Software Requirements Specification

for

DigiCampus

**Version 2.0**

**Prepared by**

**Group #: 1 Group Name: Achievers**

|  |  |  |
| --- | --- | --- |
| **Dishay Mehta** | **200341** | **dishaymehta952@gmail.com** |
| **Abhishek Pardhi** | **200026** | **abhipardhi326@gmail.com** |
| **Samarth Arora** | **200849** | **samartharora03@gmail.com** |
| **Ankur Kumar** | **200140** | **ankurrk04@gmail.com** |
| **Shashwat Gupta** | **200923** | **guptashashwatme@gmail.com** |
| **Ananya Agrawal** | **200117** | **ananyaagrawal704@gmail.com** |
| **Girik Maskara** | **200387** | **girikmaskara552002@gmail.com** |
| **Sarthak Kohli** | **200886** | **sarthak811kohli@gmail.com** |
| **Aayush Kumar** | **200008** | **aayushk0072@gmail.com** |
| **Aryan Vora** | **200204** | **aryanvora23@gmail.com** |

|  |  |
| --- | --- |
| **Course:** | **CS253** |
| **Mentor TA:** | ***Aman*** |
| **Date:** | **27/04/22** |

Content

**Contents ii**

**Revisions ii**

**1** **Introduction 1**

1.1 Product Scope 1

1.2 Intended Audience and Document Overview 1

1.3 Definitions, Acronyms and Abbreviations 1

1.4 Document Conventions 1

1.5 References and Acknowledgments 2

**2** **Overall Description 2**

2.1 Product Overview 2

2.2 Product Functionality 3

2.3 Design and Implementation Constraints 3

2.4 Assumptions and Dependencies 3

**3** **Specific Requirements 4**

3.1 External Interface Requirements 4

3.2 Functional Requirements 4

3.3 Use Case Model 5

**4** **Other Non-functional Requirements 6**

4.1 Performance Requirements 6

4.2 Safety and Security Requirements 6

4.3 Software Quality Attributes 6

**5** **Other Requirements 7**

**Appendix A – Data Dictionary 8**

**Appendix B - Group Log 9**

**Revisions**

| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| --- | --- | --- | --- |
| 1.0 | Dishay Mehta,Abhishek Pardhi,Samarth Arora,Ankur Kumar,Shashwat Gupta,Ananya Agarwal,Girik Maskara,Sarthak Kohli,Aayush Kumar,Aryan Vora | SRS Document created. | 01/02/22 |
| 2.0 | Dishay Mehta,Abhishek Pardhi,Samarth Arora,Ankur Kumar,Shashwat Gupta,Ananya Agarwal,Girik Maskara,Sarthak Kohli,Aayush Kumar,Aryan Vora | SRS Document updated. | 27/04/22 |

# 

# Introduction

## Product Scope

DigiCampus is an application designed to be used by IITK students. It allows two main functionalities, a digitalized system to track entry and exit from hall gates, and a digital register that keeps track of the extras that can be bought at the mess.

The application aims to make hall entry and exit smoother, faster, and hassle-free, while also bringing transparency to the system that is currently in place in the hall messes.

DigiCampus will have three profile classes, students, security guards, and mess managers. These profiles will get access to relevant information and features.

## Intended Audience and Document Overview

This document is mainly intended for developers who might want to add additional functionalities, optimize the current ones, or re-use these functionalities in their own projects. It can also be used by testers and users to understand how to use the system correctly, and in what situations the system is to be used.

It is important to first understand what the system does, and so it is recommended that the audience first reads the Overall Description of the system and then proceed to the Specific Requirements.

The system is currently intended for the IIT Kanpur campus population.

## Definitions, Acronyms and Abbreviations

* **CC ID**: Computer Centre ID
* **ID**: Identity
* **SIS Guards**: Security Guards in the IIT Kanpur campus.

## Document Conventions

**Formatting Conventions:**

* Using black, Arial with a font size of 11 for text.
* Using white, bolded Arial with a font size of 18 and a grey background for headings.
* Using black, bolded Arial with a font size of 14 for subheadings.
* Using black, bolded Arial with a font size of 11 for sub subheadings.
* Using black, filled bullets to form bulleted lists.
* Using blue, Arial with a font size of 11 for updates after Version 1.0.

**Naming Conventions:**

* Use Case Models named as U#
* Functional Requirements named as F#

## References and Acknowledgments

* <https://app.creately.com/diagram>: Used to make the diagrams used in the document
* <https://www.figma.com/files/recent?fuid=1070397452285589933>: for UI design

# Overall Description

## Product Overview

**Origin of the Product:**

**(a) Hall Entry and Exit:**

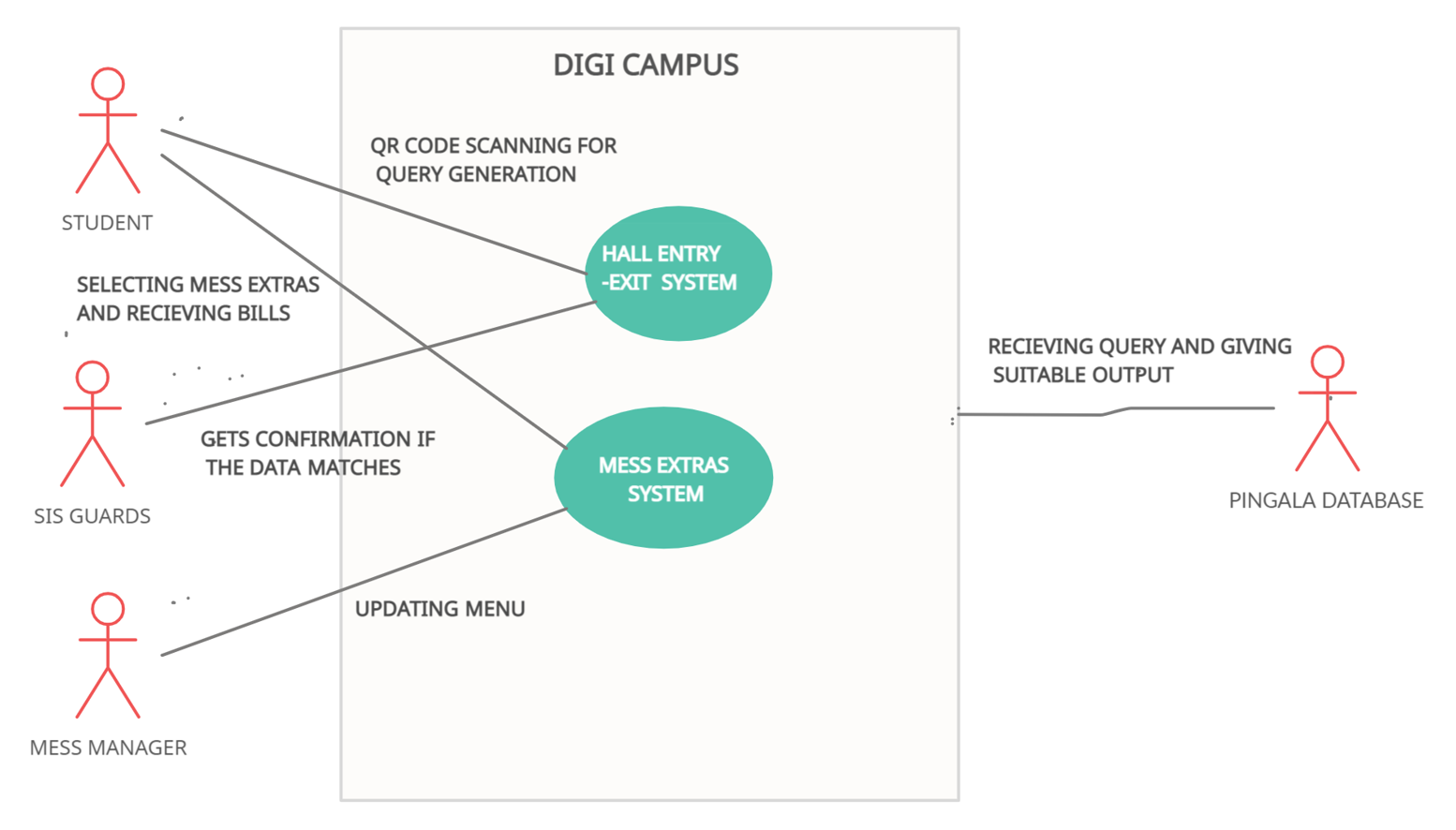
The current system for students entering into Halls is to submit their ID cards and manually enter their details into the Hall entry register. The issue with this system is false data entry and crowd gathering at Hall closing time. Our software replaces the current system with a digital database entry to provide a contactless and hassle-free movement between halls.

