACULTY



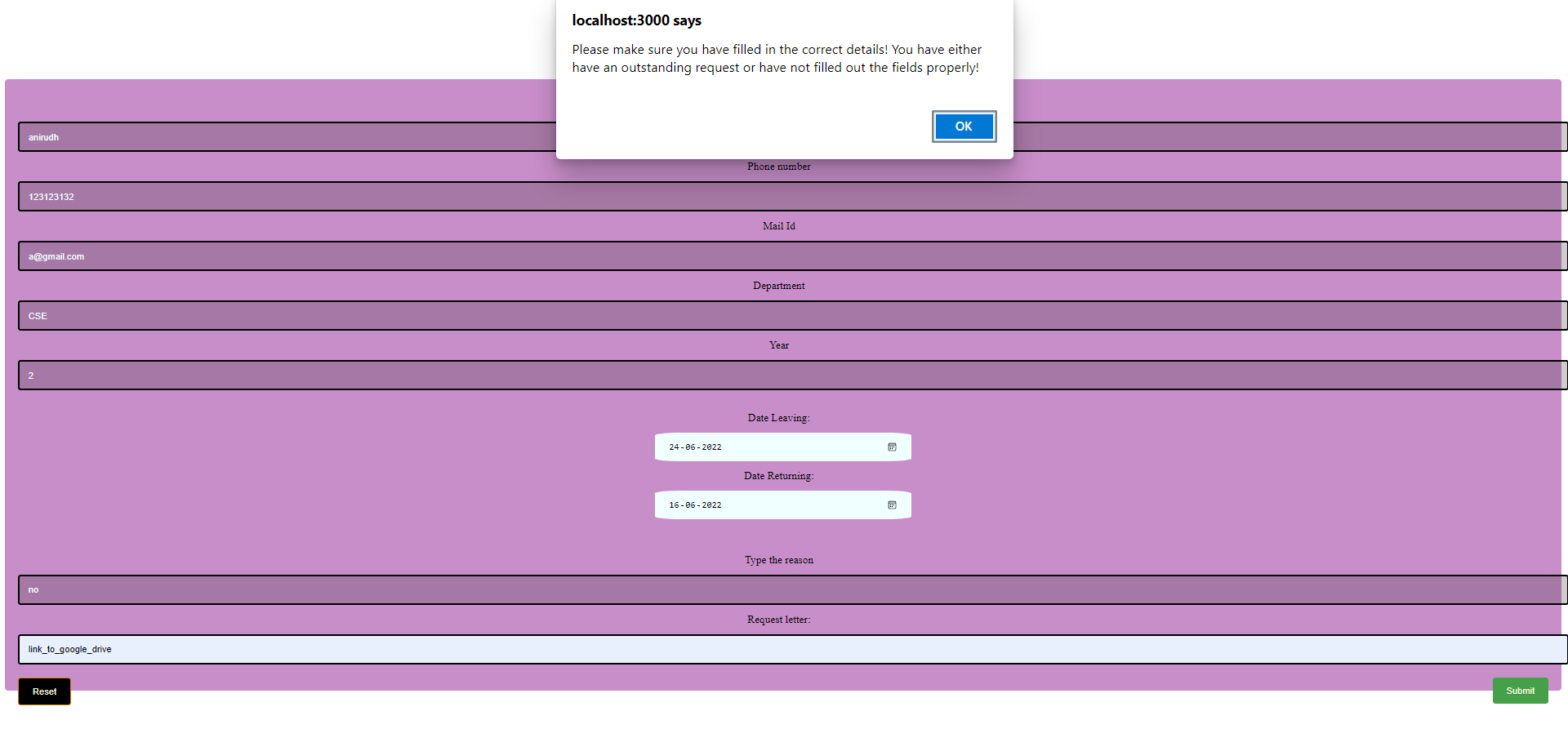
1. STUDENT: HOSTEL LEAVE FORM



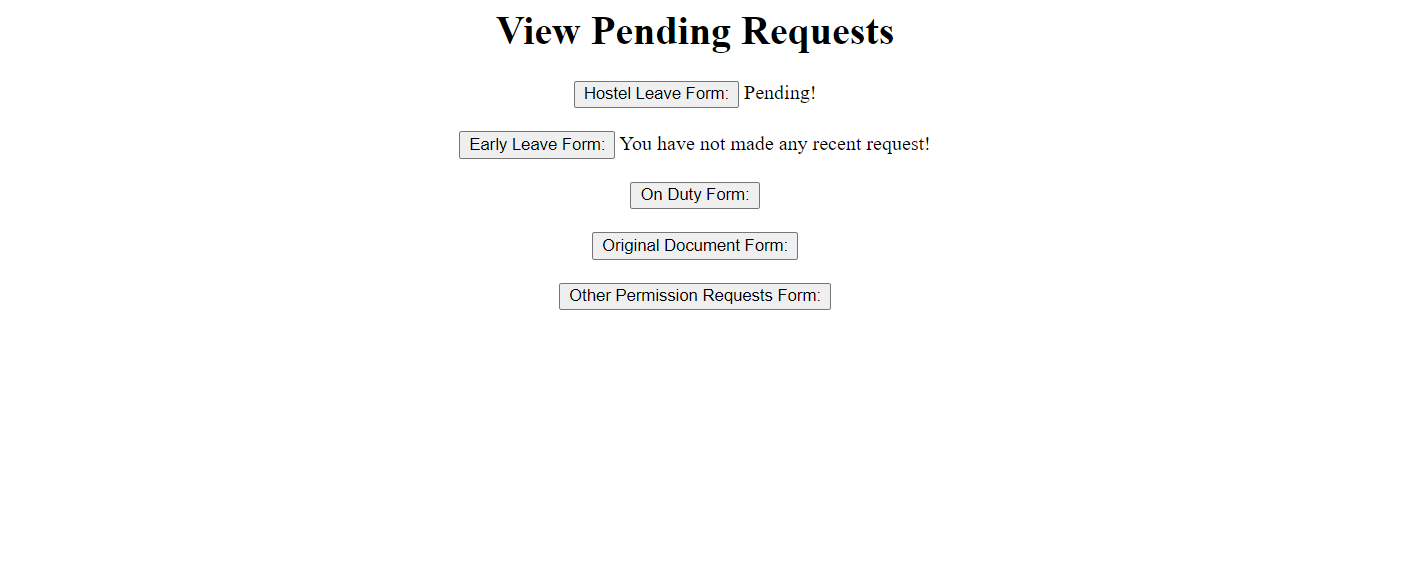
