MINI PROJECT

ON

<u>HOSTEL MESS MANAGEMENT SYSTEM</u>

Bachelor of Technology
in

INFORMATION TECHNOLOGY

SEMESTER-VI

COCHIN UNIVERSITY OF
SCIENCE AND TECHNOLOGY

March 2021

SCHOOL OF ENGINEERING

DigiMess

Prepared by

1. S Atul Krishnan
2. Rahul R S
3. Sangeeth Raj
4. Ashwin Cherukat
5. Sarath S

Table of Contents

1. Introduction

1.1 Purpose

Digimess simplifies mess management for students and mess coordinators.

1.2 Need/Motivation

Need for better UI/UX, friendlier interface, functionalities and features compared to existing system while still maintaining ease of use. Since the current system is very time consuming, it was necessary to come up with an efficient way.

1.3 Intended Audience

This document is intended for UI/UX developer, UX writer, app developers. They are required to follow this document as a reference to develop the application throughout the project development.

1.4 Scope

This project will constitute a mobile application which has all the essential features to manage mess subscription and coordination. Digimess allows students to login and logout, makes payment easier, constantly updates students about upcoming events, keeps track of payment history & file complaints.

2.Overall Description

2.1 Project features

- Account management
- Mess management
- Online payment

2.2 Operating environment

Operating environment is as given

- Android application client.
- Firebase noSQL database.
- Development OS: Windows.
- Platforms: Flutter/Dart, Android SDK.
- IDE: Android Studio.

2.3 Design and implementation constraints

- Administrative installation required.
- Payment gateway is UI only.
- Mess coordinators are on-boarded by the admin.
- Android app only.
- Lollipop or above android version required.
- Firebase free plan only includes 20,000 reads and 10,000 writes per day.

3. System Features and Requirements

3.1 Functional Requirements

1) Student requirements

- **1.1 Authorization** -
  - **1.1.1** Students should be able to register with details such as name, date of admission, date of completion, date of birth, admission number, phone number, email id, password, and food preference (veg/non-veg).
  - **1.1.2** Students should use their admission number and password to login.
  - **1.1.3** Students should also be able to log out whenever they want.
- **1.2 Online mess fee payment** -
  - **1.2.1** Students should be able to pay online using their card.
  - **1.2.2** Card number, expiration date, CVV and card holder's name should be provided.
  - **1.2.3** OTP should be sent for payment confirmation.
  - **1.2.4** Payment success/failure should be shown correspondingly.
- **1.3 Notices** -
  - **1.3.1** Students should be notified about upcoming mess events. These may include events such as mess holidays.
- **1.4 Daily menu** -
  - **1.4.1** Students should be able to see the daily menu (breakfast, lunch and dinner) on the dashboard as cards, along with the images and available time interval.
- **1.5 Available food** -
  - **1.5.1** Students should be able to search for all available food items in the mess.
  - **1.5.2** Veg/non-veg filters should be provided.

- **1.6 Mess card** -
  - **1.6.1** Students should have access to a mess card which indicates whether they are eligible to take food from mess or not.
  - **1.6.2** Only those who have paid a mess fee should be eligible.
- **1.7 Mess cut/leaves** -
  - **1.7.1** Students should be able to take mess cuts/leaves.
  - **1.7.2** The mess fee amount to be paid should depend on the number of leaves taken.
- **1.8 payment history** -
  - **1.8.1** Students should be able to keep track of their payment history.
- **1.9 Complaints & Suggestions** -
  - **1.9.1** Students should be able to file complaints against mess concerning hygiene issues, service issues and other categories.
- **1.10 Profile** -
  - **1.10.1** Students should have a profile page which shows their username and other details.
  - **1.10.2** Profile page should show mess card and current subscription details.
  - **1.10.3** Profile page should have an option to close account or log out.

2) Admin requirements

- **2.1 View subscription details** -
  - **2.1.1** Admin should be able to view all subscribed students and their details.
  - **2.1.2** Search and filter options should also be available.
- **2.2 Menu updation** -
  - **2.2.1** Admin should be able to update the menu from the available food items in the database.

- **2.3 Daily menu updation** -
    - **2.3.1** Admin should be able to update the day’s menu daily by adding breakfast, lunch and dinner food items.
- **2.4 Complaints review** -
    - **2.4.1** Admin should be able to review the complaints submitted by students.
- **2.5 Mess cut/leaves** -
    - **2.5.1** Admin should be able to check mess cuts/leaves taken by each student.
- **2.6 View payments** -
    - **2.6.1** Admin should be able to check for who all have paid/not paid the mess fee and update their mess card accordingly.

3.2 External Interface Requirements

1) Hardware Interface requirements

- **1.1** Android SDK version 21 and above should be supported.
- **1.2** All data should be only sent with the Firebase client library and not through any other REST api.

2) Software Interface requirements

- **2.1** App should communicate with firebase only.
- **2.2** Flutter app should run on dart VM using libraries for OS specific tasks.

3) Communication Interface requirements

- **2.3** Read and write requests to firebase should be minimized wherever possible.

3.3 Non Functional Requirements

1) Application Architecture

- **1.1** Application should follow MVI architecture with modularised code.

2) Application UI

- **2.1** UI should be minimal and concise with fluidic view transitions.
- **2.2** Proper colors should be used with appropriate and readable fonts.
- **2.3** Material design components should be used wherever possible.
- **2.4** An FAQ section about application's functionality and features should be provided.
- **2.5** User interface design should be as per design prototype.
- **2.6** Proper feedback with the user should be provided when performing tasks in the application.

3) Error Messages and Constraints

- **3.1** Error handling should be done appropriately with feedback to users in error messages.
- **3.2** Application should not crash or not work when an error occurs.

4) Scalability

- **4.1** Application should be scalable to more features in the future and expandable to more users.

5) Compatibility

- **5.1** Application should not conflict with other processes or applications running in the same environment.

- **5.2** Application should be asynchronous and should not block the main UI thread.
- **5.3** Shared data race conditions should be avoided.
- **5.4** Application should be portable and easily distributable while supporting latest android versions.

6) Security

- **6.1** Users data should be stored securely with proper encryption.
- **6.2** Proper security rules should be followed so as to give different levels of accessibility to data.
- **6.3** Proper implementations should be made to detect and rectify memory leaks.