**(B) Mess Extras:**

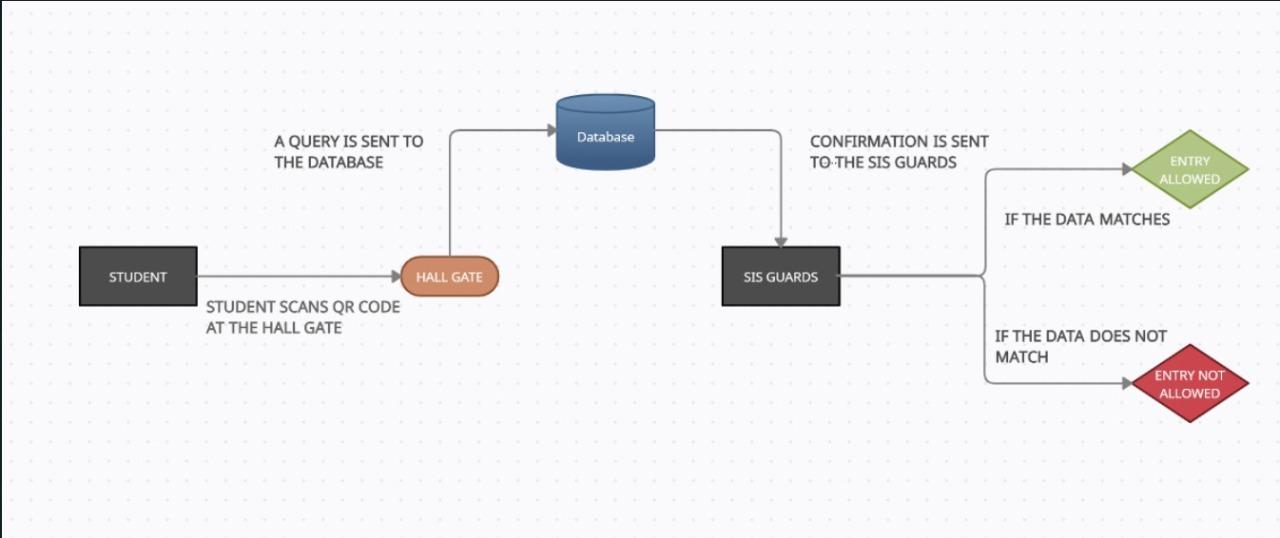
In the mess, two types of items are served: Regulars and Extras. The Extras are optional and their value is added to the mess bill. Currently, the way to obtain these extras is by making an entry at their respective counters. The issue with this system is there is no guarantee that the student has entered the correct details and there is no transparency/record the student has of the items he/she has ordered. Replacing this system, our software will handle digital entries at all the mess counters and will be stored in a database that can be viewed by the student. This provides transparency to the students and protects both the mess managers and students from potential frauds.

This Digitalization will make the whole process hassle-free.

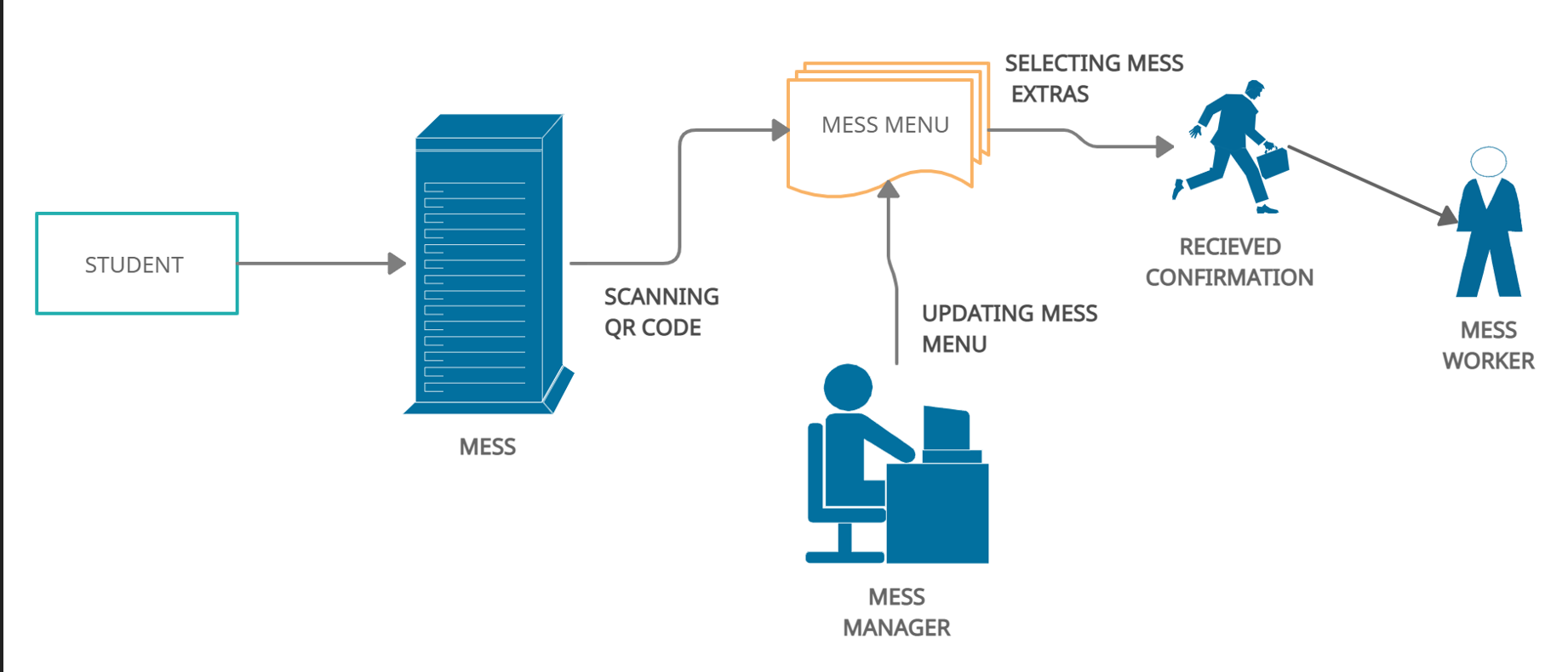
The system environment looks like this:



**Overview Of Hall Entry/Exit System:**

****

**Overview Of Mess Extras System:**

****

## Product Functionality

* The system will allow the student to scan the hall’s QR code at the hall entry/exit which will update the real-time location of the student in the student database.
* The system allows the security to access the dynamic list of students currently visiting a hall, the security also has the feature to contact the students.
* The system will allow the students to select/pre-order their Mess Extras and will be provided with a detailed mess bill.

## Design and Implementation Constraints

* The Student Database will be given by Pingala Servers which requires the permission of authorities.
* The initial set-up will be done by our engineers and we will be maintaining the system for the first 3 months to cater to bugs if they arise. Further maintenance of the product is the customer's responsibility.
* Language requirements: The software will use only the English language.

## Assumptions and Dependencies

* We assume that the authorities will provide us with the student database.
* We assume that all students are having a smartphone or a laptop with a working internet connection through which they can access our system.
* We assume that security guards will be able to use our product or the authorities will provide them assistance to make them comfortable with the User Interface.

# Specific Requirements

## External Interface Requirements

### User Interfaces

The user interface for the software shall be compatible with any browser such as Internet Explorer, Mozilla, or Netscape Navigator by which the user can access the system. We hope that our software is compatible with all Operating systems: Android, Windows Linux, etc, as it is a web-based application, and therefore will not have many OS dependencies.

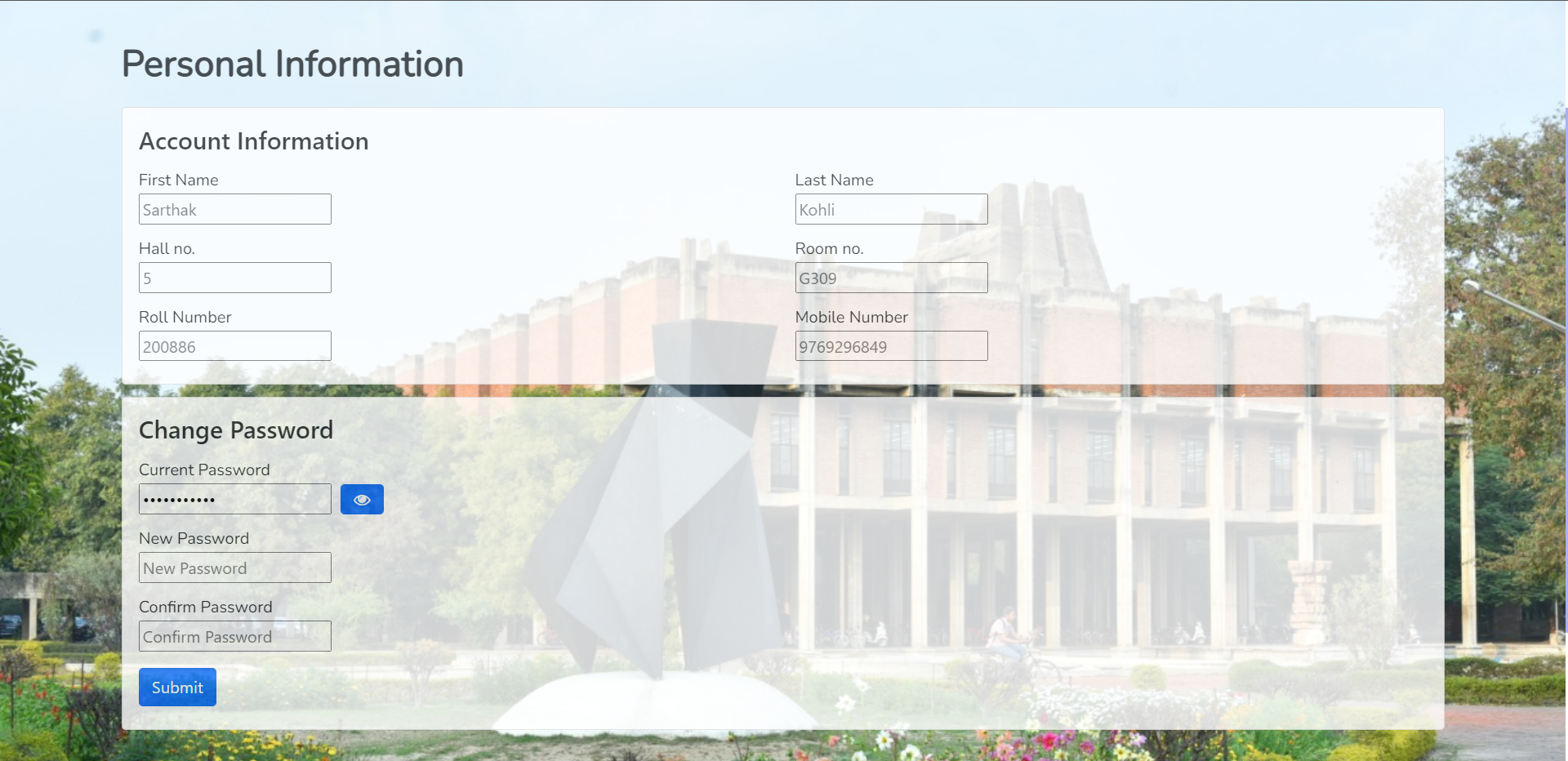
The user interface (Frontend) shall be implemented using HTML, CSS, and JavaScript.

