Department of Computer Science

University of Gujrat

TITLE

Feed Humble

Session: BSCS 20

Project Advisor: Mr. Muhammad Jabbar

Submitted By

Husnain Afzal Cheema	20021519- 126
Asad Ali	20021519-114
Umair Raza	20021519-098

STATEMENT OF SUBMISSION

This is certify **Husnain Afzal Cheema** Roll No. **20021519- 126**, **Asad Ali** Roll No. **20021519-114** and **Umair Raza** Roll No **20021519-098** has successfully completed the final year project named as **Feed Humble** at the Department of Computer Science, University of Gujrat, to fulfill the requirement of the degree of **BS in Computer Science**.

Project Supervisor

Project Coordination Office
Faculty of C&IT -UOG

Chairperson
Department of Computer Science

Acknowledgement

We truly acknowledge the cooperation and help make by **Mr. Muhammad Jabbar** Chairman, Department of Computer Science, University of Gujrat. He has been a constant source of guidance throughout the course of this project. We would also like to thank ______________________ for his help and guidance throughout this project. We are also thankful to our friends and families whose silent support led us to complete our project.

Date:

Abstract

Due to ever increasing demand of transporting huge amount of information generated from various sources such as voice, data, video, etc., modern telecommunication networks have been transformed into all digital and broadband. Depending on the characteristics of information sources and the availability of facility, the mode of transportation can be either constant bit rate (CBR) using circuit switched networks or variable bit rate (VBR) using packet switched networks. For efficient utilization of the network, all kinds of information can be transported using BISDN (Broadband Integrated Services Digital Network) and ATM (Asynchronous Transfer Mode) technology. One important research area in Network Technology is the design of high-speed digital network with good performance. The issues need to be investigated include modeling of Variable Bit Rate video traffic, efficient assignment of different traffic classes with diverse quality of services, optimal bandwidth allocation, routing and call admission control etc. This project not only relates to study of Digital Subscriber Line, which is a Broadband technology to provide high-speed data, voice and video but also addresses the above-mentioned issues. What are the provisions made in DSL implement QoS quality of service.

TABLE OF CONTENTS

Chapter 1: Project Feasibility Report

1.1. Introduction

Feed Humble is a mobile application designed to facilitate the donation of surplus food to those in need. The app connects individuals and businesses with excess food to local people and shelters, ensuring that no food goes to waste. This project combines technology and altruism to make a meaningful impact on the community, promoting sustainability and social responsibility.

1.2. Project/Product Feasibility Report

When initiating a project like Feed Humble, it is crucial to first assess its feasibility. Feasibility refers to the extent to which appropriate data and information are readily available or can be obtained using available resources such as staff, expertise, time, and equipment. This assessment serves as a measure of how practical or beneficial the development of the Feed Humble application will be. Feasibility evaluations recur throughout the project lifecycle to ensure ongoing viability and alignment with goals.
There are many types of feasibilities:

- Technical
- Operational
- Economic
- Schedule
- Specification
- Information
- Motivational
- Legal and Ethical

1.2.1. Technical Feasibility

Technical feasibility involves determining whether the Feed Humble application can be developed with the current technology and expertise. This is a critical initial step, as it assesses the theoretical and practical limits of the technology applicable to the project. Another key consideration is whether the project team possesses the necessary technical skills and resources to develop the app effectively.

1.2.2. Operational Feasibility

Operational feasibility evaluates whether the staff has the capability to operate and maintain the Feed Humble application. This includes assessing whether the problem of food waste and hunger is significant enough to warrant a technological solution and whether the proposed solution is effective. It also involves gauging the acceptance of the solution by end-users and managers.

1.2.3. Economic Feasibility
Economic feasibility justifies the project through a benefit/cost analysis. This includes dividing costs into development or acquisition costs (one-time) and maintenance and operation costs (ongoing). Development costs can be estimated by breaking the project into tasks and using lifecycle cost models, leveraging experiences from similar projects. Benefit estimates encompass both tangible benefits (e.g., reduced costs, increased revenues) and intangible benefits (e.g., improved information quality, job satisfaction, external reputation).

1.2.4. Schedule Feasibility
Schedule feasibility assesses whether the Feed Humble project can be completed within the available timeframe, given the staff and resources. This involves evaluating the ability to meet deadlines and milestones, which is crucial for project success.

1.2.5. Specification Feasibility
Specification feasibility focuses on the clarity and definiteness of the project requirements. It involves ensuring that the system's features are well-defined and that the scope boundaries are clearly understood and feasible.

1.2.6. Information Feasibility
Information feasibility assesses the completeness, reliability, and meaningfulness of the data required for the project. This ensures that the information needed to develop and operate the Feed Humble application is accurate and adequate.

1.2.7. Motivational Feasibility
Motivational feasibility evaluates the readiness and willingness of the client and staff to perform the necessary steps for project success. This involves assessing whether the team is motivated to implement the Feed Humble application correctly and promptly.

1.2.8. Legal & Ethical Feasibility
Legal and ethical feasibility examines any potential legal issues or ethical considerations arising from the project. This includes ensuring that the Feed Humble application complies with relevant laws and regulations and does not pose any ethical dilemmas or liabilities.

1.3 Project/Product Scope

Scope is a very dominant factor. Scope and context are both intertwined as both involve the boundaries of a system. Context would be referring to what holds outside the boundary the system. While scope would indicate whatever is inside the boundary of the system.

The scope of a project is defined by the set of requirements allocated to it. Managing project scope to fit the available resources (time, people, and money) is key to managing successful projects. Managing scope is a continuous activity that requires iterative or incremental development, which breaks project scope into smaller more manageable pieces.

1.3.1. Online Donation Listing System:

This feature allows donors (individuals and businesses) to list surplus food items for donation through an online platform. It also enables users to view the availability of food donations in real-time and claim donations accordingly.

