



**NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI – 620 015**

CSMI15

Software Engineering

Project on

**MULTIPURPOSE MESS APPLICATION FOR
STUDENTS AND CATERERS**

Done by:

102116007 Anirudh Krishnan

103116023 Hari Prasanth S

110116004 Antony Terence

110116007 Archana Ganesh

110116086 Sriram V

TABLE OF CONTENTS

ABSTRACT:	4
1. Introduction	5
1.1 Features:	5
1.2 Database Schema	6
1.3 Technology stack used	9
2. Software Development Model used	11
3. Requirements Gathering	13
3.1 Requirements elicitation	13
3.1.1 Interviews :	13
3.1.2 Brainstorming :	13
3.1.3 Facilitated Application Specification Technique :	13
3.1.4 Quality Function Deployment :	13
3.2 Requirements specification	14
3.3 Requirements verification and validation	14
3.4 Requirements management	14
4. Project Analysis	14
4.1 Study of the existing system	14
4.2 Need of the proposed system	15
4.3 Scope of the system	15
4.4 Feasibility	15
5. Software Design Model	16
6. Implementation	16
6.1 UI components	17
6.2 Backend implementation	21

7. Testing strategies	25
7.1 Unit testing	25
7.2 Integration testing	25
7.3 Functional testing	25
8. Project Management	26
9. Application demonstration	28
9.1 Student screen	28
9.2 Caterer screen	34
10. Conclusion	40
11. References	40

ABSTRACT:

A multipurpose mess application was created to avoid food wastage. Students get a QR code which is verified by the caterers so that they can't eat from a mess they are not registered to. The amount of food prepared and consumed are analysed to make predictions to avoid wastage. React native was used to create a multipurpose mess application with an intuitive User Interface. Django was used to frame API endpoints and to design database schema for the application. Holt-Winters forecasting was used for food wastage prediction.

1. Introduction

1.1 Features:

Proposed Functionality for Caterers:

- Food Consumption Stats (History)
- Food Consumption Update
- Quantity Prediction (Less food wastage)
- QR Check-in
- Update students with menu changes etc.
- Complaints Dashboard

Proposed Functionality for Students:

- QR Check-in
- Dashboard
 - Next Meal Menu
 - Updates
- Mess Menu
- Meal skips
- Complaints and Compliments

1.2 Database Schema

The normalization techniques were taken into consideration and the tables have been created by implementing those techniques.

1. Messes table

Messes	
messID	int
messName	varchar
Password	varchar

This table is created for storing the registered messes' information. Here messId is the primary key.

2. Student table

Student	
studentID	int
Name	varchar
rollNo	int
messID	int
pref	varchar
password	varchar

This table is created for storing the information of the students who are registering in the portal. Here studentID is the primary key.

3. Menu table

Menu	
menuID	int
messID	int
day	varchar
nameOfFood	varchar
mealType	varchar

The above table is created for storing the menu information corresponding to the particular mess. Here menuID is the primary key.

4. Visited table

Visited	
messID	int
date	varchar
mealType	varchar
studentID	int

The above table is created for storing the details of the students who have visited that mess on a particular date, along with the type of meal they had. Here messID is the primary key.

5. Reviews table

Visited	
messID	int
review	varchar

The Reviews table is created to store the reviews given by the students for that particular mess identified by the messID.

6. foodStats table

foodStats	
preparedQ	int
consumedQ	int
leftoverQ	int
date	varchar
menuID	int

The foodStats table is created to store the information about the consumption, preparation quantity and leftover quantities on a particular date.

1.3 Technology stack used



React Native:

React Native is an open-source mobile application framework created by Facebook. It was used to create a native application that'd work in Android, iOS and Web. It was chosen because of its ability to share across different platforms.



JavaScript:

JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. It was used to make react native components.



Node

Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a web browser. It uses the chrome's v8 engine. It was used to bundle the JS code into native app.



npm

npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It is used to include open source libraries and for storing project dependencies.



Expo

Expo is a toolchain built around React Native to help you quickly start an app. It provides a set of tools that simplify the development and testing of React Native apps and arms you with the components of user interface and services that are usually available in third-party native React Native components.



SQLite

SQLite is a relational database management system contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. It was used to store all the data.



Python

Python is an interpreted, high-level, general-purpose programming language. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. It was used in the server side.



Django

Django is a Python-based free and open-source web framework that follows the model-template-view architectural pattern. It is an MVC framework in python and was used for server side development.



Git

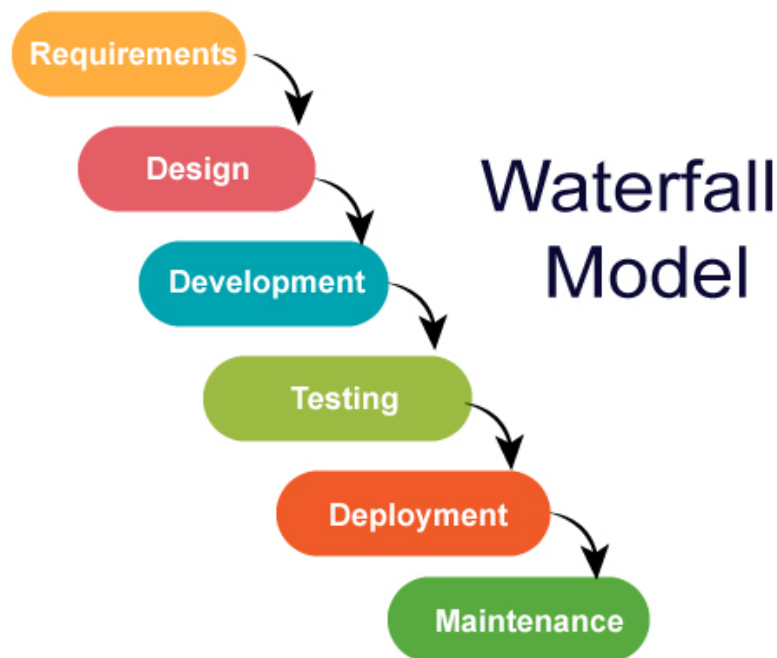
Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. It was used to build the project at the same time by multiple developers working on different features.

2. Software Development Model used

The Software Development Life Cycle (SDLC) is a framework that describes the activities performed at each stage of a software development project. The SDLC process is used by the software industry to design, develop and test high quality software. It aims to produce the quality software that meets or exceeds customer expectations, reaches completion within time and budget.

Waterfall model was chosen for the SDLC of the project because of the following nature of the application:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.