**Features:**

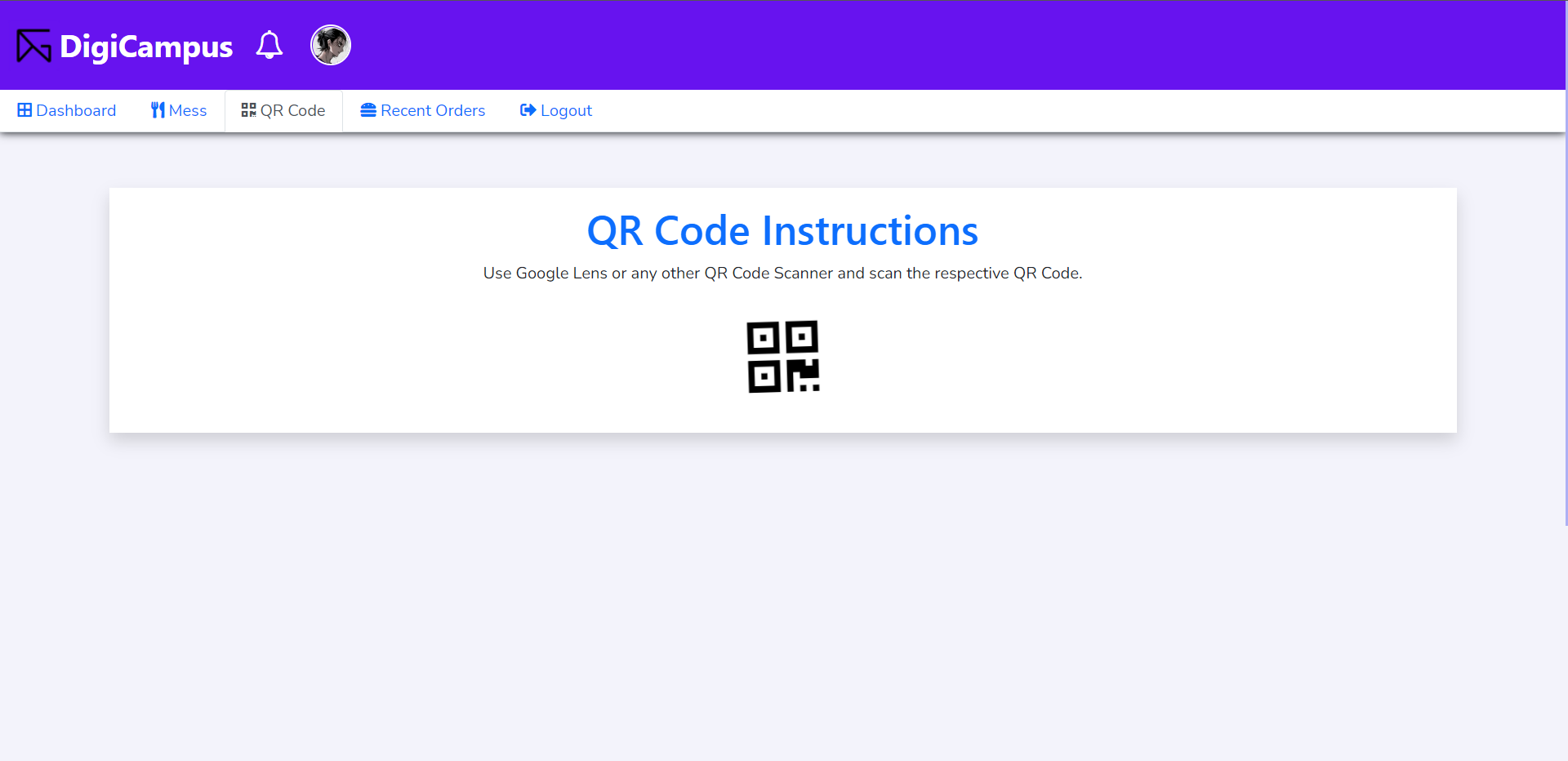
1. The system shall provide a uniform look and feel between all the web pages.
2. The system shall provide a photo for the user for his profile, as well as other interface visual aids like the hall photos and mess photos.
3. The system shall provide the use of icons and toolbars.

The final UI looks like this:

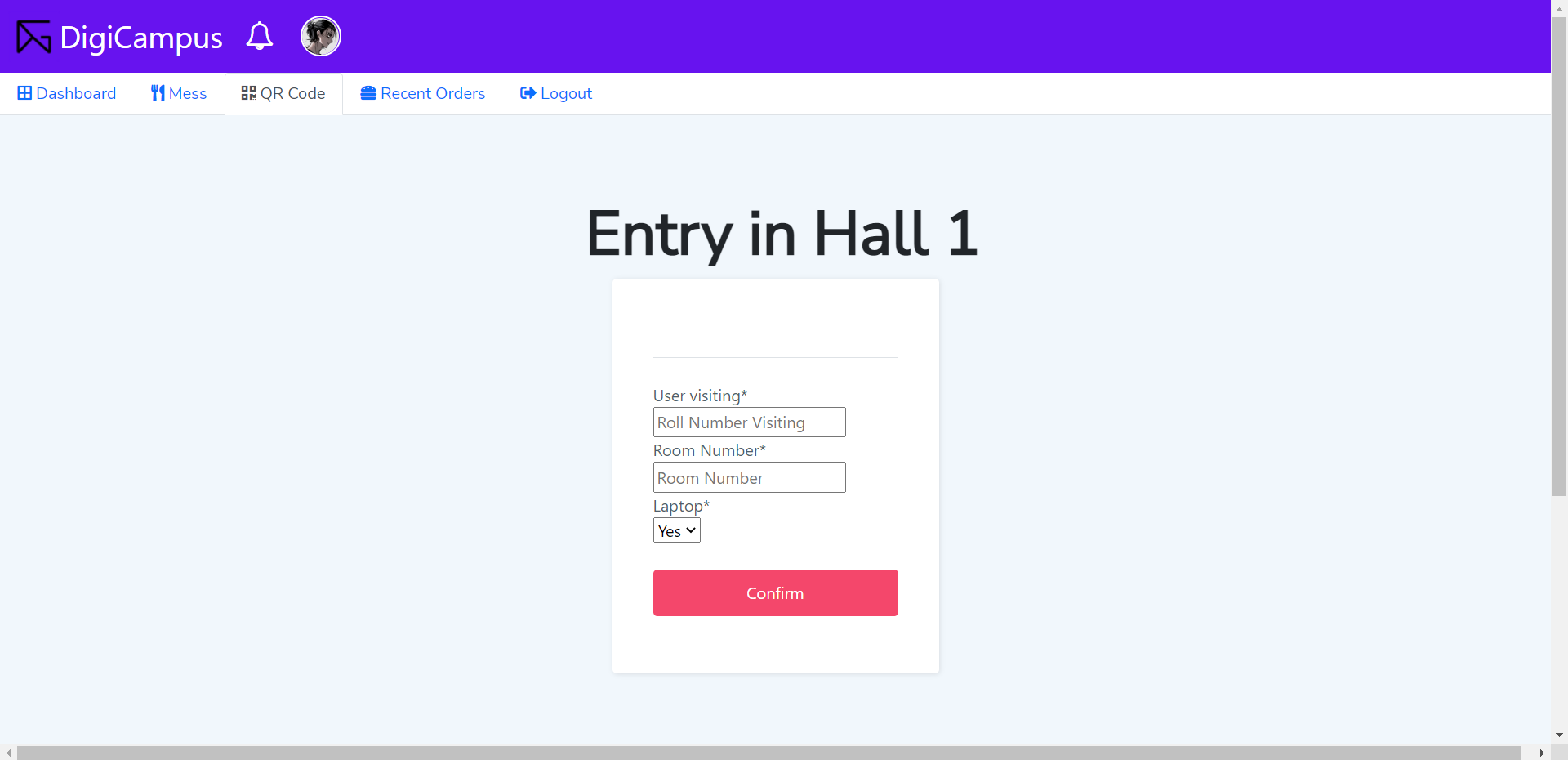
* **Student Profile:**



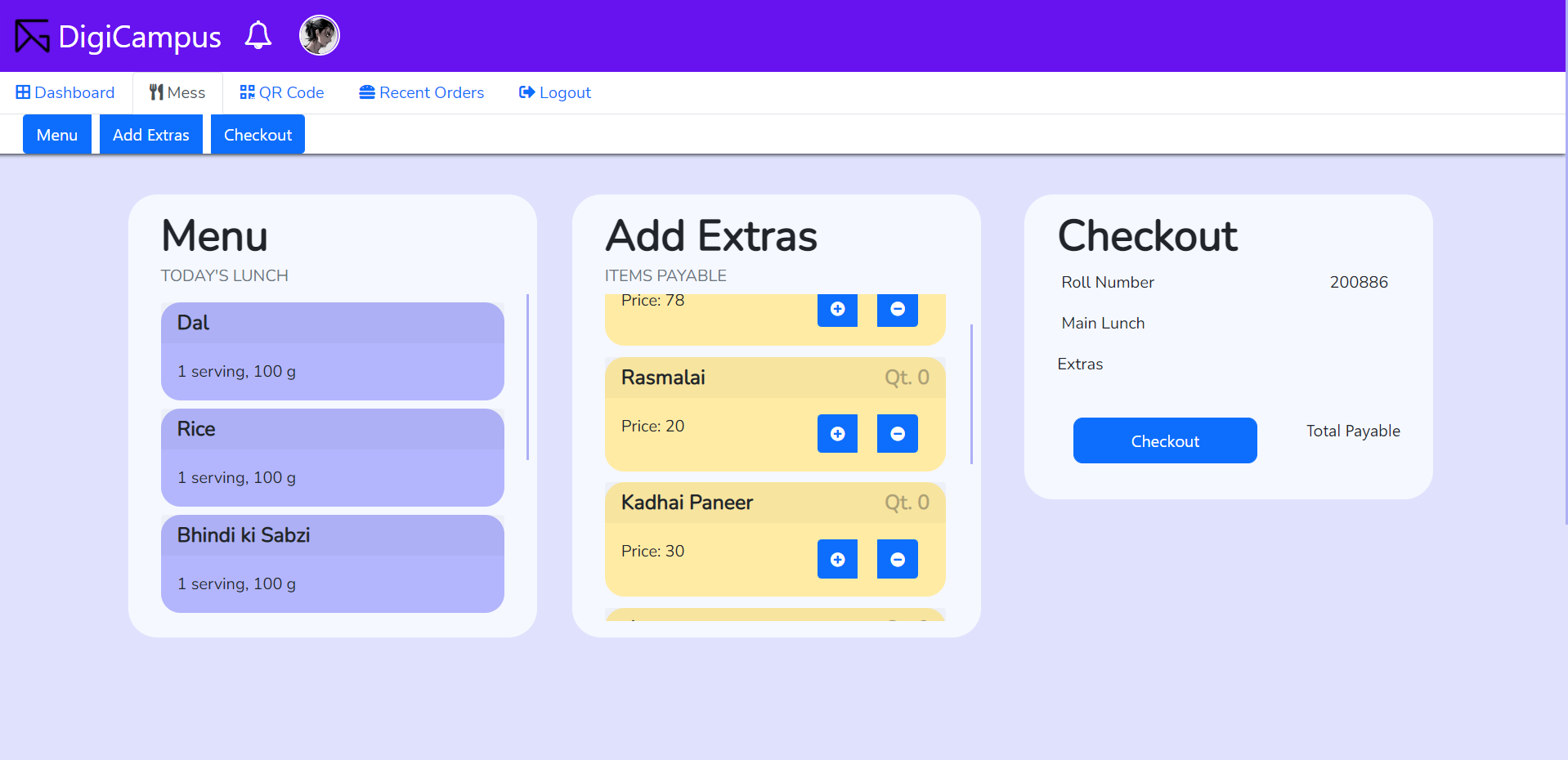
* **Scanning Code At The Gate**



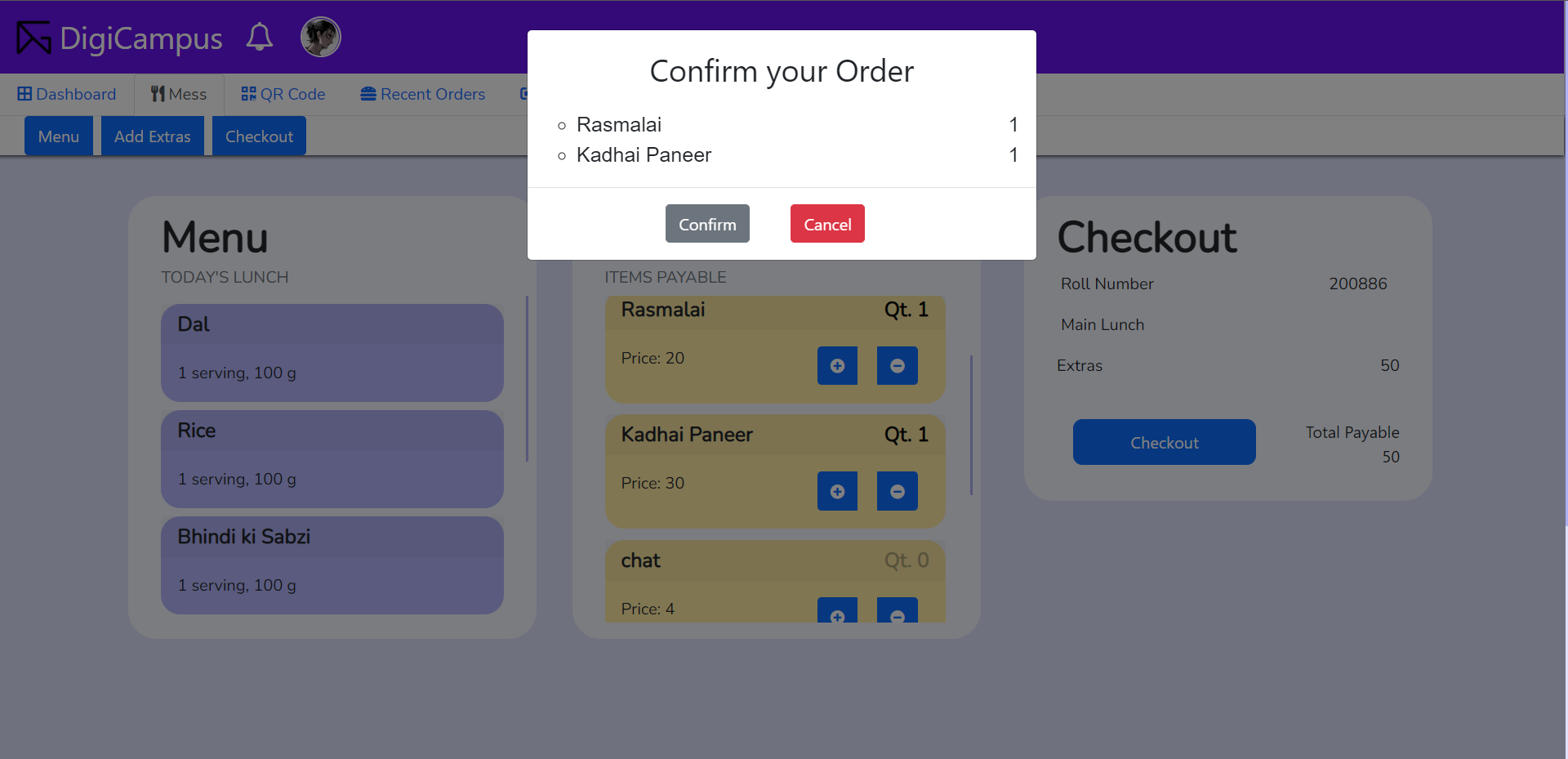
* **Entry/Exit Details**



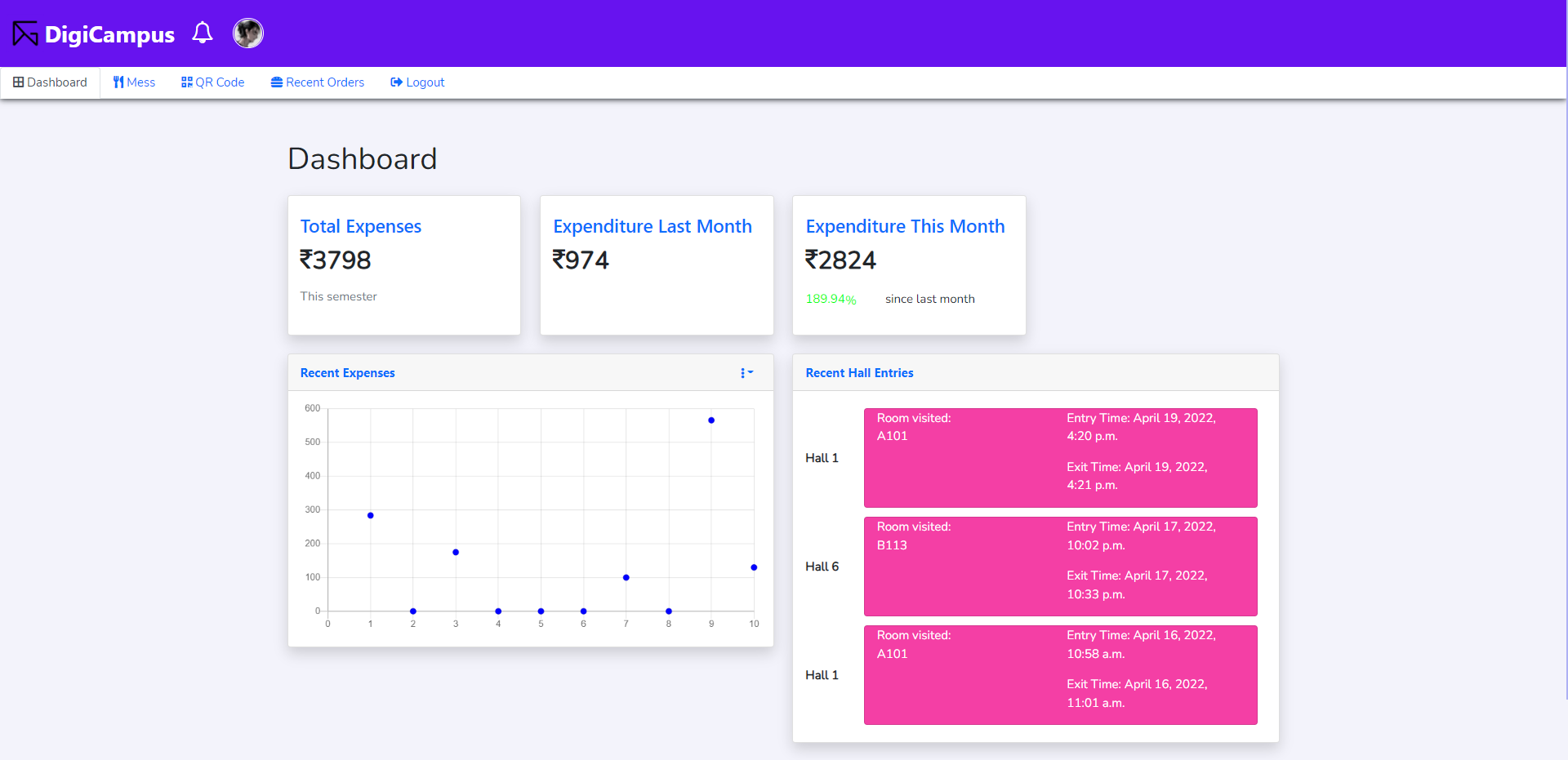
* **Mess Menu and Extras**



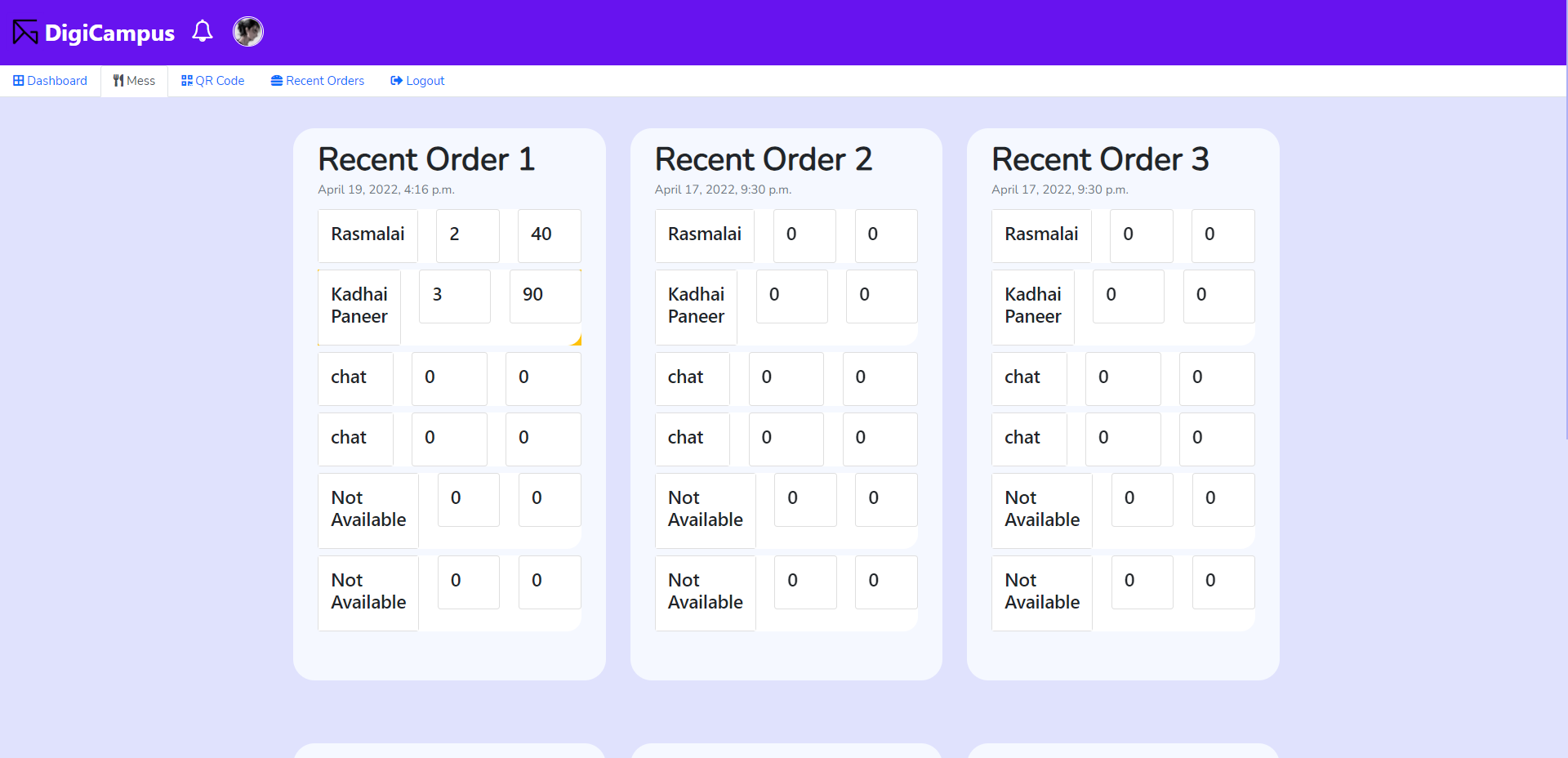
* **Confirmation Page**