1.3.2. Automated Notification System:

Feed Humble can integrate an automated notification system, notifying recipients (charities and shelters) of new food donations and nearby available donations. The system can also provide users with real-time updates on donation statuses.

1.3.3. Mobile Application:

A mobile application can be developed for Feed Humble that allows users to list and claim food donations, view food availability, and receive notifications. This feature can also enable users to view their donation history and manage their account details.

1.3.4. Donation Analytics:

Feed Humble can include a donation analytics feature that provides users with data on donation usage, including peak donation times, frequency of donations, and impact metrics.

1.3.5. Integration with Navigation Systems:

Feed Humble can integrate with navigation systems to provide users with real-time information on the locations of food donations. This feature can enable recipients to find donation sites quickly and efficiently.

1.4. Project/Product Costing

The Project was completely code based which allowed us to do build it without having to pay for anything.

1.5. Task Dependency Table

Task	Task Name	Duration(days)	Dependencies
T1	Idea Selection	7	
T2	Proposal defense	3	T1
T3	Requirement gathering	10	T2
T4	Prototyping, Layout Design	48	T3
T5	Development	40	T3, T4
T6	Backend Coding	20	T4, T5
T7	Database Connectivity	24	T6
T8	Software testing with dummy data	7	T7

1.6. CPM - Critical Path Method

As the project progresses, the actual task completion times will be known and the network diagram can be updated to include this information. A new critical path may emerge, and structural changes may be made in the network if project requirements change.

1.6.1 **Project Initialization**:

- Define project scope and requirements.
- Set up Flutter environment.
- Set up Firebase.

1.6.2 User Authentication:

- Implement login functionality.
- Implement signup functionality.
- Integrate authentication with Firebase.

1.6.3 Home Page (Restaurants):

- Design the home page layout.
- Fetch restaurant data from Firebase.
- Display restaurant list with details (name, address, ratings).

1.6.4 Restaurant Detail Page:

- Design the restaurant detail page layout.
- Fetch detailed restaurant data (available food, donation status).
- Display food availability and donation status.

1.6.5 Donation Page:

- Design the donation page layout.
- Implement form for donor input (restaurant name, address, servings available).
- Save donation data to Firebase.

1.6.6 History Page:

- Design the history page layout.
- Fetch donation and received food history from Firebase.
- Display history records.

1.6.7 Navigation Setup:

- Implement navigation bar.
- Ensure proper navigation between pages (Home, Donation, History).

1.6.8 Testing and Debugging:

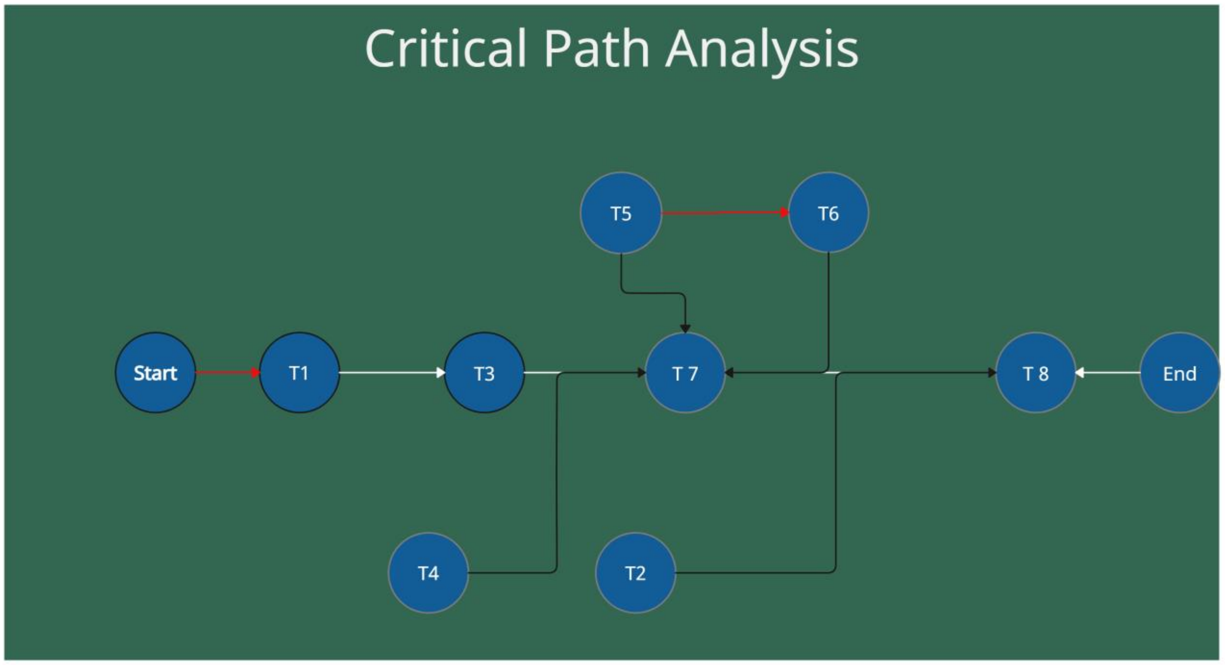
- Test user authentication flow.
- Test data retrieval and display for all pages.
- Test donation functionality.
- Test navigation and overall app stability.

1.6.9 Deployment:

- Prepare app for deployment.
- Deploy app to the desired platform (e.g., Google Play Store).