- Requirement Gathering and analysis – All possible requirements of the system to be developed were captured in this phase using surveys and interviews.
- System Design – The requirement specifications from the first phase were studied in this phase and the system design was prepared. UI and DB schema was agreed on. This system design helped in specifying hardware and system requirements and in defining the overall system architecture.
- Implementation – With inputs from the system design, the system was first developed in small programs called units, which were integrated in the next phase. Each unit was developed and tested for its functionality, which is referred to as Unit Testing.
- Integration and Testing – All the units developed in the implementation phase were integrated into a system after testing of each unit. Post integration the entire system was tested for any faults and failures.
- Deployment of system – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market. The application is yet to be deployed.
- Maintenance – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment. This is currently not applicable to this application.

3. Requirements Gathering

3.1 Requirements elicitation

3.1.1 Interviews :

- Several students across a wide range of age groups and departments were surveyed. Their concerns were taken into consideration while developing this software.
- Mess representatives were contacted to craft an application that served the needs of the caterers as well.

3.1.2 Brainstorming :

- Discussed different ideas and methods to ensure that students don't enter other messes and to make the generated QR code as unique as possible.
- Discussed various algorithms for time series analysis for the prediction of the quantity of the food to be produced.

3.1.3 Facilitated Application Specification Technique :

- Different use cases and all the possible applications related to mess were considered and the easy accessibility and feasibility of the applications was also considered.

3.1.4 Quality Function Deployment :

- Normal requirements : Food consumption stats, Food consumption update, Menu updates
- Expected requirements : Qr check in, Mess menu, complaints and compliments, next meal menu.
- Additional requirements : Food quantity prediction based on available data.

3.2 Requirements specification

The data acquired using requirement elicitation process was used to list of the specifications required of the product:

- DB schemas were listed out and discussed to bring it into a normalized form
- UI design was agreed upon and listed out which includes :
Background design, theme of the software etc
- The project was divided into components and what is expected of each of them were agreed upon.

3.3 Requirements verification and validation

- The requirements generated were verified and validated within the team and also the intended audience.

3.4 Requirements management

- The requirement gathering process was analyzed, documented, tracked, prioritized to get smooth working experience and user approval for the developed application.

4. Project Analysis

4.1 Study of the existing system

The existing mess system involves manually checking if the student belongs to the particular mess. This is done with the help of mess cards. Also, as per the current system, the amount of food to be made is decided based on the number of students registered and doesn't take into consideration the holidays etc.

4.2 Need of the proposed system

The proposed system solves the problem of verification by making it automatic. The students get a QR code which is verified by the caterers. When the caterer scans the QR code, the student's roll number is verified with the list of registered students in the database. This is an efficient solution for the verification problem. Also to avoid food wastage, details about the prepared amount of food and the amount consumed are gathered. Holt-Winters forecasting is applied to obtain food wastage prediction. This application facilitates the menu to be stored in the database and changed if there are any menu changes from the mess side. So students can easily view the menu in the application. Feedback system has been included which helps students convey their compliments and grievances easily to the caterers.

4.3 Scope of the system

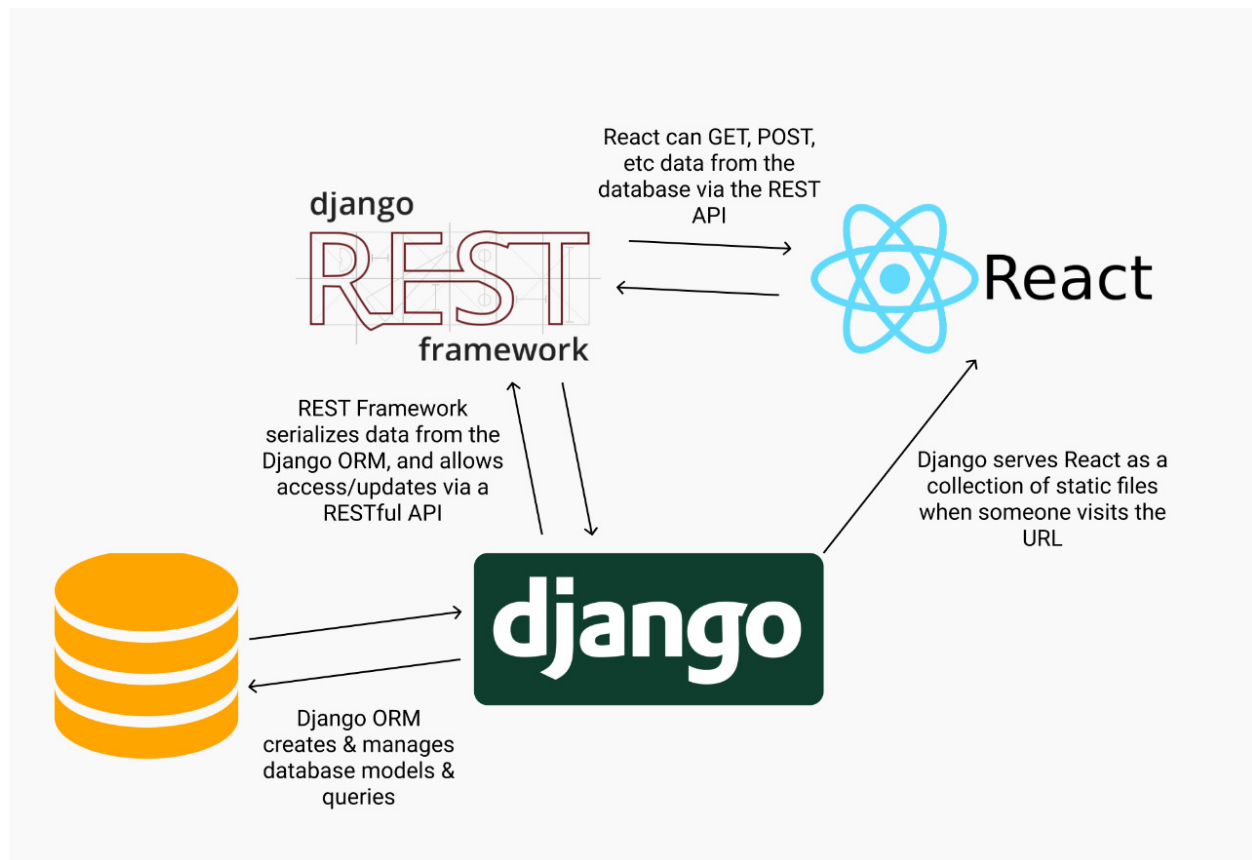
Project scope is the part of project planning that involves determining and documenting a list of specific project goals, deliverables, tasks, cost and deadlines. This proposed system can be used to create an integrated portal for students and caterers alike for facilitating communication between them and to provide an intuitive interface as a whole.