* **Student Dashboard**



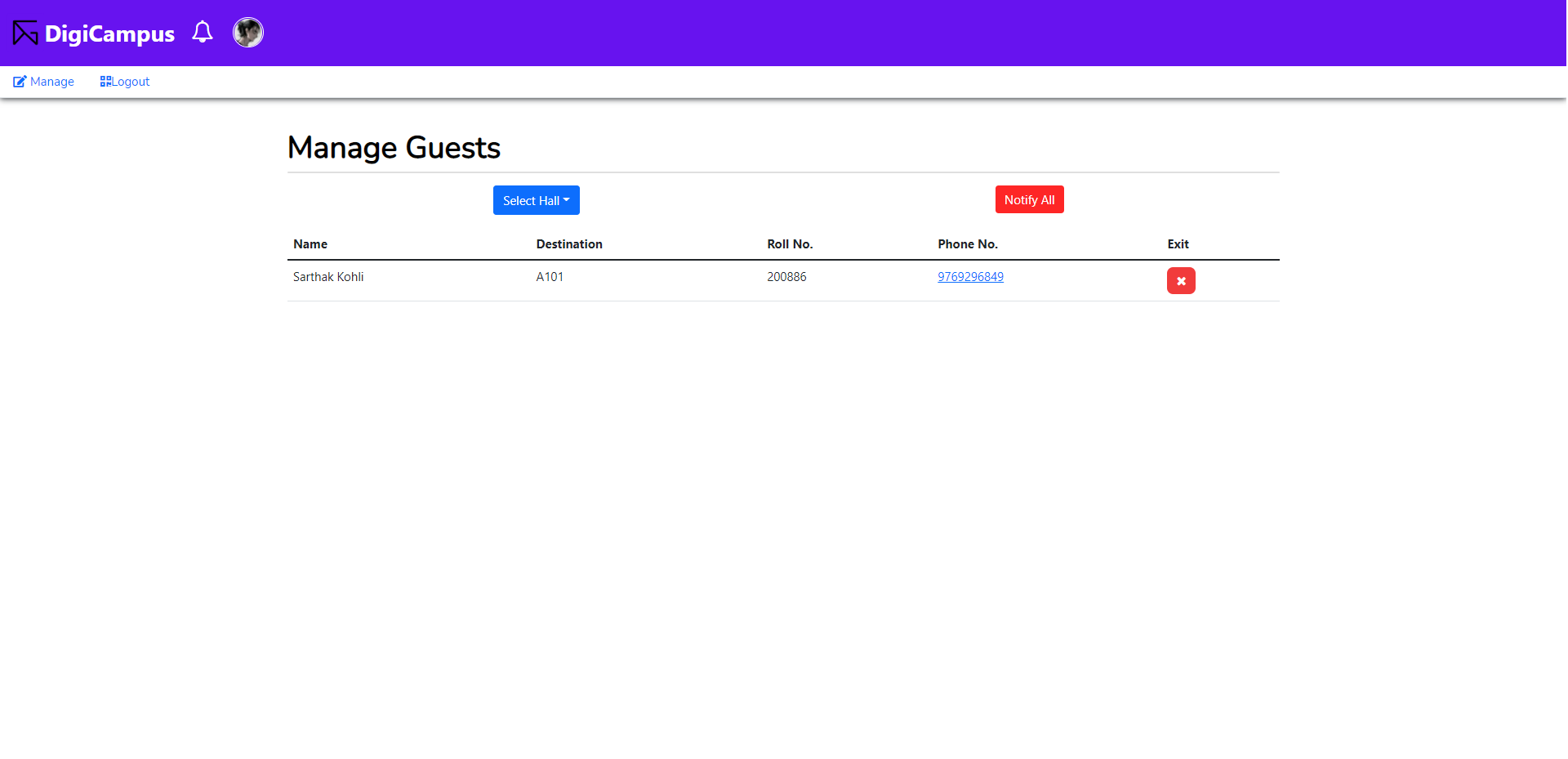
* **Order List**



* **Mess Updation:**



* **Security Interface:**



### Hardware Interfaces

Since the application must run over the internet, all the hardware required to connect to the internet will be the hardware interfaces for the system. As for e.g. Modem, WAN – LAN, Ethernet Cross-Cable.