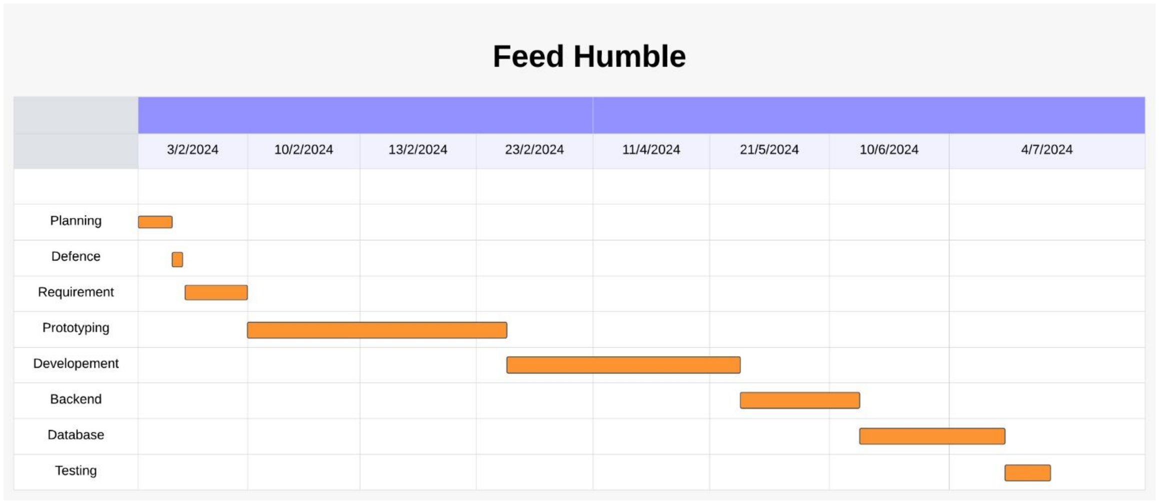
Example:

Task ID	Task Name	Duration (Weeks)	Dependencies
T1	Project Initialization	3	-
T2	User Authentication	5	T1
T3	Home Page	4	T1
T4	Restaurant Detail Page	3	T3
T5	Donation Page	4	T1
T6	History Page	4	T1
T7	Navigation Setup	3	T3, T5, T6
T8	Testing and Debugging	5	T2, T4, T7
T9	Deployment	2	T8



1.7. Gantt chart

ID	Tasks	Start Date	End Date	Estimated	Predecessors
1	Planning	3/2/2024	10/2/2024	7 Days	
2	Defense	10/2/2024	13/2/2024	3 Days	1
3	Requirement	13/2/2024	23/2/2024	10 Days	2
4	Prototyping	23/2/2024	11/4/2024	48 Days	3
5	Development	11/4/2024	21/5/2024	40 Days	3,4
6	Backend	21/5/2024	10/6/2024	20 Days	4,5
7	Database	10/6/2024	4/7/2024	24 Days	6
8	Testing	4/7/2024	11/7/2024	7 Days	6



1.8. Allocation of Members to Activities

A brief but a concise introduction of the team members should be provided signifying their skill set. This skill set would especially be representative of the tasks and activities assigned to him.

Task ID	**Task Name**	**Duration**	**Dependencies**	**Assigned Member**
T1	Project Initialization	3 Weeks	-	Member A
T2	User Authentication	5 Weeks	T1	Member B
T3	Home Page	4 Weeks	T1	Member A
T4	Restaurant Detail Page	3 Weeks	T3	Member B
T5	Donation Page	4 Weeks	T1	Member C
T6	History Page	4 Weeks	T1	Member C
T7	Navigation Setup	3 Weeks	T3, T5, T6	Member A
T8	Testing and Debugging	5 Weeks	T2, T4, T7	Member A, B
T9	Deployment	2 Weeks	T8	Member C

1.9. Tools and Technology with reasoning

The development of the Feed Humble application requires a thoughtful selection of tools and technologies for both the front-end and back-end systems. The reasons for choosing these tools are based on various factors such as the development process, platform requirements, programming languages, existing tools, organizational distribution, development effort size, and budget and time constraints.

1.9.1 Development Process

Given that Feed Humble is likely to employ an iterative development process, automated testing tools are essential. This will enable continuous integration and testing throughout the project's lifecycle, ensuring high quality and rapid deployment of new features.

1.9.2 Host (or Development) Platforms

- **Operating Systems:** Windows, macOS, Linux
- **IDEs:** Visual Studio Code, Android Studio

Reason: These platforms and IDEs support a wide range of development tools and languages, facilitating a versatile development environment.

1.9.3 Target Platforms

- **Mobile Operating Systems:** iOS, Android

Reason: The primary target users will access the application via mobile devices. Developing for both iOS and Android ensures broad accessibility and user reach.

1.9.4 Programming Languages

- **Front-End:** Dart
- **Back-End:** Firebase
- **Database:** Firebase

Reason: Flutter allows for cross-platform mobile application development, reducing development time and effort. Firebase provides a comprehensive backend infrastructure, including real-time database, authentication, and cloud storage, which simplifies backend development and maintenance.

1.10. Vision Document

The Vision Document outlines the high-level goals, objectives, and desired outcomes of the Feed Humble project. It serves as a guiding document for the project team.

- **Goal**: To create an innovative, user-friendly, and impactful mobile application that facilitates the donation of surplus food to those in need.
- **Objectives**:
    - ✓ To reduce food waste by connecting donors (individuals and businesses) with consumers and shelters.
    - ✓ To enhance food security for underprivileged communities by providing easy access to surplus food.
    - ✓ To promote sustainability and social responsibility within the community.
    - ✓ To streamline the donation process for users through technology-driven solutions.
    - ✓ To build a scalable and efficient system that can grow and adapt to future needs.