4.4 Feasibility

The currently existing system has details of students and registered messes stored. But they are not used efficiently. This system can use existing data to create an integrated system for students and caterers. The technology stack which was agreed upon by the team can be used efficiently to develop the system as per the requirements.

5. Software Design Model

The interaction of the various components used in our project is pictorially represented in the following diagram:



6. Implementation

The implementation has been divided into frontend and backend implementation which will be explained in detail in this section.

6.1 UI components

The frontend implementation was divided in to various components :

Student components:

Login page

State variables :

Username, password based on the value in the input field. The values in these state variables would be later used to send data to specific API endpoints.

Props variables :

type (as student) this is to determine whether the user is a student or not based on the tab in the form they are filling.

Navigation this is to navigate to a different component after a particular action has been made.

API endpoints :

The API endpoints used in this components are

[/login](#) : A post request to authenticate the student log in. The roll number and password is sent in the body.

Home page

State variables :

Name, roll number and mess which are first set to null then changed to respected values after fetching from local storage.

Props variables :

Navigation to navigate to different components on the click of the respective buttons.

API endpoints :

None

Dashboard

State variables :

Menu1, day1, meal_type1, menu2, day2, meal_type2 are nil at first and then are set the values based on the next 2 menus.

Props variables :

None

API endpoints :

[/user/menu](#) is used twice to get data for both the meals and the state variables menu1 and menu2 are changed. The body is the day, mess Id and the meal type.

Menu

State variables :

None

Props variables :

None

API endpoints :

[/user/allmenu](#) meal to get the menu for all days of the week and display the menu. The body of the request consists of mess Id.

QR check in

State variables :

Roll number and timestamp of student entering.

Props variables :

None

API endpoints :

None

Feedback

State variables :

feedbacktext contains the value inside the input field.

Props variables :

None

API endpoints :

[/user/review](#) input is a post request to send the feedback. The body contains messId and feedback text.

Caterer components:

Login page

State variables :

Username, password based on the value in the input field. The values in these state variables would be later used to send data to specific API endpoints.

Props variables :

type (as caterer) this is to determine whether the user is a caterer or not based on the tab in the form they are filling.

Navigation this is to navigate to a different component after a particular action has been made.

API endpoints :

The API endpoints used in this components are

[/login](#) : A post request to authenticate the caterer log in. The messId and password is sent to the body.

Home page

Just a functional component without any need for state or props variables.

There was no request made from this component either.

Feedback

State variables :

None

Props variables :

None

API endpoints :

[/caterer/complaints](#) this to get the list of all the complaints received. The body contains the messId.

QR scanner

State variables :

Rollnumber, messId, timeStamp the values of which are received after scanning the qr code in the students phone.

Props variables :

None

API endpoints :

[/caterer/messCheck](#) this is to see if a particular student belongs to this mess. The body contains the rollNumber and timestamp.

Analytics

State variables :

studentData is a list of all the student data for each day which is initially nil but is later set after an endpoint call.

Props variables :

Mode to depict the layout the caterer wants to see in. It can be either grid mode or list mode.

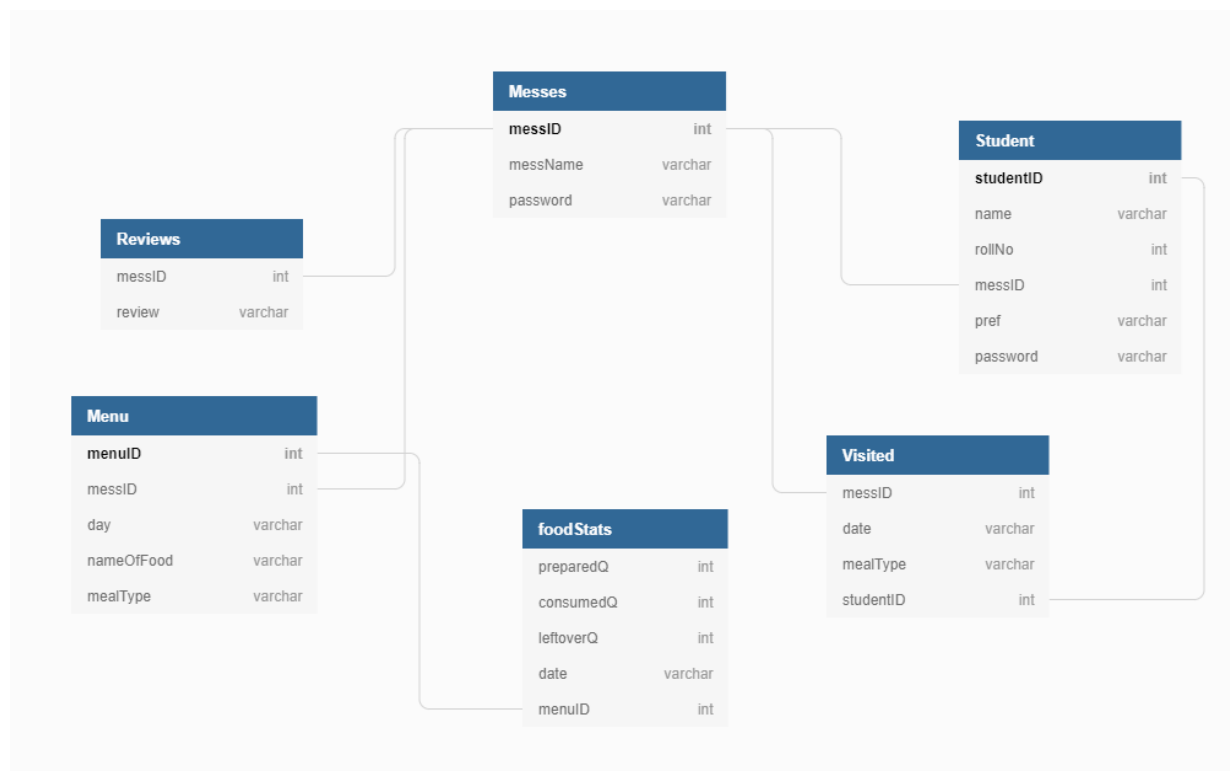
API endpoints :

[/caterer/getStudentData](#) is a post request used to fetch all the days data. The body of the request contains messID which is fetched from asyncstorage.

6.2 Backend implementation

Backend is generally referred to the Database schema design for the information storage, Application Program Interface (API) endpoints framing and the general website architecture design. In our project, for backend development we used Django, a high-level Python Web framework, since it is fast, secure and extremely scalable.

The database schema we designed is as follows:



As inferred from the above schema, we have created multiple tables interlinked with foreign keys, for efficient and optimal data access.