We have chosen the IIT Kanpur servers to host our WebApp Service.

### Software Interfaces

**Operating System:**

As it is a web-based service, the operating system should not affect the performance of the system. The operating system should have a working browser and internet connection.

**Database Service:**

To save the information of students, we have planned to use PostgreSQL

**Framework:**

We will be using the Django framework because it is easy to implement. We are using Javascript, HTML, and CSS for the front-end. For Backend, we are using PostgreSQL and Django.

**Browser:**

A web browser that supports Django, HTML, and CGI is perfect for running this app.

## Functional Requirements

**3.2.1 F1: Configuring User**

1. This allows the user to log in to his account on the device using his CC ID and password
2. If the user signs up for the first time, then other details like password generation and its confirmation, name, phone number, etc will be required.
3. Similarly, the user can sign out

### This feature will allow for portability on any device

### 3.2.2 F2: QR Scanner.

1. The student scans a QR code.
2. (a) If the QR Code is for Hall Entry/Exit, it leads to F2/F3.

~~(b) If the QR Code is for Mess Extras, it leads to F6.~~

### 3.2.3 F3: Hall Entry.

The student enters the roll number, room number, and name of the person in the hall who he/she is going to meet.

The student’s data gets added to the database of outside-hall students currently in the hall.

### 3.2.4 F4: Hall Exit.

The student’s data gets removed from the database of outside-hall students currently in the hall.

### 

### 3.2.5 F5: Notification.

~~At 23:50 hrs, a notification is sent from email and SMS to all the outside-hall and guests students to vacate the hall.~~

The security has the ability to call the registered mobile number of the student from the security portal.

**3.2.6 F6: Query of Students inside the Hall.**

1. The security guard asks for a query of the list of students inside the hall.
2. The database gets searched, returning the list of students.
3. ~~The security guard can search for a particular name in the given list, and get his/her details.~~
4. Queries can also be run for editing the details and deleting in case of accidental wrong entry.

**3.2.7 F7: Ordering Extras.**

1. ~~After scanning the QR code in the mess, the student will have an option of selecting the item(s) from the menu specified by the mess manager from F8.~~

The student can select the extra(s) from the menu specified by the mess manager from F8.

1. The Student will be given a drop-down list where he selects his/her choice and is succeeded by a post-confirmation message.

**3.2.8 F8: Order History.**

1. The student can request their order history.
2. ~~The system will return all the orders in their name in the past month.~~

The system returns their past 6 orders. The total amount they have spent is always stored.

**3.2.9 F9: Menu Updation.**

1. The mess manager will have access to edit the daily menu database.
2. They can remove and add new items along with their prices.

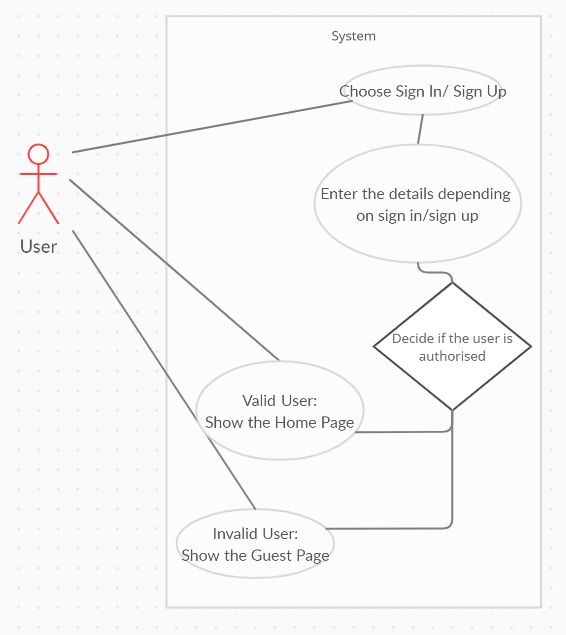
**~~3.2.10 F10: Entries Updation.~~**

1. ~~As a safety feature, we are providing the facility of manually editing the database of entry and exit. However, the edits will be shown on the original database.~~
2. ~~Only the authorized person (security guards) with edit access will be able to edit the entries.~~
3. ~~Up to 2 edits are allowed and all the versions are stored in the database.~~

## 

## Use Case Model

### 3.3.1 U1: User Configuration



**Author –** Shashwat Gupta

**Purpose** - Demonstrate how the user will sign in/sign up

**Requirements Traceability –** F10

**Priority** - Medium

**Preconditions** - The user puts his ID and details to sign in~~/sign up~~

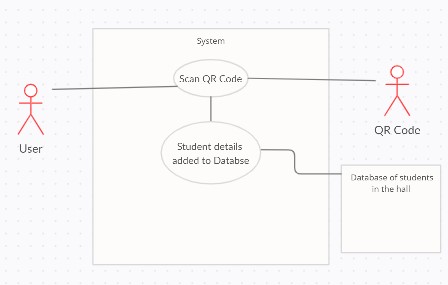
**Postconditions** - The main window of the WebApp appears.

**Actors** – Any user, database

**Exceptions** -

**Notes/Issues** -

### 3.3.2 U2: Entering a hall.



**Author –** Aryan Vora

**Purpose** - Demonstrate how a student will gain entry to a hall.

**Requirements Traceability –** F2, F3

**Priority** - Medium

**Preconditions** - A student scans the QR code using the web-app.

**Postconditions** - The database gets updated with details of the student who scanned the QR code.

**Actors** – Student, QR Code, Database

**Exceptions** -

**Notes/Issues** -

### 

### 

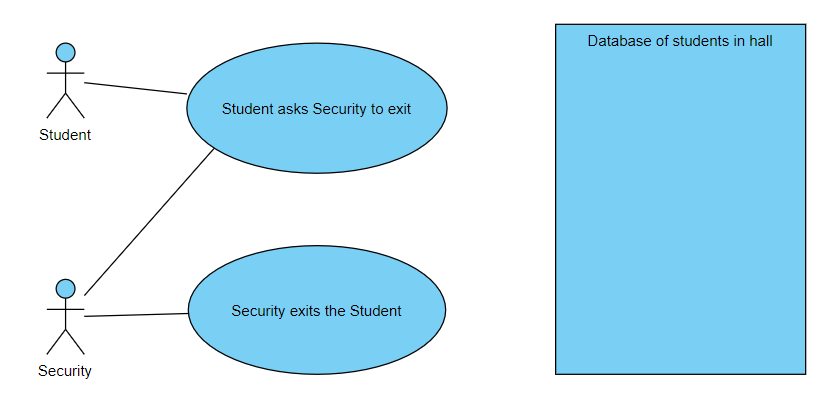
### 

### 

### 

### 

### 3.3.3 U3: Exiting a hall.



**Author –** Aryan Vora

**Purpose** - Demonstrate how a student will exit a hall.

**Requirements Traceability –** F2, F4

**Priority** - Medium

**Preconditions** - ~~A student scans the QR code using the web-app.~~ The student has entered the hall.

**Postconditions** - The database gets updated and removes the details of the student ~~who scanned the QR code~~ chosen by the security.