1. STUDENT: SUCCESSFUL FORM SUBMISSION



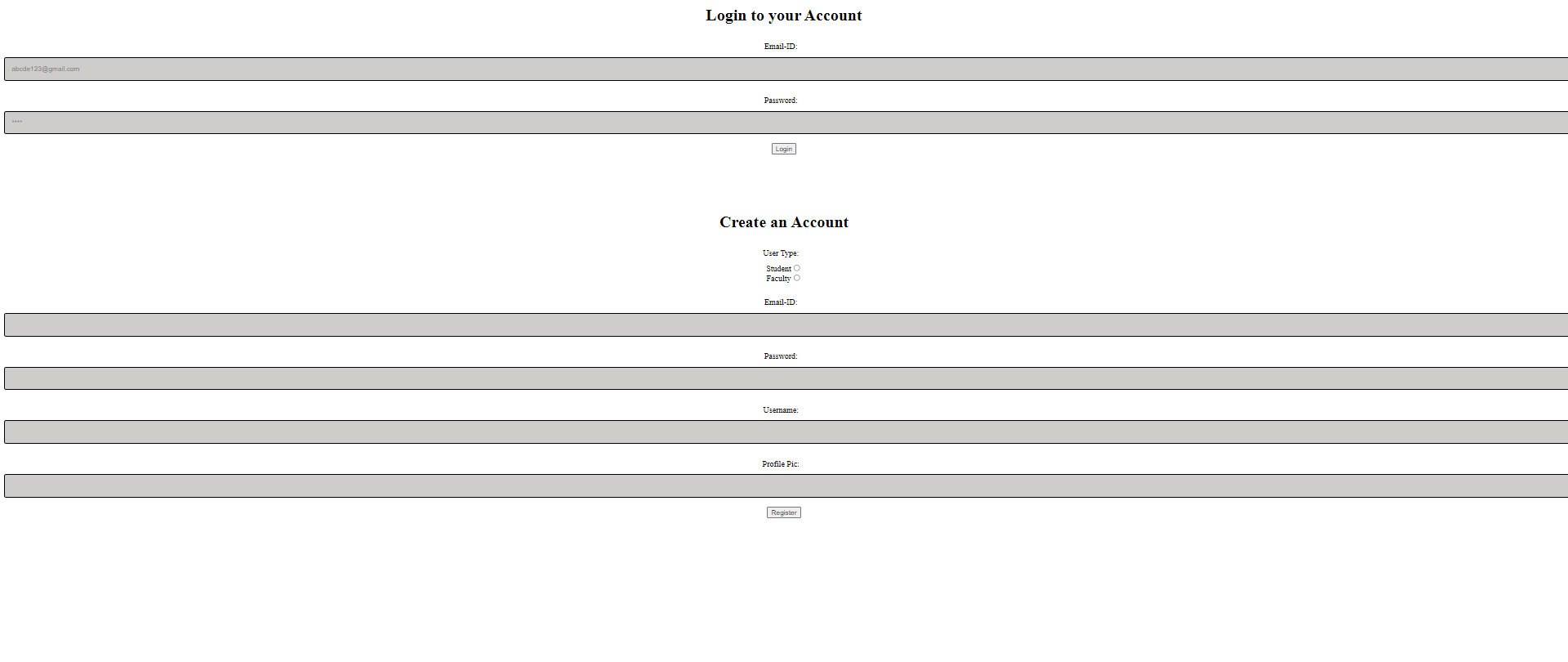
1. STUDENT: UNSUCCESSFUL FORM SUBMISSION