1.11. Product Features/ Product Decomposition

The main features of the Feed Humble application are:

- **Real-time Food Donation Availability**:
    - ✓ Up-to-date listings of available food donations.
    - ✓ Notifications for nearby or new food donations.
- **Mobile App**:
    - ✓ User-friendly mobile application for easy access and use.
    - ✓ Available on both Android and iOS platforms.
- **Food Donation Listing**:
    - ✓ Easy-to-use interface for donors to list available surplus food.
    - ✓ Detailed descriptions and images of food donations.
- **User Profiles**:
    - ✓ Donor and recipient profiles to manage donations and claims.
    - ✓ Personalized dashboards for tracking activities and history.
- **Charity and Shelter Management**:
    - ✓ Tools for charities and shelters to manage and claim food donations.
    - ✓ Administrative dashboard for monitoring and reporting.
- **Flutter Frontend**:
    - ✓ Cross-platform mobile development framework for a seamless user experience on both Android and iOS.
- **Firebase Backend**:
    - ✓ Scalable and flexible backend infrastructure using Firebase for real-time database management, authentication, and cloud storage.

Chapter 2: Software Requirement Specification (For Object Oriented Approach)

2.1 Introduction:

In an object-oriented approach, the Feed Humble application can be designed by identifying the key objects involved in the system and their interactions. For instance, you could have objects like "Donor," "Consumer," and "Food Donation." Each object would possess its own properties and behaviors, representing users, donated food items, and donation transactions. These objects would interact with each other through methods and messages, facilitating the listing, claiming, and management of food donations. This approach aids in organizing and modularizing the codebase, enhancing maintainability and scalability for future enhancements and updates.

2.2 Systems Specifications

In the object-oriented approach for the Feed Humble application, the system specification would involve defining the classes, objects, and their relationships. Here are some key components to consider:

1. Consumer Class.
2. Admin Class.
3. History Class.
4. Restaurant Class.
5. Food Donation Class.

2.2.1. Identifying External Entities

- **Users**:

  These are individuals utilizing the Feed Humble mobile application to search for and claim surplus food donations.

- **Restaurants:**

  Entities responsible for listing surplus food donations available for donation through the Feed Humble platform.

2.2.2. Context Level Data Flow Diagram:

Context level data flow diagram contains only one process, representing the entire system. The process is given the number zero and all external entities are shown on the context diagram as well as major data flow to and from them. The diagram does not contain any data stores.

2.2.3. Capture "shall" Statements:

Identify “shall” statements, as they would be all functional requirements.

Para	Initial Requirements
1.0	A user “shall” register themselves to the system.
1.0	The system “shall” provide user authentication with login and signup processes.
1.0	The system “shall” verify the user’s credentials during login.
1.0	A user “shall” log in to the system and can change their password.
1.0	The system “shall” store and manage user details in Firebase.
1.0	The system “shall” update user details upon request (e.g., personal information, authentication details).
2.0	The system “shall” display a list of restaurants with their names, addresses, and ratings.
2.0	A user “shall” view the details of a selected restaurant, including available food for donation and donation history.
2.0	A donor “shall” input and submit details of food donations, including the restaurant name, address, and quantity of food.
2.0	The system “shall” update the restaurant’s food availability and donation records based on donor input.
2.0	The system “shall” allow privileged users (e.g., verified donors) to update or cancel their donations if the food has not been distributed.
2.0	The system “shall” generate and display a history of donated and received food for users to view.
3.0	An action event “shall” be generated for an administrator when a request is placed for updating food donations or user details.
3.0	The corresponding administrator “shall” view an action list containing different actions and process these pending actions.
3.0	The administrator “shall” accept, reject, or temporarily waive requests based on the provided credentials and validity of the requests.
3.0	When the action processing is completed or if the action is just a notification message, the administrator “shall” delete these actions from the action list.

2.2.4 Allocate Requirements:

Para	Initial Requirements	Use Case Name
1.0	A customer "will" place order for goods.	UC_Place_Order
1.0	A customer "shall" register himself to the system.	UC_Registration_Request
1.0	The system "shall" provide two types of registration process, normal and privileged.	UC_Place_Order_Request
1.0	CA "shall" accept, reject and temporarily waive the requests on the basis of credentials provided.	UC_Process_Customer_Request
1.0	A customer "shall" login to the system and can change his password.	UC_Login
1.0	System "shall" update the customers Request.	UC_Update_Request
1.0	System "shall" process different types of updating e.g. updating of his personal/shipping details, or upgrading of his status from registered to privileged customer, or updating of his payment methodology.	UC_Change_Status
1.0	A customer "shall" view his details for verification purposes.	UC_View_Customer_Details
1.0	System "shall" search any customer details.	UC_Search_Customer
1.0	CA "shall" accept, reject and temporarily waive the requests on the basis of credentials provided.	UC_Accept_Customer_Request UC_Reject_Customer_Request UC_View_Customer_Request
2.0	Both registered and privileged customers "will" order for goods.	UC_Place_Order_Privileged
2.0	Customer "will" make payment; either through cash or through a credit card.	UC_Pay_For_Order
2.0	System "will" generate invoice, confirmation receipt and finally will place order.	UC_Invoice_Generation
2.0	User "shall" view the status of their orders by providing the Order Number.	UC_Serach_Orders
2.0	Privileged customers "shall" place the request for the updating of their orders if the orders are not shipped.	UC_Update_Request
3.0	The System "shall" generate an action event for a corresponding administrator when a request is placed for updating of orders or customer details etc.	UC_Create_Action
3.0	Corresponding administrator "shall" view his Action List containing different actions, and correspondingly process these pending actions.	UC_View_Action