**Actors** – Student, ~~QR Code,~~ Database, Guards

**Exceptions** -

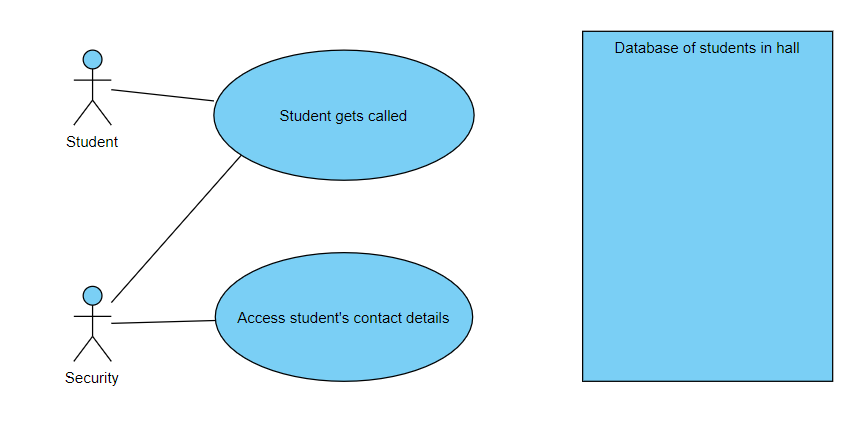
**Notes/Issues** -

### 

### 

### 

### 3.3.4 U4: Notification.

~~~~

**Author –** Aryan Vora

**Purpose** - Demonstrate how students get notified.

**Requirements Traceability –** F5, F6

**Priority** - High

**~~Preconditions~~** ~~- The time is 2350hrs.~~

**~~Postconditions~~** ~~- At 2350hrs, the system will generate a query to get the students outside the hall and send them an auto-generated email and text message that notifies them to leave the hall. At 0000hrs, the system again does the same. Also, at 0000hrs the system provides the details of students still inside the hall to the security for further action.~~

##### **Preconditions** - The student is in the hall.

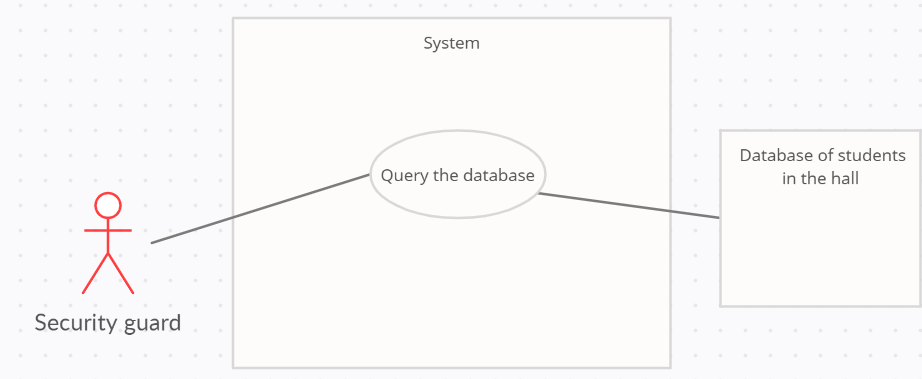
##### **Postconditions** - Security can access the phone numbers of all the students who are currently in the hall. The security can call the student in one click.

**Actors** – Database, Students, Guards

**Exceptions** -

**Notes/Issues** -

### 3.3.5 U5: Query of Students inside a hall.



**Author –** Ankur Kumar/Aryan Vora

**Purpose** - Provision of data of students inside the hall.

**Requirements Traceability –** F6

**Priority** -High

**Preconditions** -The user needs to be of the Guard profile

**Postconditions** - None

**Actors** – Security Guards.

**Exceptions** - Empty Record

**Notes/Issues** -

### 

### 

### 

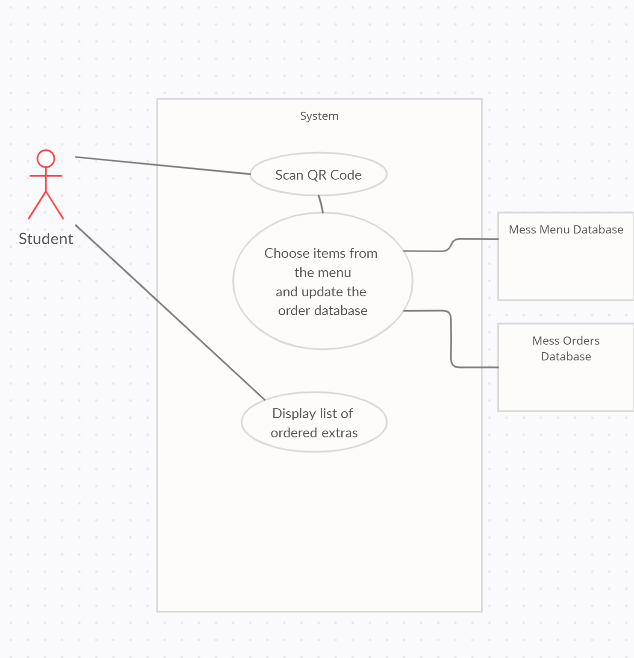
### 

### 

### 

### 

### 3.3.6 U6: Ordering Mess Extras



**Author –** Ankur Kumar/Aryan Vora

**Purpose** - Register the extras in the database

**Requirements Traceability –** F2, F7

**Priority** - Low

**Preconditions** - ~~QR code scanning~~

**Post \conditions** - The details of the transaction will be added to the database

**Actors** – Students

**Exceptions** -

**Notes/Issues** -

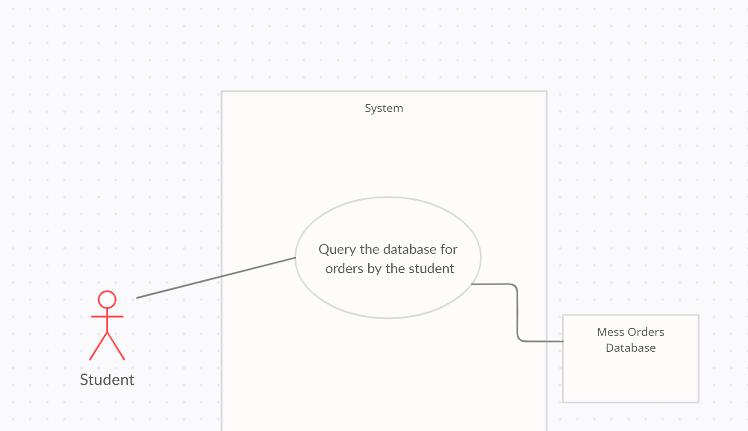
### 

### 

### 

### 

### 3.3.7 U7: Order History



**Author –** Ankur Kumar/Aryan Vora

**Purpose** - To enumerate all the extras bought by the particular student.

**Requirements Traceability –** F8

**Priority** - Low

**Preconditions** - None

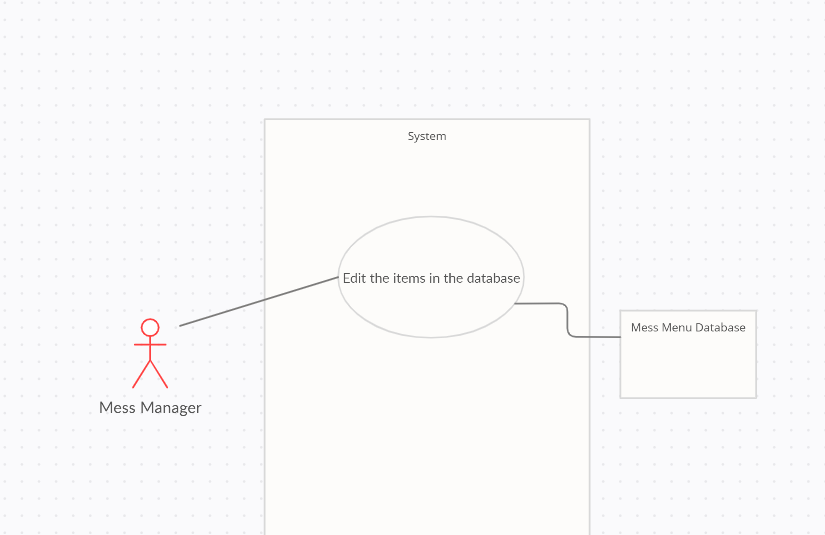
**Postconditions** - The user shall be provided with the required info

**Actors** – Student

**Exceptions** -

**Notes/Issues** -

**3.3.8 U8: Menu Updation**



**Author –** Ankur Kumar/Aryan Vora

**Purpose** - Enables the mess manager to update the daily mess menu database

**Requirements Traceability –** F9

**Priority** - Medium

**Preconditions** - The user must be a Mess Manager.

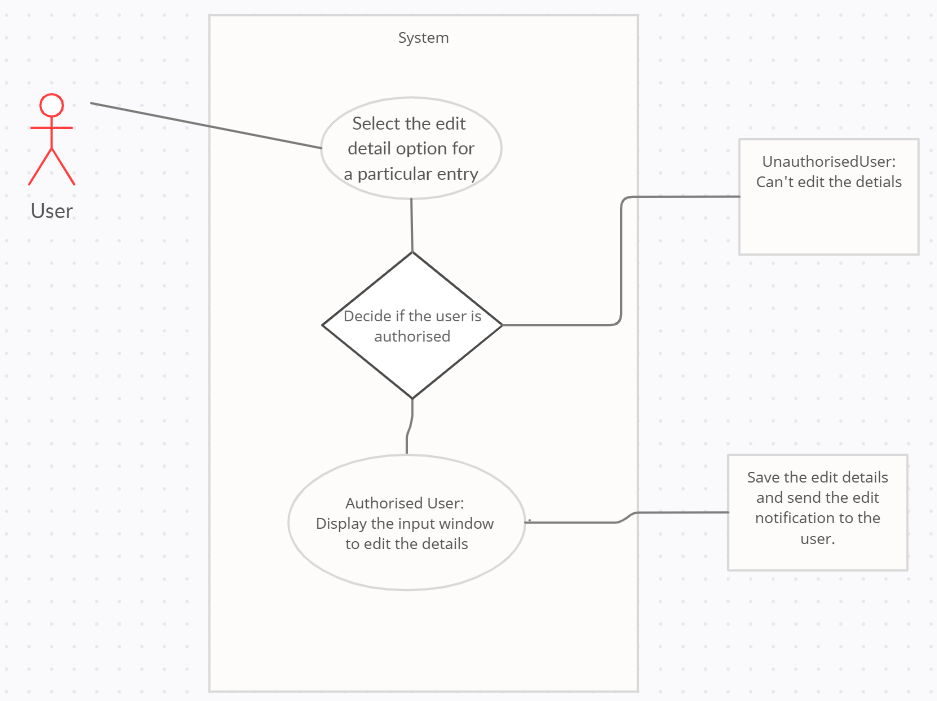
**Postconditions** - The mess menu database updated.

**Actors** – Mess Manager

**Exceptions** -

**Notes/Issues** -

**~~3.3.10 U10: Editing Hall Entry/Exit Details~~**