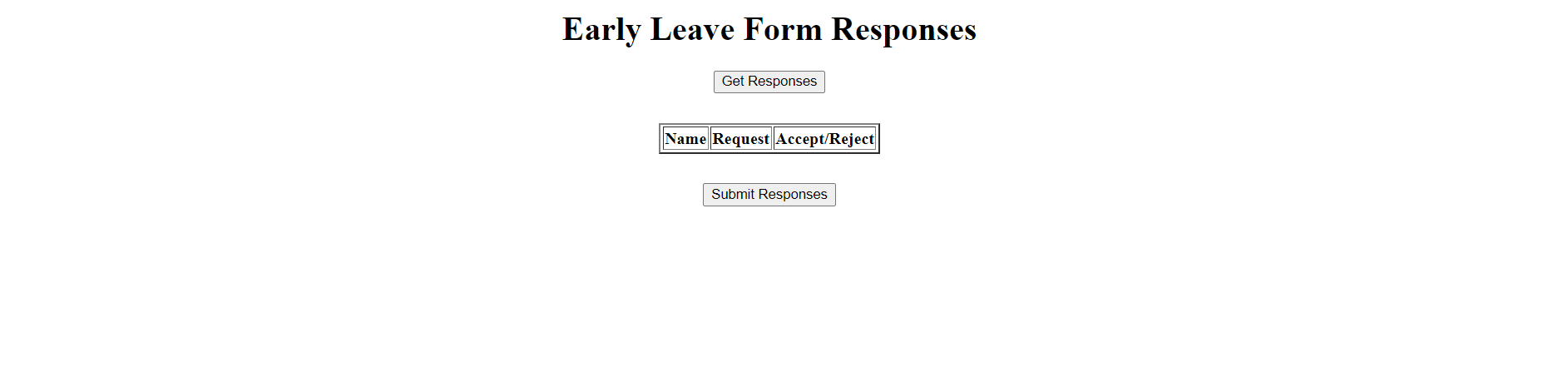
1. STUDENT: VIEW REQUEST STATUS



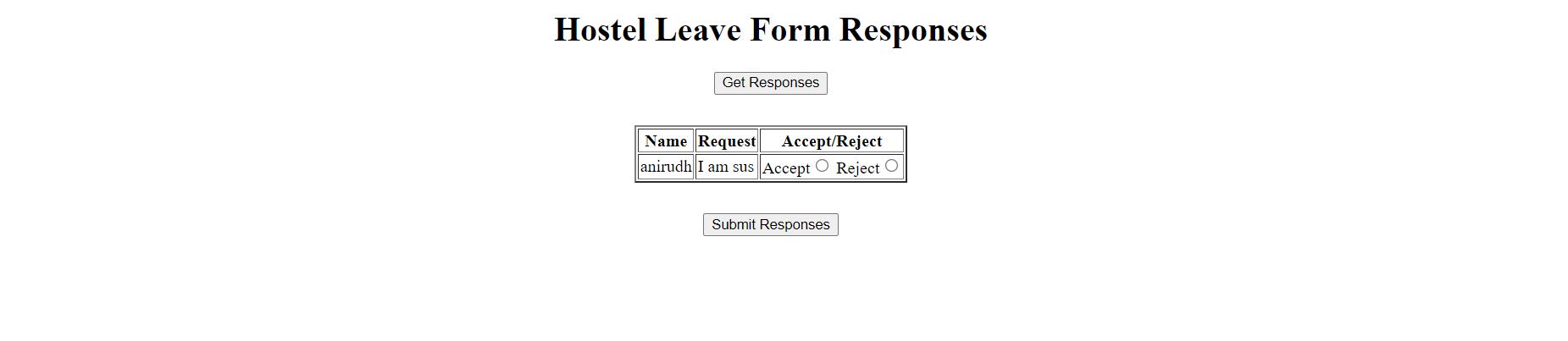