2.2.5. Prioritize Requirements (If Any):

Para	Rank	Initial Requirements	Use Case ID	Use Case Name
1.0	Highest	A customer “will” place order for goods.	UC_1	UC_PlaceOrder
1.0	Highest	A customer “shall” register himself to the system.	UC_2	UC_Registration_Request
2.0	Highest	Customer “will” make payment either through cash or through a credit card.	UC_3	UC_Pay_For_Order
2.0	Highest	System “will” generate invoice, confirmation receipt and finally will place order.	UC_4	UC_Invoice_Generation
2.0	Medium	Both registered and privileged customers “will” order for goods.	UC_5	UC_Place_Order_Privleged
1.0	Medium	The system “shall” provide two types of registration process, normal and privileged.	UC_6	UC_Place_Order_Request
3.0	Medium	The System “shall” generate an action event for a corresponding administrator when a request is placed for updating of orders or customer details etc.	UC_7	UC_Create_Action
1.0	Medium	CA “shall” accept, reject and temporarily waive the requests on the basis of credentials provided.	UC_8, UC_9, UC_10	UC_Accept_Customer_Request, UC_Reject_Customer_Request, UC_View_Customer_Request
1.0	Medium	System “shall” update the customers Request.	UC_11	UC_Update_Request
1.0	Medium	A customer “shall” view his details for verification purposes.	UC_15	UC_View_Customer_Details
1.0	Medium	System “shall” search any customer details.	UC_16	UC_Search_Customer
2.0	Medium	User “shall” view the status of their orders by providing the Order Number.	UC_17	UC_Serach_Orders
2.0	Medium	Privileged customers “shall” place the request for the updating of their orders if the orders are not shipped.	UC_18	UC_Update_Request
2.0	Medium	Privileged customer “shall” place the request for the cancellation of the order. But all these updating and cancellation requests are to be viewed by the Order Administrator in order to accept, reject, or waive them.	UC_19, UC_20, UC_21	UC_View_All_Orders, UC_Manage_Order
1.0	Lowest	A customer “shall” login to the system and can change his password.	UC_22, UC_23	UC_Login
3.0	Lowest	Corresponding administrator “shall” view his Action List containing different actions, and correspondingly process these pending actions.	UC_24	UC_View_Action
3.0	Lowest	When the action processing is completed or if the action is just a notification message then administrator “shall” delete these actions from the action list.	UC_25	UC_Delete_Action

2.3. Existing Systems / Literature Review:

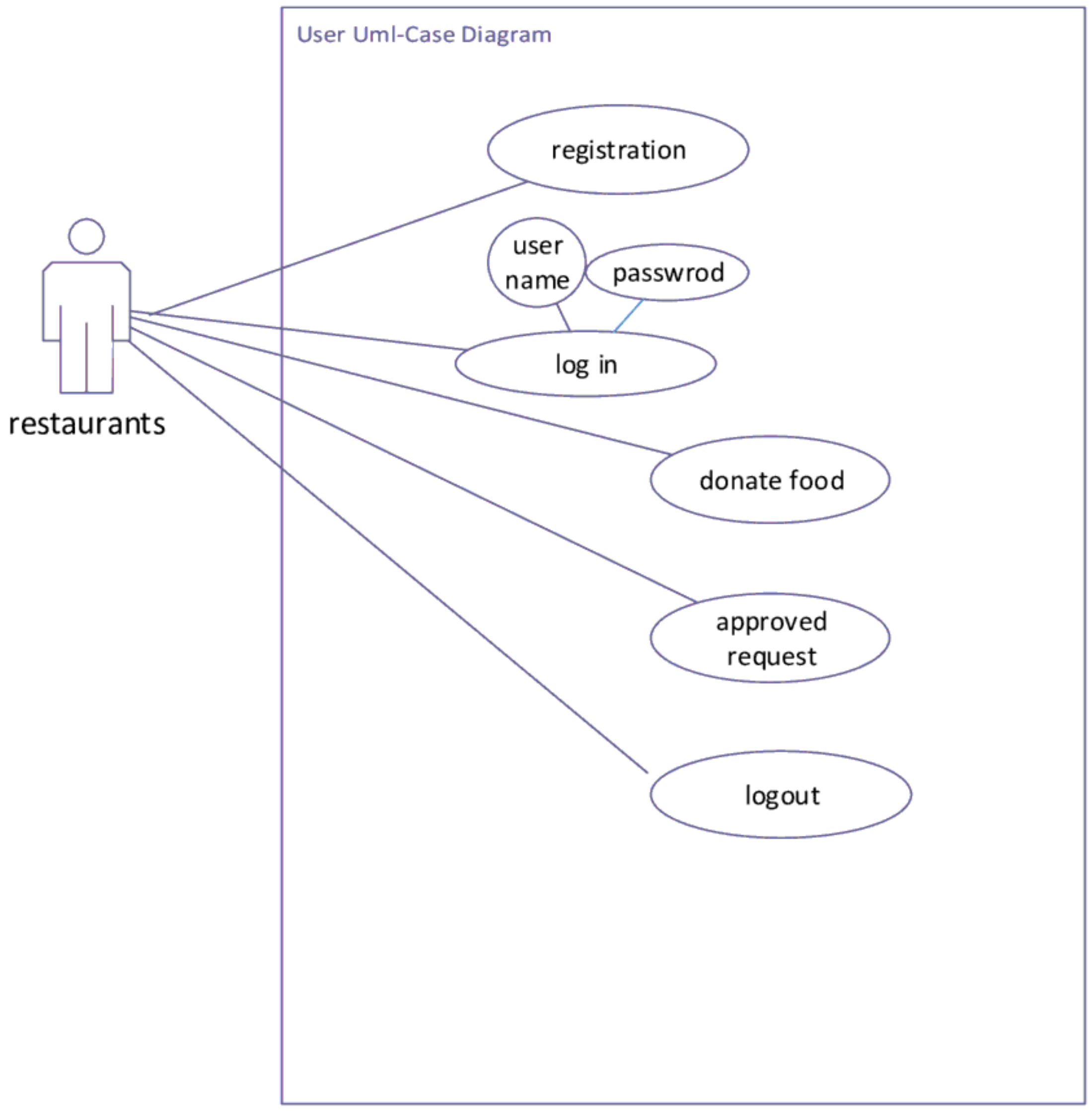
- **System A:**
  - ✓ Provides a platform for users to donate food and resources to those in need.
  - ✓ Offers features for users to view and select nearby donation centers or individuals in need.
  - ✓ Does not include any payment method, all services are free to help the needy and poor.