In django, the table structures are implemented using a concept called models.

1.) Mess model:

```
class Messes(models.Model):  
    messID = models.AutoField(primary_key=True, auto_created=True)  
    messName = models.CharField(max_length=25)  
    password = models.CharField(max_length=100)
```

Here the mess ID is taken as the primary key for identifying a particular record. It also consists of the name of the mess and the password they used to register. This mess ID is also used as foreign keys in other models.

2.) Student model:

```
class Student(models.Model):  
    studentID = models.AutoField(primary_key=True, auto_created=True)  
    name = models.CharField(max_length=30)  
    rollNo = models.IntegerField()  
    messID = models.ForeignKey(Messes, on_delete=models.CASCADE)  
    pref = models.CharField(max_length=10)  
    password = models.CharField(max_length=100)
```

Here the student Id is taken as the primary key for identifying a particular record. It also consists of the student information like name, roll number, preference and his password. In this model, the messID is used as a foreign key to access the records specific to that particular mess.

3.) Menu model:

```
class Menu(models.Model):  
    menuID = models.AutoField(primary_key=True, auto_created=True)  
    messID = models.ForeignKey(Messes, on_delete=models.CASCADE)  
    day = models.CharField(max_length=10)  
    nameOfFood = models.CharField(max_length=15)  
    mealType = models.CharField(max_length=15)
```

In this model, the menuID is taken as the primary key for identifying a particular record. It also consists of the particular day that menu belongs to, the names of the food items and the meal type. In this model, the messID is used as a foreign key to access the menu belonging to that particular mess.

4.) Food stats model:

```
class foodStats(models.Model):
    #foodID = models.AutoField(primary_key=True, auto_created=True)
    preparedQ = models.IntegerField()
    consumedQ = models.IntegerField()
    leftoverQ = models.IntegerField()
    date = models.CharField(max_length=25)
    menuID = models.ForeignKey(Menu, on_delete=models.CASCADE)
```

The foodStats model is used to store the information regarding the necessary statistics required for the caterers like Prepared quantity, consumed quantity and leftover quantity. In addition to these statistics, it also contains the date entry. In this model, the menuID is used as a foreign key to access the food statistics for that particular menu.

5.) Visited model:

```
class Visited(models.Model):
    messID = models.ForeignKey(Messes, on_delete=models.CASCADE)
    date = models.CharField(max_length = 25)
    mealType = models.CharField(max_length=15)
    studentID = models.ForeignKey(Student, on_delete=models.CASCADE)
```

In this model, both the messID and the studentID field is used as a combination of foreign keys to access the records of the students who visited that particular mess. This model also consists of the date visited and the meal type information.

6.) Review model:

```
class Reviews(models.Model):
    messID = models.ForeignKey(Messes, on_delete=models.CASCADE)
    review = models.CharField(max_length=100)
```

In the review model, the messID is used as the foreign key to access the reviews given for that particular mess.

Now, for the API endpoints, Django uses a functionality called views to satisfy the requirements. We have developed the following views for our application:

- **user/login** : This view is accessed when the user logs into our application.
- **caterer/login** : This view is accessed when the caterer logs into our application.
- **user/menu** : This view is accessed when the user wants to get the menu details for a particular date and from the particular mess.
- **user/allMenu** : This view is accessed when the student wants to view the entirety of the menu for a particular mess
- **caterer/studentAuth** : This view is accessed when the caterer wants to authenticate and retrieve the details of the students who visited their mess on a particular date.
- **user/review** : This view is accessed when the user wants to leave a complaint or a general feedback about a particular mess.
- **caterer/getStudentData**: This view is accessed when the caterers want to retrieve a particular students record who has registered for their mess.
- **caterer/foodConsumed**: This view is accessed when the caterers want to analyze the food consumption statistics for a particular date.
- **caterer/complaints**: This view is accessed by the caterers when they want to view the feedback given by the students.
- **caterer/getFoodData** : This view is accessed by the caterers to obtain the menu information for a particular date.
- **food_predict** : This view consists of the algorithm for obtaining the food consumption statistics.
- **time_series**: This view contains the Holt-Winters forecast algorithm for predicting the food wastage statistics.

7. Testing strategies

7.1 Unit testing

Also known as Program Testing, it is a type of testing where the analyst tests or focuses on each program or module independently. We have done a number of unit testing for different unit modules. We used jest and Enzyme for unit testing of the frontend application. For the backend unittest module built-in to the Python standard library was used for unit testing.

7.2 Integration testing

In Integration Testing, the analyst tests multiple modules working together. It is used to find discrepancies between the system and its original objective, current specifications, and systems documentation. The npm package React Testing Library was used for integration tests.

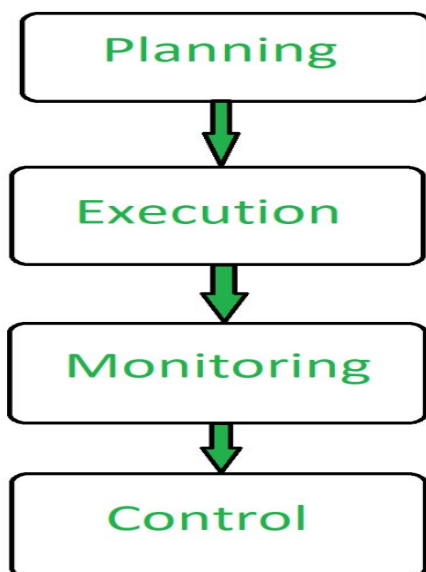
7.3 Functional testing

Function testing determines whether the system is functioning correctly according to its specifications and relevant standards documentation. Functional testing typically starts with the implementation of the system, which is very critical for the success of the system. For functional testing we wanted to use libraries that are platform agnostic and don't require too much knowledge of either platform so we went ahead with appium and webdriverio.

8. Project Management

Software Project Management (SPM) is a proper way of planning and leading software projects. It is a part of project management in which software projects are planned, implemented, monitored and controlled.

Aspects of project management:



Software project management consists of the following tasks:

Planning: This means putting together the blueprint for the entire project from ideation to fruition. It will define the scope, allocate necessary resources, propose the timeline, delineate the plan for execution, lay out a communication strategy, and indicate the steps necessary for testing and maintenance.

In our project, we initially laid out the blueprint for a first stage prototype with all the basic functionalities like user and caterer login, feedback portal, QR scanner for authentication, access menu and such. We decided to use github as a code collaboration platform for efficient communication.

Allocation of Roles: A software project management requires an efficient assembly of developers in various roles including Frontend developers, Backend Developers, analysts, testers.