1. LOGOUT



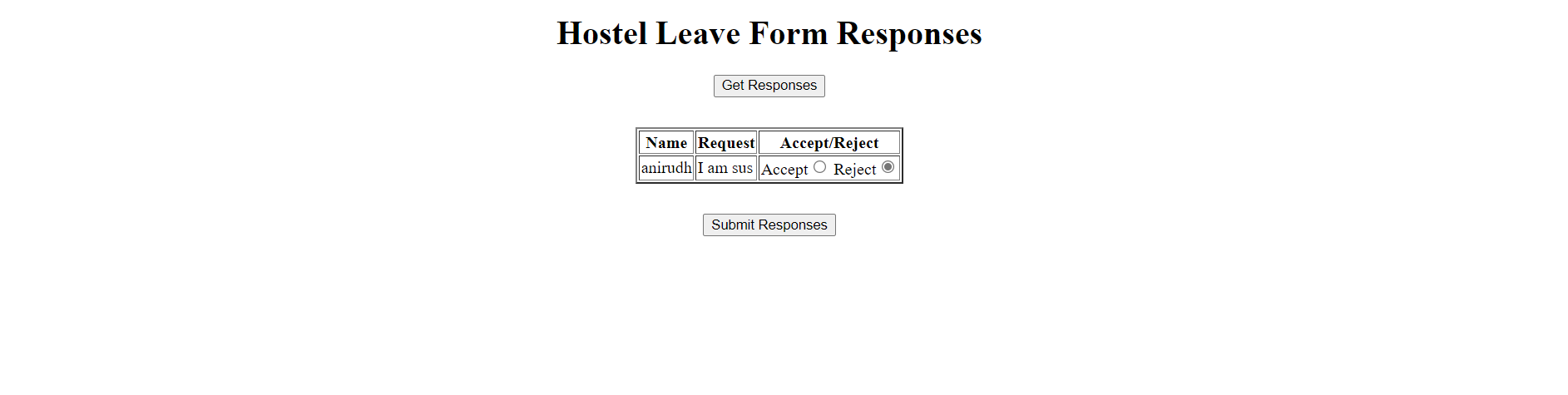
1. FACULTY: VIEW REQUESTS
   1. NO OUTSTANDING REQUESTS