- **System B:**
  - ✓ Facilitates the donation of surplus food from restaurants and individuals to charities and shelters.
  - ✓ Allows users to schedule food pickups and track the donation process.
  - ✓ Ensures that all donations are free and aimed at assisting those in need without any monetary transactions.

- **Literature Review:**
  - ✓ Various studies and articles highlight the importance of providing free assistance to the needy and poor, emphasizing the social responsibility of individuals and organizations.
  - ✓ The literature emphasizes the need for efficient systems to facilitate donations and aid distribution without financial barriers.

- **Comparison:**
  - ✓ Both System A and System B align with the concept of providing free assistance to the needy and poor, focusing on efficient donation processes without involving payment methods.
  - ✓ The literature review supports the idea of free assistance as a crucial aspect of social welfare and community support.

2.4. Use Case Diagram of Your Project:



2.4.1. Use Case Description

- **View Restaurants:**
  - ✓ Actors: User
  - ✓ Description: Allows users to view a list of restaurants offering food donations.
- **View Profile:**
  - ✓ Actors: User
  - ✓ Description: Displays the user's details (name, email, etc.) and provides an option to edit the profile.
- **Logout:**
  - ✓ Actors: User
  - ✓ Description: Logs the user out of the system.
- **View Restaurant Details:**
  - ✓ Actors: User
  - ✓ Description: Shows detailed information about a selected restaurant, including available food items and donation status.
- **View Donation Form:**
  - ✓ Actors: User
  - ✓ Description: Displays a form for the user to fill out with details about the donation (restaurant name, address, number of servings available).
- **Submit Donation:**
  - ✓ Actors: User
  - ✓ Description: Submits the donation information to the system for processing and updates the restaurant's donation status.
- **View History:**
  - ✓ Actors: User
  - ✓ Description: Shows the complete history of received and donated food items.

Chapter 3: Design Document (For Object Oriented Approach)

3.1. Introduction:

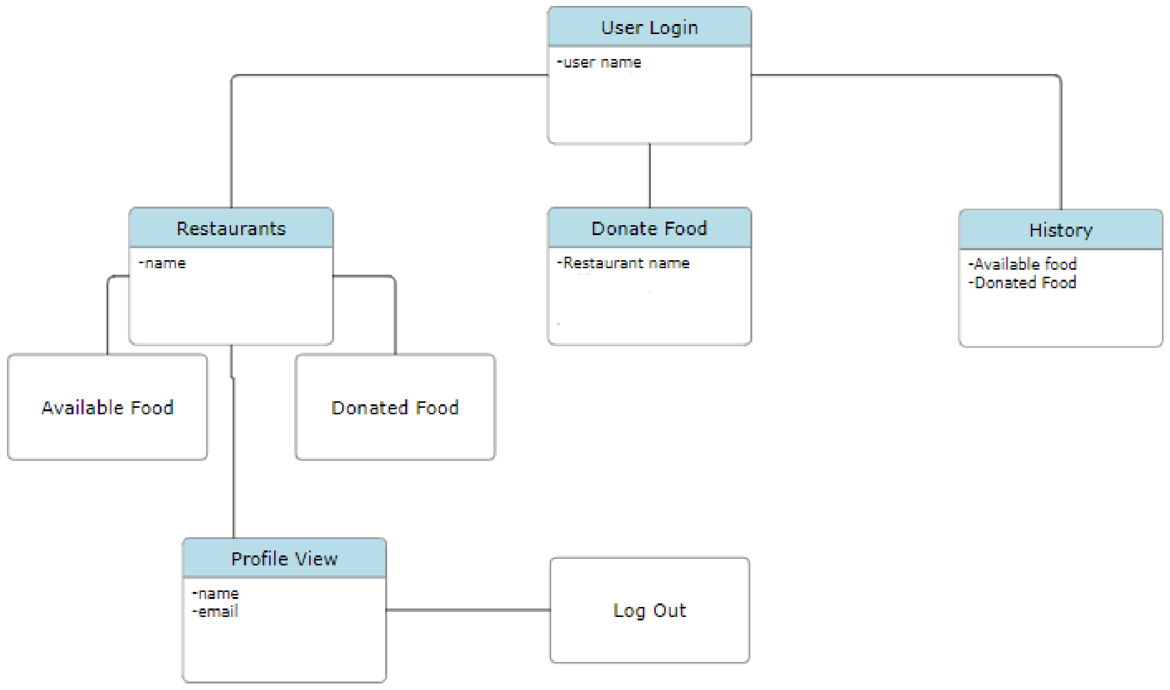
Second deliverable is all about the software design. In the previous deliverable, analysis of the system is completed. So we understand the current situation of the problem domain. Now we are ready to strive for a solution for the problem domain by using object-oriented approach. Following artifacts must be included in the 3rd deliverable.