Frontend Developers ensure the development of the application UI is implemented properly and all the required components are built and integrated. Backend developers take responsibility for the database architecture, API endpoints and general website architecture. Analytics team solves problems related to the statistical prediction of the data that we have integrated into our project. Testers ensure that the entire application is working perfectly and is bug free.

Execution: During the execution of the project, the progress is constantly monitored at each stage of the execution. Frequent team check-ins are also done for monitoring role wise progress.

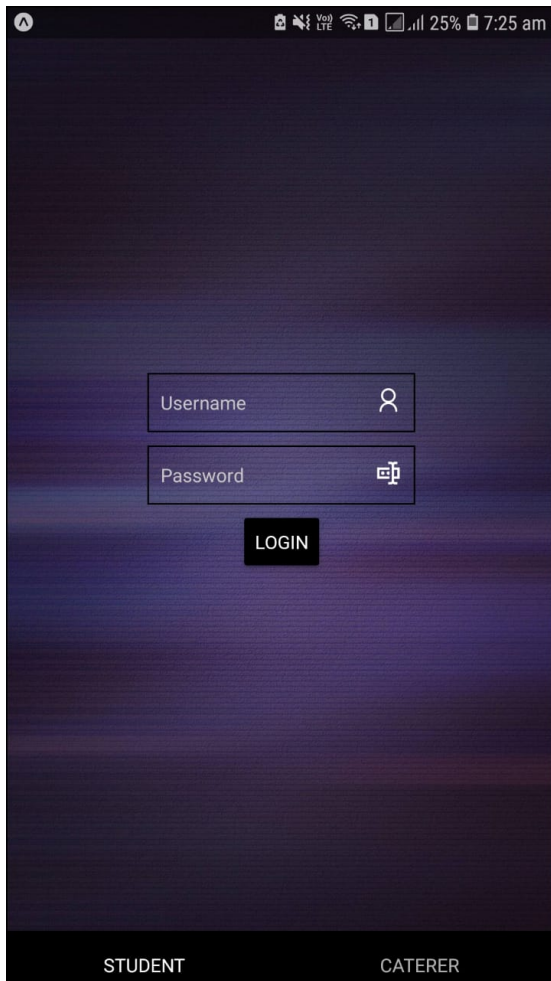
Time management: Staying on schedule is crucial to the successful completion of any project, but it's particularly challenging when it comes to managing software projects because changes to the original plan are almost certain to occur as the project evolves. We have executed the project using waterfall model as stated above, and the deadlines were properly met.

Maintenance: Software project management typically encourages constant product testing in order to discover and fix bugs early, adjust the end product to the customer's needs, and keep the project on target. Constant tests were done during the testing phase of the execution model and the project maintenance was done accordingly.

9. Application demonstration

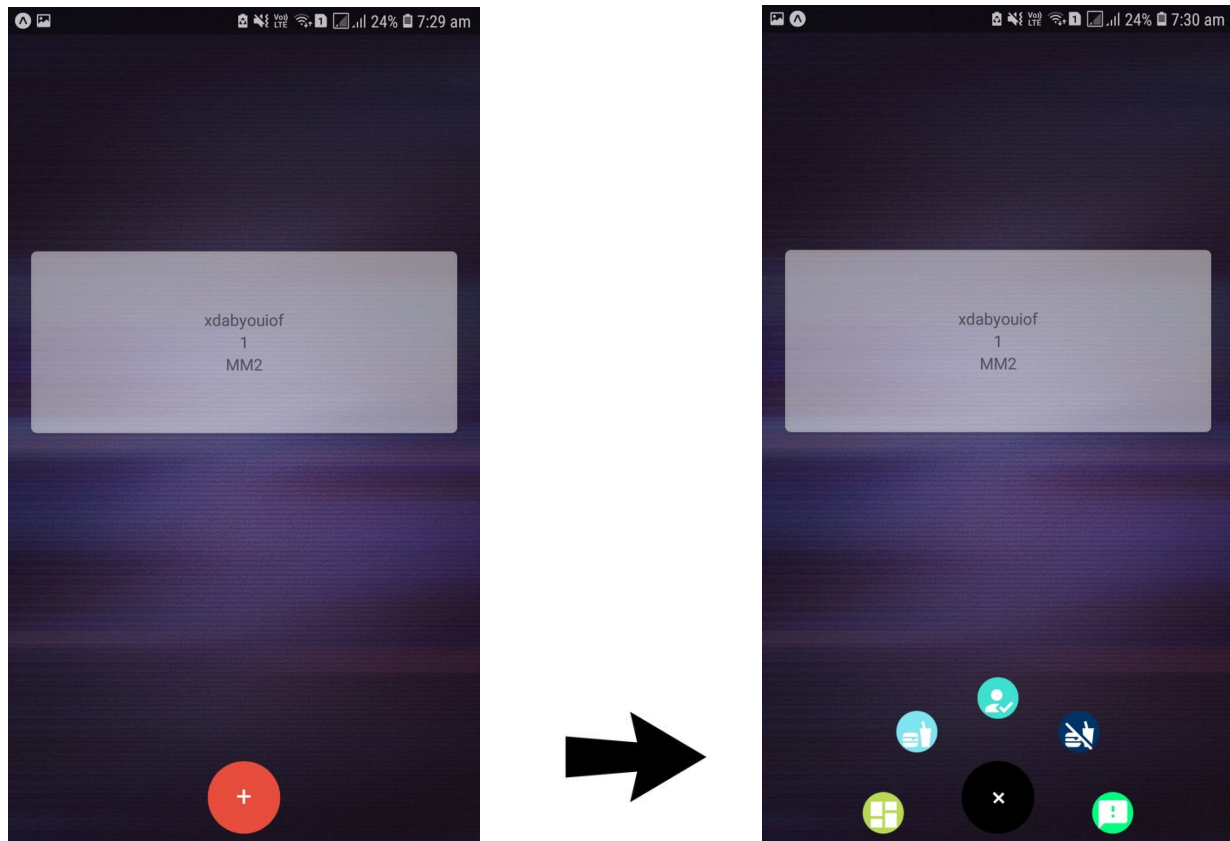
9.1 Student screen

Login page



The above picture shows the login screen for students. The username is the roll number and the password is created at the time of mess registration. The students can login to see their profile which consists of the mess their name, roll number and the mess they are registered to. The application has a dashboard and many other features for students.