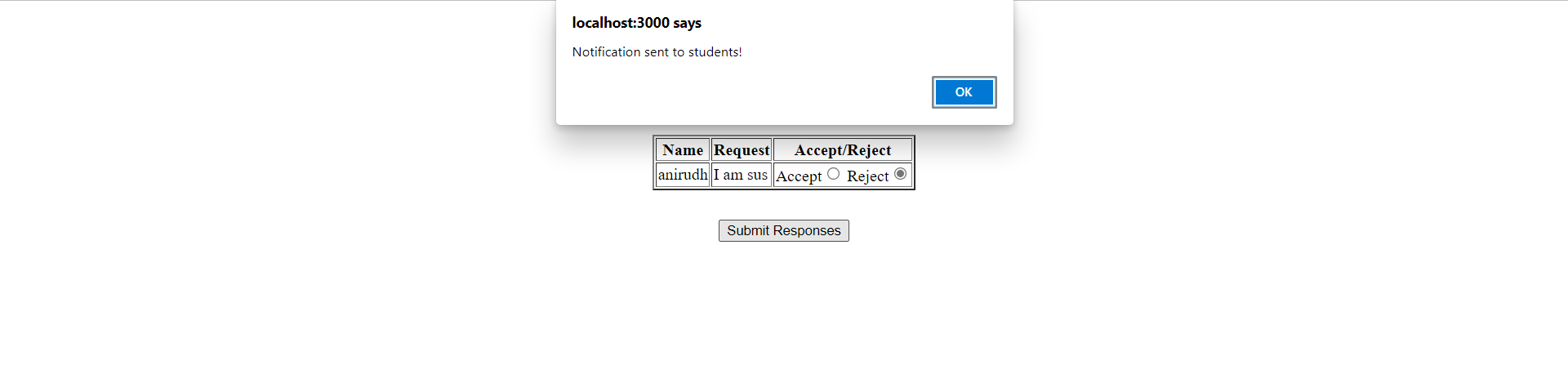
* 1. ONE OR MORE OUTSTANDING REQUESTS



1. FACULTY: ACCEPT OR REJECT OUTSTANDING REQUESTS



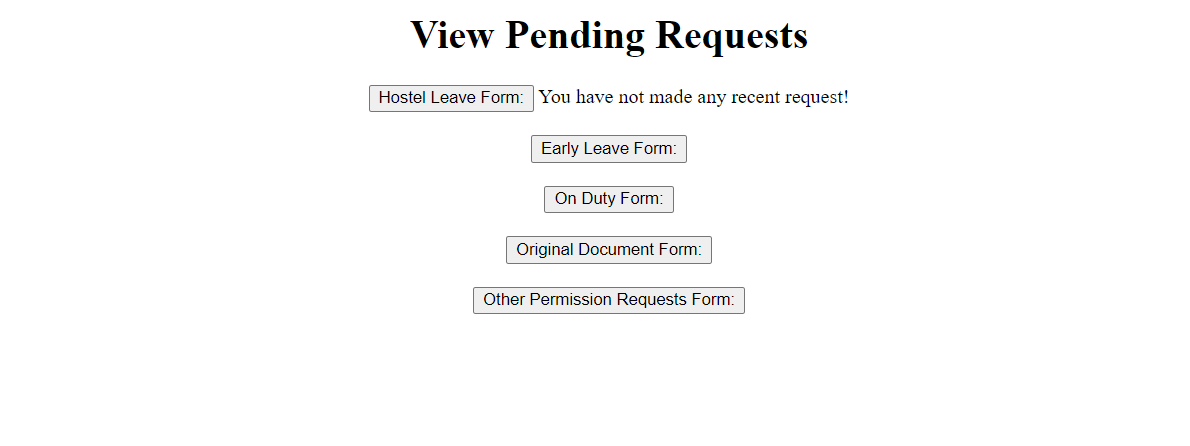
1. FACULTY: SUBMIT DECISIONS



1. VIEW PENDING REQUESTS AFTER UPDATE



1. STUDENT: VIEW PENDING REQUESTS AFTER DECISION



**Permission and Request Online System – Test Cases**

**Ex.No:** 8

**Date:** 08-05-2022

**Aim**

To develop test cases for the Permission and Request Online System and try to improve the design based on the results of testing.