~~~~

**~~Author –~~** ~~Shashwat Gupta~~

**~~Purpose~~** ~~- Demonstrate how the authorized person will be able to edit the details.~~

**~~Requirements Traceability –~~** ~~F10~~

**~~Priority~~** ~~- Medium~~

**~~Preconditions~~** ~~- The authorized person scans the QR code using the web-app.~~

**~~Postconditions~~** ~~- The window appears which enables the update of the entry-exit information.~~

**~~Actors~~** ~~– Authorised Person, QR Code, Database~~

**~~Exceptions~~** ~~- The Authorised person is not able to scan the code either due to an incorrect device or due to the error in QR generation.~~

**~~Notes/Issues~~** ~~- How to deal with the exception mentioned above~~

# Other Non-functional Requirements

## Performance Requirements

* The DigiCampus application should load and be usable within 5 seconds to ensure no lag on the user side.
* The DigiCampus application should update the interface on interaction within 5 seconds to ensure no lag on the user side.
* The DigiCampus application should update the used databases within 5 seconds to ensure consistent operation.
* The database of the DigiCampus application should be normalized to prevent redundant data and improve performance.
* The database should be distributed to prevent outages.
* At any particular time, all IITK students should be able to use the application simultaneously without any overload to prevent crashing.
* A stable and functional application on Android, iOS and Windows devices.
* The application is hosted on a web server that can be accessed concurrently
* The storage of the database should be such that it can handle the data of all students of IITK and be updated for more students if required.

## Safety and Security Requirements

* ~~Databases should use sharding to be redundant to prevent the loss of data.~~
* ~~Backups of the database should be done hourly and be kept for a week.~~
* Authentication will be required by the user to access their personal data.
* Access to the master database will be restricted to the institute/hall administration.
* Provision of blocking access to any user’s data using their ID card if misplaced will be provided.
* I~~nitial login will be authenticated using CC ID login credentials.~~ For now, since we don’t have access to pingala server, we are not using CC ID. The Admin can create IDs as per requirements
* Any keys used for the DigiCampus programming interface should be stored securely.
* Only the DigiCampus programming interface should be able to connect to the databases.
* ~~Databases should be behind a firewall and be encrypted.~~

## Software Quality Attributes

**4.3.1 Availability:**

This application will be made available to all IITK students owning an ID card and CC details.

**4.3.2 Correctness:**

This application will never allow anyone to view any person’s data except their own. The users will be able to log in to the DigiCampus application and view only their own data and records.

**4.3.3 Maintainability:**

The application should use continuous integration so that features and bug fixes can be deployed quickly without downtime.

**4.3.4 Usability:**

The interface will be kept simple and user-friendly and allow the authorised user to view their records and use its features with ease.

**4.3.5 Interoperability:**

The systems will interact with a database that will in turn interact with other systems.

**4.3.6 Reliability:**

Based on the limited number of users as well as limited database size, we can ensure the reliability and stability of the application through sufficient failsafe implementations.

# Other Requirements

* Database of Students.
* Authorized permission from IIT-Kanpur is required before accessing any kind of data of students.
* The security guards must be provided with either smartphones or Computers for this software as many of the security guards may not have a smartphone.
* Since the software will be used very frequently(on a daily basis) it is important that the client maintains it and fixes any bugs if they arise.
* Aimed to be integrated with other features to make a one-stop app for all campus needs.

**Appendix A – Data Dictionary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **Field** | **Datatype** | **Description** | **Operation** |
| **HALL\_DATA** |  |  | Database corresponding to Hall Entry/Exit status. |  |
| HALL\_DATA | Hall Number | Text | Hall Number | F2,F3,F4,F5,F6, F10 |
| HALL\_DATA | Student Name | Text | The name of the student entering the hall. | F3,F4,F5,F6,F10 |
| HALL\_DATA | Student Roll Number | Integer | The roll number of the student entering the hall. | F3,F4,F5,F6,F10 |
| HALL\_DATA | Student Phone Number | Integer | The phone number of the student entering the hall. | F3,F4,F5,F6,F10 |
| HALL\_DATA | To Meet Name | Text | The name of the student whom the outside student is visiting. | F3,F4,F5,F6,F10 |
| HALL\_DATA | To Meet Roll Number | Integer | The roll number of the student whom the outside student is visiting. | F3,F4,F5,F6,F10 |
| HALL\_DATA | Room Number | Text | The room number of the student whom the outside student is visiting. | F3,F4,F5,F6,F10 |
| HALL\_DATA | In-Time | Time | The time the student enters the hall. | F3,F4,F5,F6,F10 |
| HALL\_DATA | Out-Time | Time | The time the student exits the hall. | F3,F4,F5,F6,F10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **Field** | **Datatype** | **Description** | **Operations** |
| **MESS\_MENU** |  |  | Database corresponding to the mess extras menu. |  |
| MESS\_MENU | Item Name | Text | The name of the extras that are available for the meal. | F7,F9 |
| MESS\_MENU | Item Price | Integer | The price of the extras that are available for the meal. | F7,F9 |
| MESS\_MENU | Hall Number | Integer | Hall Number of the Mess | F7,F9 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **Field** | **Datatype** | **Description** | **Operations** |
| **MESS\_ORDERS** |  |  | Database corresponding to the mess extras orders. |  |
| MESS\_ORDERS | Item Name | Text | The name of the extras that were ordered. | F7,F8,F9 |
| MESS\_ORDERS | Item Price | Integer | The price of the extras that were ordered. | F7,F8,F9 |
| MESS\_ORDERS | Item Quantity | Integer | The quantity of the extras that were ordered. | F7,F8,F9 |
| MESS\_ORDERS | Total | Integer | Total price that the student has to pay for that transaction. | F7,F8,F9 |
| MESS\_ORDERS | Ordered Date | Date | The date on which the extra was ordered. | F7,F8,F9 |
| MESS\_ORDERS | Roll Number | Integer | The roll number of the student who ordered the extras. | F7,F8,F9 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **Field** | **Datatype** | **Description** | **Operation** |
| **STUDENT\_ DATA** |  |  | Database corresponding to student data. |  |
| **~~STUDENT\_DATA~~** | ~~Student Full Name~~ | ~~Text~~ | ~~Full name of the Student~~ | ~~F1,F3,F4,F5, F6,F7,F8,F9,F10~~ |
| STUDENT\_ DATA | Student First Name | Text | The first name of the student. | F1,F3,F4,F5, F6,F7,F8,F9,F10 |
| STUDENT\_DATA | Student Last Name | Text | The last name of the student | F1,F3,F4,F5, F6,F7,F8,F9,F10 |
| STUDENT\_ DATA | Student Roll Number | Integer | The roll number of the student. | F1,F3,F4,F5, F6,F7,F8,F9,F10 |
| STUDENT\_DATA | Student Email ID | Email | The email id of the student | F1,F3,F4,F5, F6,F7,F8,F9,F10 |
| STUDENT\_ DATA | Student Phone Number | Integer | The phone number of the student. | F1,F3,F4,F5, F6,F7,F8,F9,F10 |
| STUDENT\_DATA | Student Room Number | Text | The Room Number of Student | F1,F3,F4,F5, F6,F7,F8,F9,F10 |
| STUDENT\_DATA | Student Current Expense | Integer | Expense of the current month of the student | F7,F8 |
| STUDENT\_DATA | Student Expense Last Month | Integer | Expense of the previous month of the student | F7,F8 |
| STUDENT\_DATA | Student Total Expense | Integer | Total Expense of the student since the web-app used. | F7,F8 |
| STUDENT\_DATA | Student Expense Day | Integer | Day-Wise Expense of the student | F7,F8 |
| STUDENT\_DATA | Student Order\_ID | Integer | Order-ID of the order made by the student | F7,F8 |
| STUDENT\_ DATA | CC ID | Text | The CC ID of the student. | F1,F3,F4,F5, F6,F7,F8,F9,F10 |

**Appendix B - Group Log**

<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist in determining the effort put forth to produce this document>

* 22nd January- Aryan completed 1.1 and 1.2.
* 24th January- Group meeting with TA in charge to address queries.
* 24th January- Group meeting to divide work and discuss the way forward.

MoM (24th January):

1. Section 1.3,1.4,1.5 - To be done once the SRS is over
2. Section 2-> Samarth, Sarthak, Abhishek, Ananya
3. Section 3->Ankur,Shashwat,Aryan
4. Section 4->Dishay,Aayush,Girik
5. A weekly meeting of the group will be held every Monday.
6. Will complete first draft by 27th January

* 26th January- Ankur, Aryan, and Shashwat met to discuss developments in section 3.
* Samarth, Sarthak met to complete Section 2. Dishay, Girik, Aayush completed Section 4.
* 27th January- Use Case Models were completed by Aryan, Sarthak, and Shashwat which led to a better understanding of how the software will actually function and which helped to think about the minute details of what each step will look like.
* 29th January- Other requirements completed by Sarthak.
* 30th January- Sarthak, and Aryan worked on Data Dictionary.
* 30th January- Aryan finished 1.3, 1.4, and 1.5.
* 30th January- Proof Reading of version 1.0 by the entire team.
* 31st January-Ananya,Abhishek,Ankur made first version of UI on Figma
* 1st February- Completed Version 1.0 of SRS.
* 23rd April- Aryan and Shashwat updated Version 2.0 of SRS.