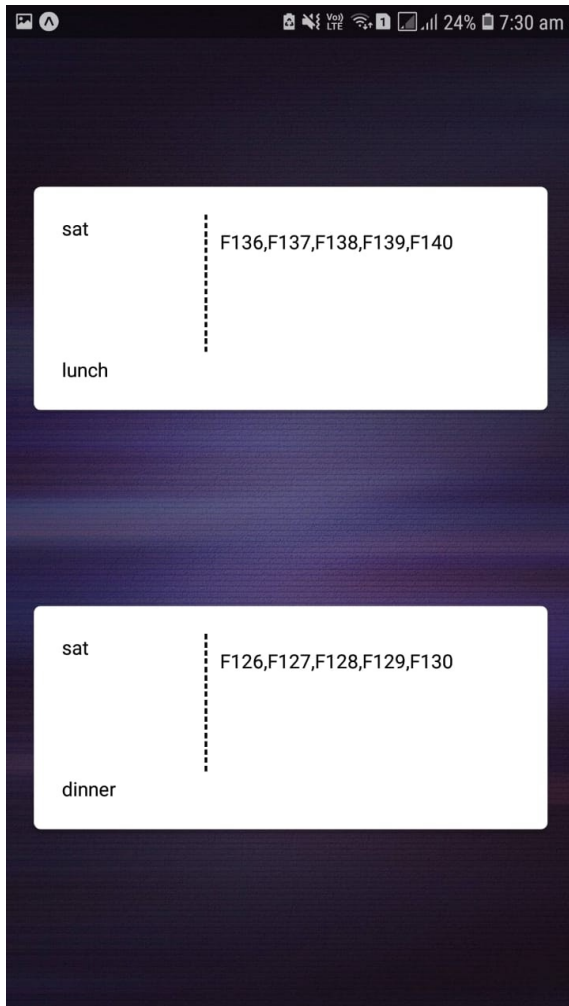
Home page



The first picture shows the screen that the students view after logging in. Profile section displays name, roll number and the registered mess. The red + icon at the bottom of the screen can be clicked to view and use the features.

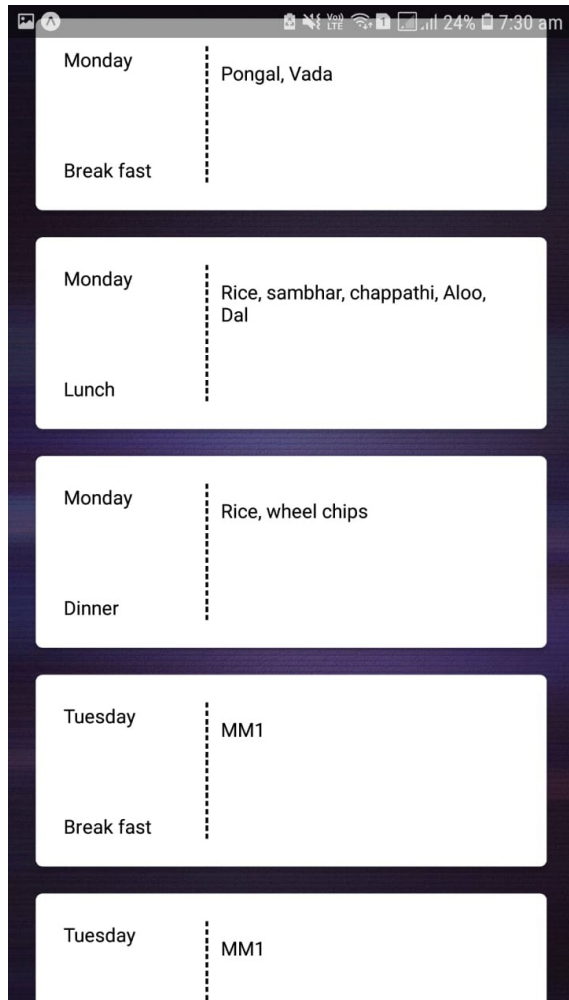
The second image shows the screen after the + icon is clicked. The features displayed are Dashboard, Menu , QR check in, Meal skip and Feedback.

Dashboard



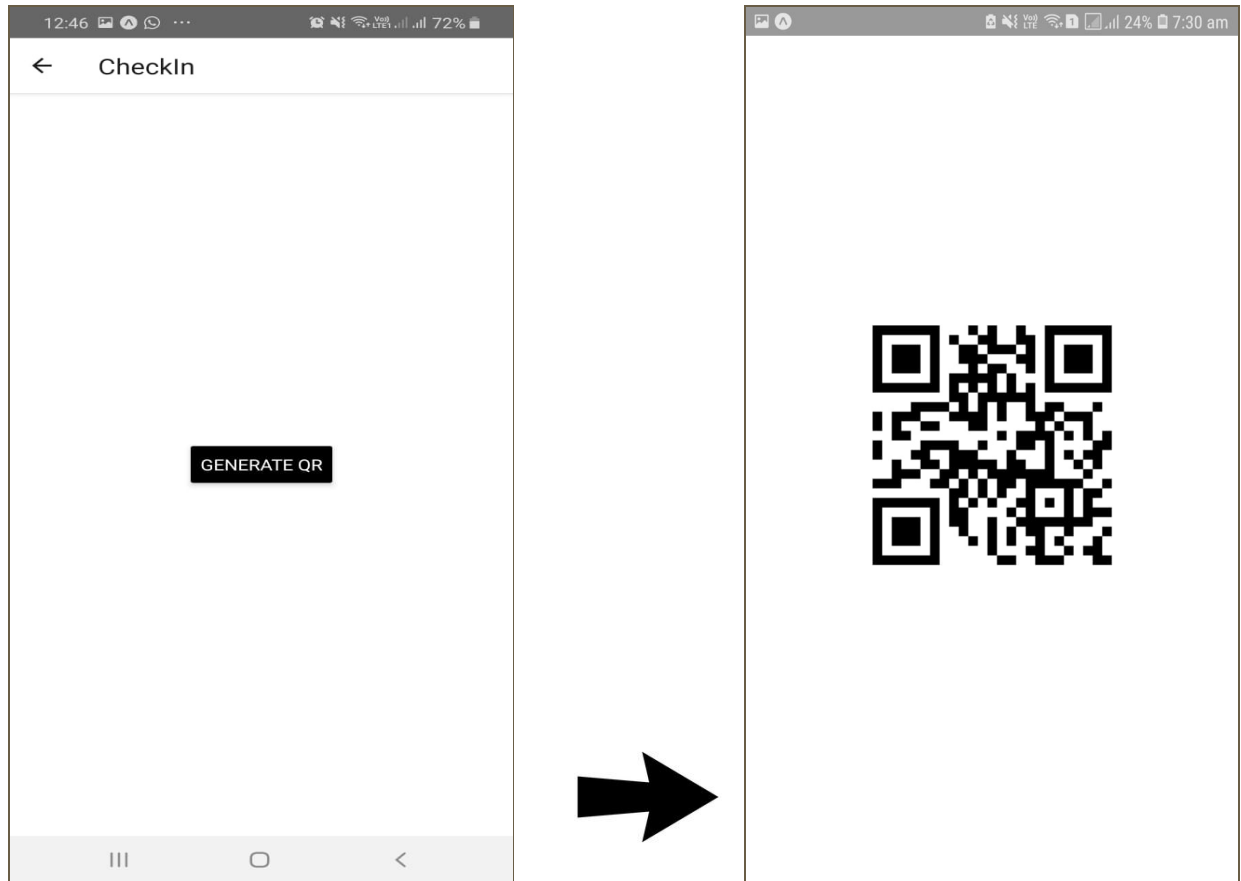
The dashboard feature displays the upcoming meals for that particular day (Mon-Sun) and that particular mess the user belongs to. Food details are shown on the right hand side of the card and the left hand side consists of the day and the meal type (Breakfast/ Lunch/ Dinner).