**Identification of Testing Scenarios**

1. Check if Student/Faculty can be authenticated

2. Check if new Student/Faculty can register successfully

3. Check if Student Past Request information is fetched properly

4. Check if New Student Request Type can be selected properly

5. Check if Student can fill and submit the Hostel Leave Form without missing mandatory fields

6. Check if Student can fill and submit the Dayscholar Early Leave Form without missing mandatory fields

7. Check if Student can fill and submit the On-Duty Form without missing mandatory fields

8. Check if Student can fill and submit the Original Documents Form without missing mandatory fields

9. Check if Student can fill and submit the Other Requests Form without missing mandatory fields

10. Check if hostel warden can Accept/Reject the Student Hostel Leave Form and send notification

11. Check if Faculty can Accept/Reject the Student Dayscholar Early Leave Form and send notification

12. Check if Faculty can Accept/Reject the Student On-Duty Form and send notification

13. Check if Department Office can respond to the Student Original Documents Request Form and send notification

14. Check if Faculty can respond to the Student Other Requests Form and send notification

**Tabulation of Test Cases**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST CASE ID** | **TEST SCENARIO** | **TESTING STEPS** | **TEST DATA** | **EXPECTED OUTCOME** | **ACTUAL OUTCOME** | **PASS OR FAIL** |
| **T01** | Student/Faculty:  User authentication | Enter login credentials like email id and password, submit the credentials | Email id: advaith19011@cse.ssn.edu.in  Password:  advaith19011 | Invalid (Wrong Username & Password Combination) | Valid | Fail |
| **T02** | Student/Faculty:  User authentication | Enter login credentials like email id and password, submit the credentials | Email id: advaith19011@cse.ssn.edu.in  Password:  Ani007 | Valid  (Redirects to respective landing page) | Valid | Pass |
| **T03** | Student/Faculty:  New user registration | Enter email id, set password, set user in-app name, user type and profile photo and submit the form. | Email id: advaith19011@cse.ssn.edu.in  Password:  Ani007  Username: Advaith  User type: Student  Profile photo:  (left blank) | Invalid (Form information incomplete ) | Valid | Fail |
| **T04** | Student/Faculty:  New user registration | Enter email id, set password, set user in-app name, user type and profile photo and submit the form. | Email id: advaith19011@cse.ssn.edu.in  Password:  Ani007  Username: Advaith  User type: Student  Profile photo:  (uploaded photo) | Valid (Form registration successful ) | Valid | Pass |
| **T05** | Student:  View Past Request History | Navigate to Student Landing Page, select View history button or logout | Student Request History is Displayed | Valid (Student Request History is Displayed) | Valid | Pass |
| **T06** | Student:  View Past Request History | Navigate to Student Landing Page, select View history button or logout | No failure case for viewing past requests | Valid (No failure case for viewing past requests) | Valid | Pass |
| **T07** | Student:  New Request Type Selection | In Student Landing Page, select New Request button  or logout | Options for New Student Request is Displayed | Valid (Options for New Student Request is Displayed) | Valid | Pass |
| **T08** | Student:  New Request Type Selection | In Student Landing Page, select New Request button or logout | No failure case New Student Request Option | Valid (No failure case New Student Request Option) | Valid | Pass |
| **T09** | Student:  Creating Request for Hostel Leave Form | Click the Hostel Leave Form button. Enter name, phone number, email id, department, year of study, reason for leave, from and to dates for leave, supporting document | Name: Advaith  Phone: 9566063085  Email id: advaith19011@cse.ssn.edu.in  Department: CSE  Year: Third  Reason:  Family Function  From date:  06/06/2022  To date:  08/06/2022  Supporting document:  Invitation letter uploaded | Valid (Request has been sent successfully) | Valid | Pass |
| **T10** | Student:  Creating Request for Hostel Leave Form | Click the Hostel Leave Form button. Enter name, phone number, email id, department, year of study, reason for leave, from and to dates for leave, supporting document | Name: Advaith  Phone: (left empty)  Email id: advaith19011@cse.ssn.edu.in  Department: CSE  Year: Third  Reason:  Family Function  From date:  06/06/2022  To date:  08/06/2022  Supporting document:  Invitation letter uploaded | Invalid (Form information incomplete) | Valid | Fail |
| **T11** | Student:  Creating Request for Dayscholar Early Leave Form | Click the Early Leave Form button. Enter name, phone number, email id, department, reason for leave, date and time for taking leave, supporting document | Name: Advaith  Phone: 9566063085  Email id: advaith19011@cse.ssn.edu.in  Department: CSE  Reason:  Doctor Appointment  Date:  06/03/2022  Time:  11:00 AM  Supporting document:  Appointment Letter | Valid (Request has been sent successfully) | Valid | Pass |