1. Domain Model
2. Design Class Diagram
3. Sequence Diagram
4. State Chart Diagram
5. Collaboration Diagram

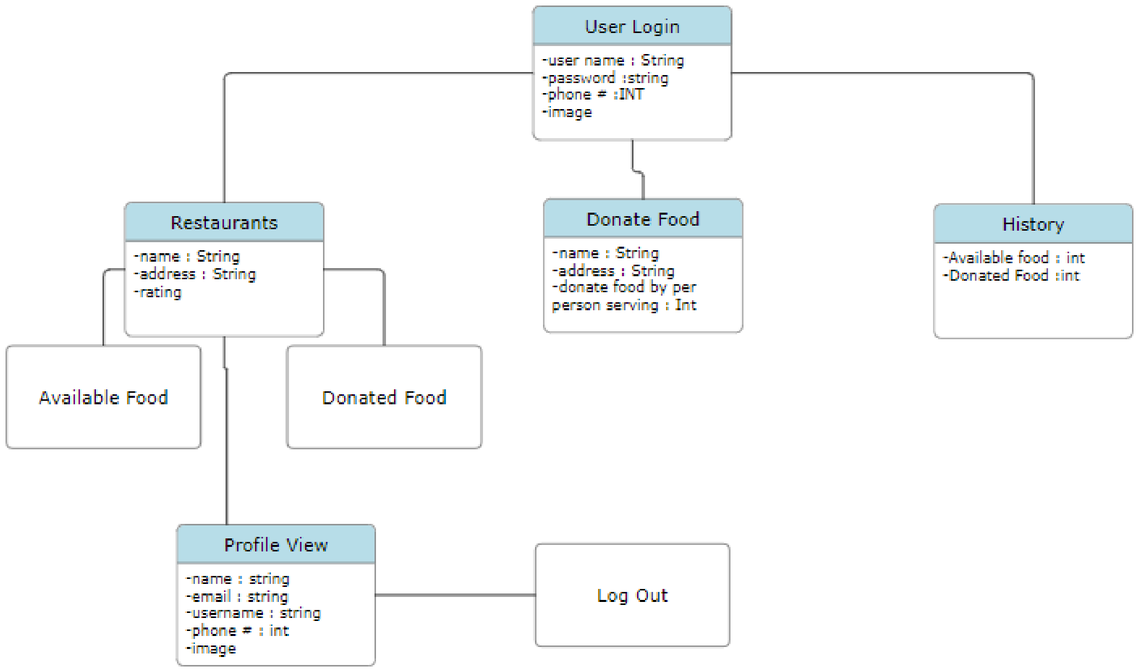
Now we discuss these artifacts one by one as follows:

3.2. Domain Model

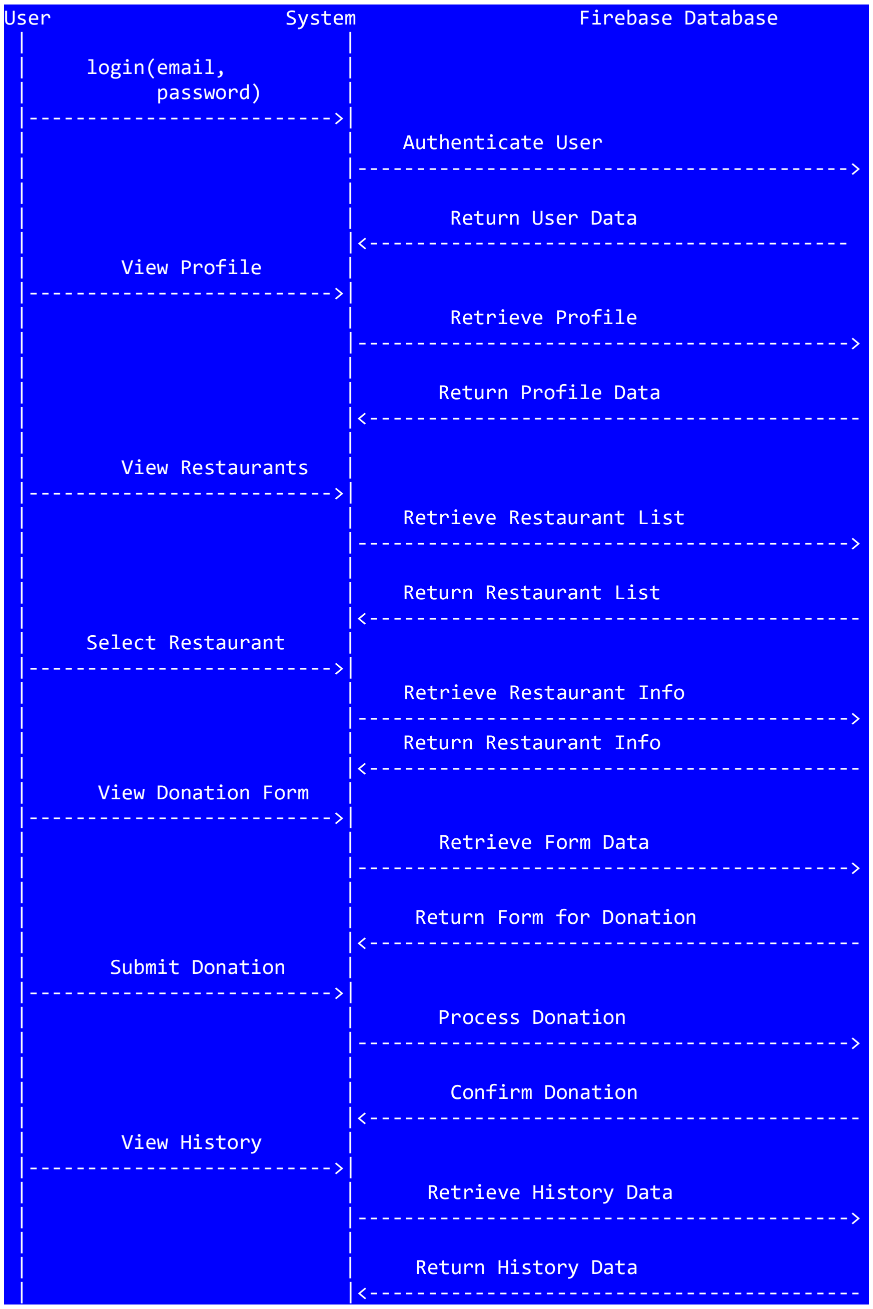
Domain model, entities such as "Donor", "Recipients", "Food Donation" and "Reservation are typically included. The Donor entity represents individuals or businesses offering surplus food donations, which are represented by the Food Donation entity. Recipients, such as consumers or shelters, claim these donations through the Reservation entity. Upon claiming a donation, a Reservation is created, and when the recipient receives the donation, the Payment entity may be processed if applicable. This model facilitates the management and tracking of food donations, ensuring an efficient process for donors and recipients.