Menu



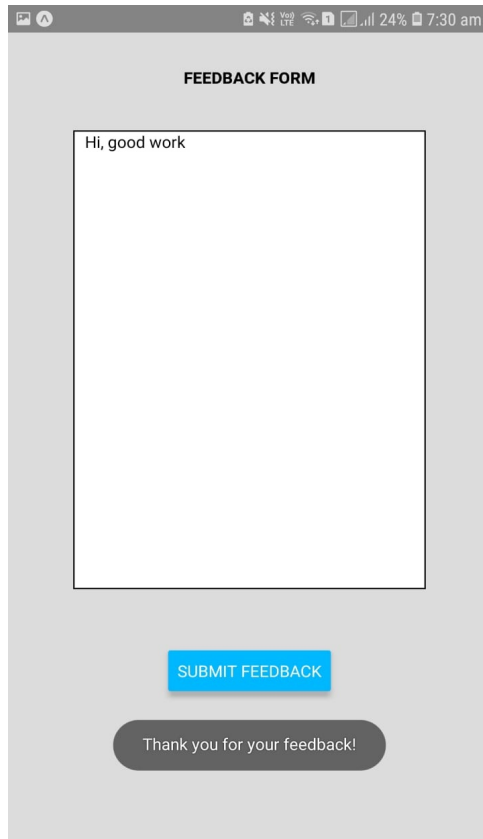
The menu screen as shown above displays the entire menu of the mess the student is registered to. They are displayed as cards. The left side of the card displays the day and the meal type (Breakfast/Lunch/Dinner) and the right side of the card shows the corresponding meal served.

QR check in



The above images show the QR check in feature for students. When the GENERATE QR button in the first screen is clicked, a QR code is generated which is shown in the second screen. To generate the QR, the student's roll number and the mess they are registered to are taken into consideration to generate an unique QR code. This ensures that the students do not eat in any other messes.

Feedback

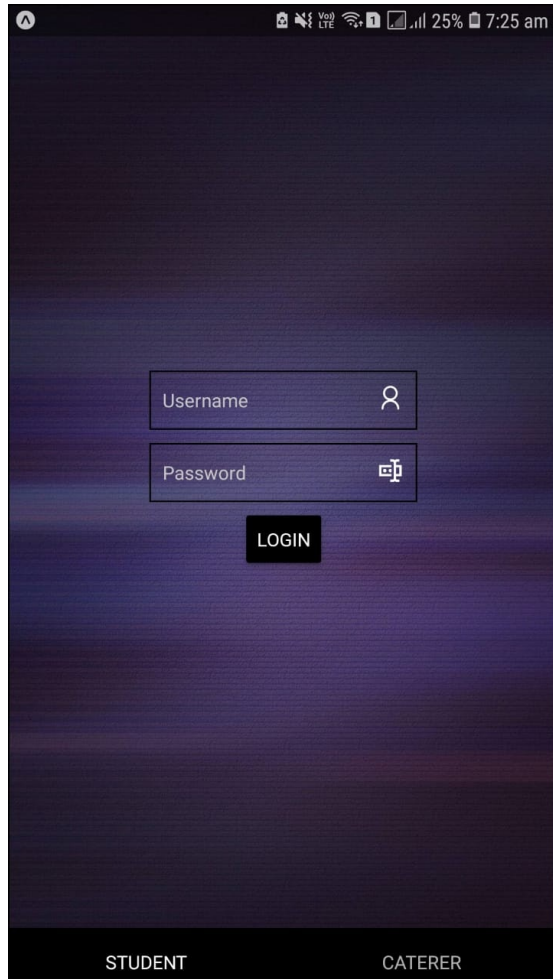


The screenshot shows a mobile application interface for a feedback form. At the top, the status bar displays various icons and the time 7:30 am. The app's title bar is labeled "FEEDBACK FORM". Below this, there is a large text input area containing the text "Hi, good work". At the bottom of the form, there is a blue button labeled "SUBMIT FEEDBACK". Below the button, a dark gray rounded rectangle contains the text "Thank you for your feedback!".