| **T12** | Student:  Creating Request for Dayscholar Early Leave Form | Click the Early Leave Form button. Enter name, phone number, email id, department, reason for leave, date and time for taking leave, supporting document | Name: Advaith  Phone: 9566063085  Email id: advaith19011@cse.ssn.edu.in  Department: CSE  Reason:  (Left empty)  Date:  06/03/2022  Time:  11:00 AM  Supporting document:  (Left empty) | Invalid (Form information incomplete) | Valid | Fail |
| **T13** | Student:  Creating Request for On-Duty Form | Click the On-Duty Form button. Enter name, phone number, email id, department, reason for leave, date and from and to time for taking OD, supporting document | Name: Advaith  Phone: 9566063085  Email id: advaith19011@cse.ssn.edu.in  Department: CSE  Reason:  Doctor Coding Competition  From date:  06/06/2022From Time:  10:00 AM  To time:  3:30 PM  Supporting document:  Participation Invitation | Valid (Request has been sent successfully) | Valid | Pass |
| **T14** | Student:  Creating Request for Dayscholar On-Duty Form | Click the On-Duty Form button. Enter name, phone number, email id, department, reason for leave, date and from and to time for taking OD, supporting document | Name: Advaith  Phone: 9566063085  Email id: advaith19011@cse.ssn.edu.in  Department: CSE  Reason:  Coding Competition Date:  06/03/2022  Time:  11:00 AM  Supporting document:  (Left empty) | Invalid (Form information incomplete) | Valid | Fail |
| **T15** | Student:  Creating Request for Original Documents Form | Click the Original Documents Request Form button. Enter name, phone number, email id, Year, department, reason, date and time of request, supporting document | Name: Advaith  Phone: 9566063085  Email id: advaith19011@cse.ssn.edu.in  Department: CSE  Year:  Third  Reason:  Bonafide Certificate  Request Date:  25/05/2022  Request Time:  9:00 AM  Supporting document:  Internship Accepatance email | Valid (Request has been sent successfully) | Valid | Pass |
| **T16** | Student:  Creating Request for Original Documents Form | Click the Original Documents Request Form button. Enter name, phone number, email id, Year, department, reason, date and time of request, supporting document | Name: Advaith  Phone: 9566063085  Email id: (left empty)  Department: CSE  Year:  Third  Reason:  Bonafide Certificate  Request Date:  25/05/2022  Request Time:  9:00 AM  Supporting document:  Internship Accepatance email | Invalid (Form information incomplete) | Valid | Fail |
| **T17** | Student:  Creating Request for Other Requests Form | Click the Original Documents Request Form button. Enter name, phone number, student email id, faculty email id, Year, department, reason, supporting document | Name: Advaith  Phone: 9566063085  Email id: advaith19011@cse.ssn.edu.in  Faculty Email id: bhuvanaj@cse.ssn.edu.in  Department: CSE  Year:  Third  Reason:  CAT Marks discrepancy  Supporting document:  Screenshot of CAT marks | Valid (Request has been sent successfully) | Valid | Pass |
| **T18** | Student:  Creating Request for Other Requests Form | Click the Original Documents Request Form button. Enter name, phone number, student email id, faculty email id, Year, department, reason, supporting document | Name: Advaith  Phone: 9566063085  Email id: advaith19011@cse.ssn.edu.in  Faculty Email id: (left empty)  Department: CSE  Year:  Third  Reason:  CAT Marks discrepancy  Supporting document:  Screenshot of CAT marks | Invalid (Form information incomplete) | Valid | Fail |
| **T19** | Hostel Warden:  Respond to Student Hostel Leave Form Request | Click the request. Select Accept/Reject request. Click the Submit Responses button | On click Submit Responses | Valid (Notification sent to students!) | Valid | Pass |
| **T20** | Faculty:  Respond to Dayscholar Early Leave Form Request | Click the request. Select Accept/Reject request. Click the Submit Responses button | On click Submit Responses | Valid (Notification sent to students!) | Valid | Pass |
| **T21** | Faculty:  Respond to On-Duty Form Request | Click the request. Select Accept/Reject request. Click the Submit Responses button | On click Submit Responses | Valid (Notification sent to students!) | Valid | Pass |
| **T22** | Department Office:  Respond to Original Documents Request Form Request | Click the request. Select Accept/Reject request. Click the Submit Responses button | On click Submit Responses | Valid (Notification sent to students!) | Valid | Pass |
| **T23** | Faculty:  Respond to Other Requests Form Request | Click the request. Select Accept/Reject request. Click the Submit Responses button | On click Submit Responses | Valid (Notification sent to students!) | Valid | Pass |