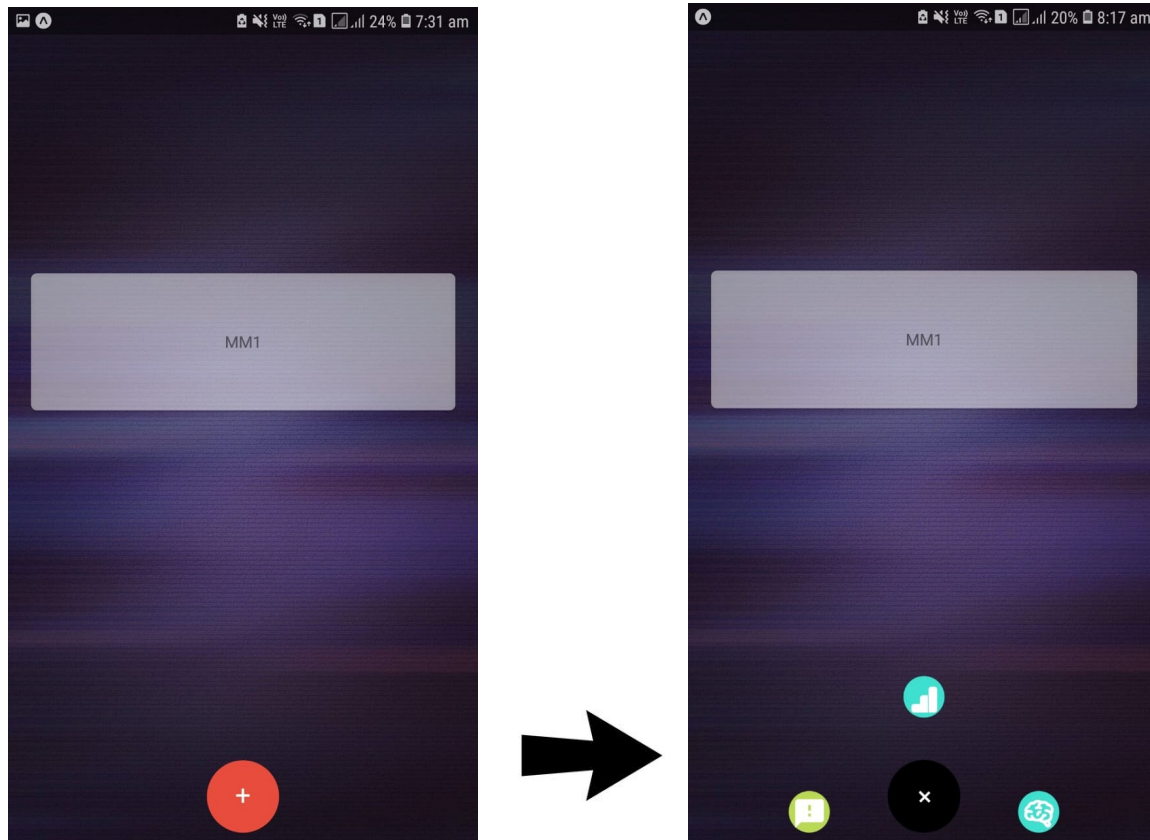
The students can leave feedback for the caterers using this feature. They can enter their thoughts and suggestions and press the submit feedback button. The app shows a confirmation message “Thank you for your feedback” to let the users know that their feedback has been submitted successfully.

9.2 Caterer screen

Login page



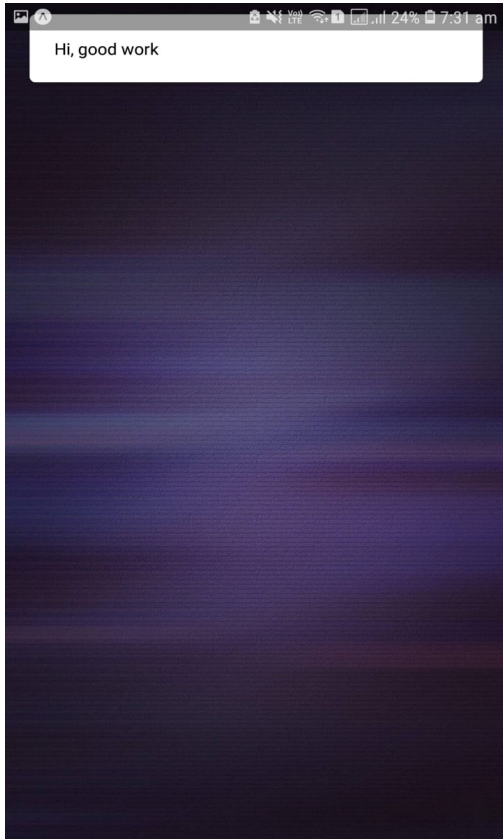
The above picture shows the login screen for caterers. The caterers can enter their details and login to see their mess details which is the name of the mess that they are registered as. The application has feedback, stats and other features for caterers.

Home page

The first picture shows the screen that the caterers view after logging in. Profile section displays the name of the mess. The red + icon at the bottom of the screen can be clicked to view and use the features.

The second image shows the screen after the + icon is clicked. The features displayed are Feedbacks, Stats and Analytics.

Feedback



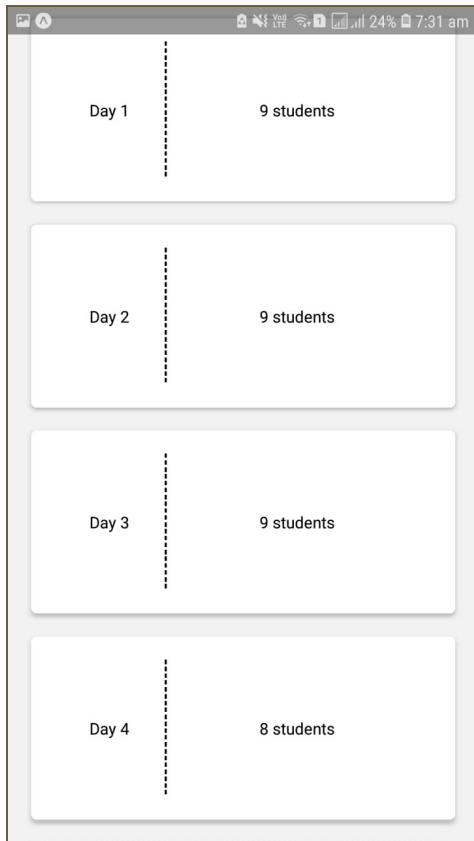
The screen shows the compliments and complaints sent as feedback to the caterers. The above picture displays the feedback which was sent in the student's feedback section. Likewise the caterers can view all the feedback received and work on it.

QR scanner



The above picture shows the QR scanner feature for caterers. From the QR code details like the roll number and the mess registered are retrieved and the caterers verify if the mess mentioned is the same as the one they belong to. This way students cannot enter messes that they are not registered to. This avoids food wastage in the mess they are registered to and food shortage in the mess that they are going to.

Stats



The screen shown in the above picture shows the stats for each day. They are displayed as cards. The left side of the card shows the day and the right side shows how many students visited. This can be used to analyse and see if there is any pattern which can help reduce food wastage.

Analytics



The above screen shows the amount of food consumed every day and for every meal. Caterers can compare these stats with the amount of food they prepared to avoid food wastage. Holt-Winters forecasting was used to predict the amount of food wasted for each day by taking into consideration data from over a period of time. This can be effectively used, with the help of caterers to prepare the right amount of food with minimal wastage and maximum consumption.

10. Conclusion

An application was conceptualized and developed, one that caters to both students and mess caterers alike. Adopting a waterfall model for the development process ensured that the requirements of all parties involved were taken into consideration while planning and developing the program. Features such as a comprehensive dashboard and quantity prediction are invaluable aids to students while mess caterers benefit from food consumption statistics in addition to predicting food quantity needs, cutting down on costs and wastage.

11. References

- Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave macmillan.
- Mall, R. (2018). Fundamentals of software engineering. PHI Learning Pvt. Ltd..
- <https://www.geeksforgeeks.org/>
- <https://reactnative.dev/docs/getting-started>
- <https://docs.djangoproject.com/en/3.0/>
- <https://www.sqlite.org/docs.html>