3.3. Design Class Diagram



3.4. Sequence Diagram

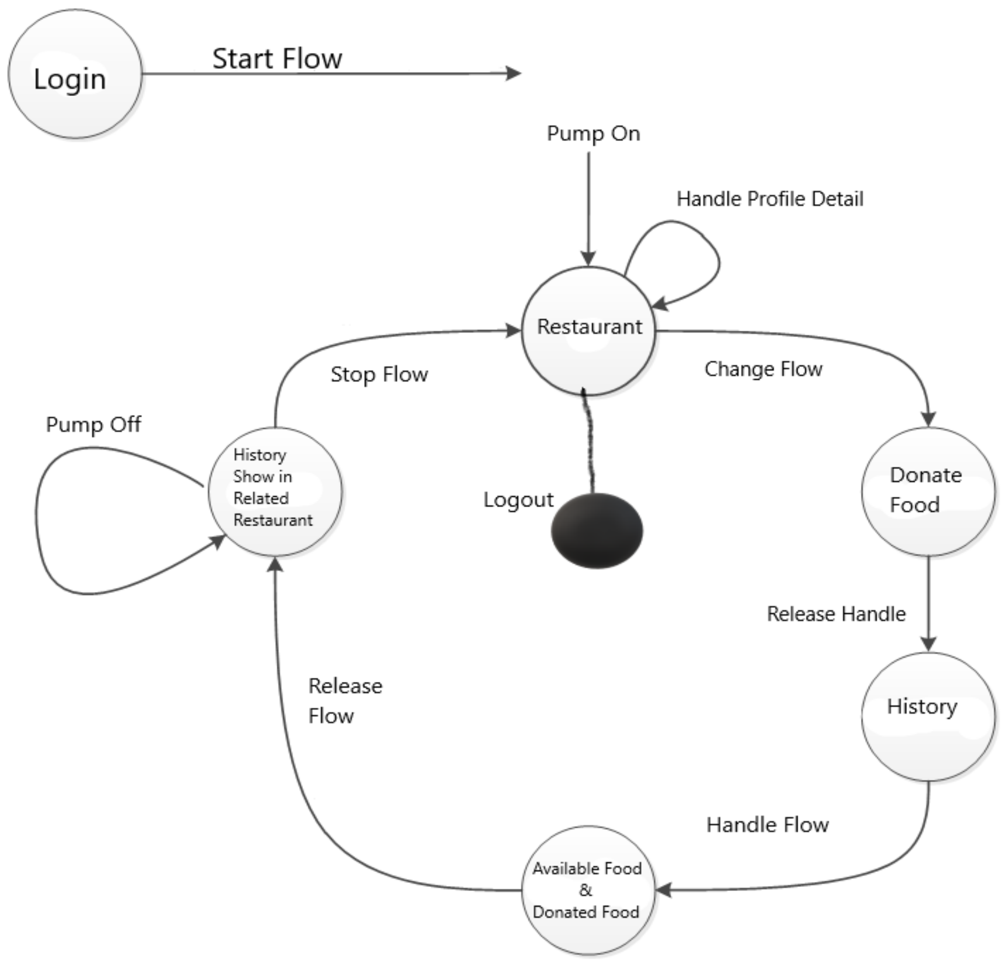


```
User                    System                    Firebase Database
 |                         |
 |     login(email,        |
 |           password)     |
 |------------------------>|
 |                         |    Authenticate User
 |                         |------------------------------------------->
 |                         |
 |                         |        Return User Data
 |                         |<------------------------------------------
 |        View Profile     |
 |------------------------>|
 |                         |        Retrieve Profile
 |                         |------------------------------------------->
 |                         |
 |                         |       Return Profile Data
 |                         |<-------------------------------------------
 |                         |
 |        View Restaurants |
 |------------------------>|
 |                         |    Retrieve Restaurant List
 |                         |------------------------------------------->
 |                         |
 |                         |    Return Restaurant List
 |                         |<-------------------------------------------
 |     Select Restaurant   |
 |------------------------>|
 |                         |    Retrieve Restaurant Info
 |                         |------------------------------------------->
 |                         |    Return Restaurant Info
 |                         |<-------------------------------------------
 |       View Donation Form|
 |------------------------>|
 |                         |       Retrieve Form Data
 |                         |------------------------------------------->
 |                         |
 |                         |     Return Form for Donation
 |                         |<-------------------------------------------
 |       Submit Donation   |
 |------------------------>|
 |                         |       Process Donation
 |                         |------------------------------------------->
 |                         |
 |                         |        Confirm Donation
 |                         |<-------------------------------------------
 |        View History     |
 |------------------------>|
 |                         |      Retrieve History Data
 |                         |------------------------------------------->
 |                         |
 |                         |     Return History Data
 |                         |<-------------------------------------------
```

In this sequence diagram:

- ✓ The user first logs in with their email and password.
- ✓ After authentication, the user can view their profile, restaurants, and donation history.
- ✓ When the user selects a restaurant, they can view its details and proceed to donate.
- ✓ The system interacts with the Firebase database to retrieve and store data related to the user, restaurants, and donations.

3.5. State chart diagram



- **State Descriptions:**

  - ✓ Logged Out: The initial state when the user is not logged in.
  - ✓ Logged In: Transitioned to after a successful login.
  - ✓ Viewing Profile: When the user is viewing their profile information.
  - ✓ Viewing Restaurants: When the user is browsing available restaurants.
  - ✓ Selecting Restaurant: After the user selects a restaurant to interact with.
  - ✓ Submitting Donation: After the user submits the donation form.
  - ✓ Viewing History: When the user is viewing their donation history.

- **State Transitions:**

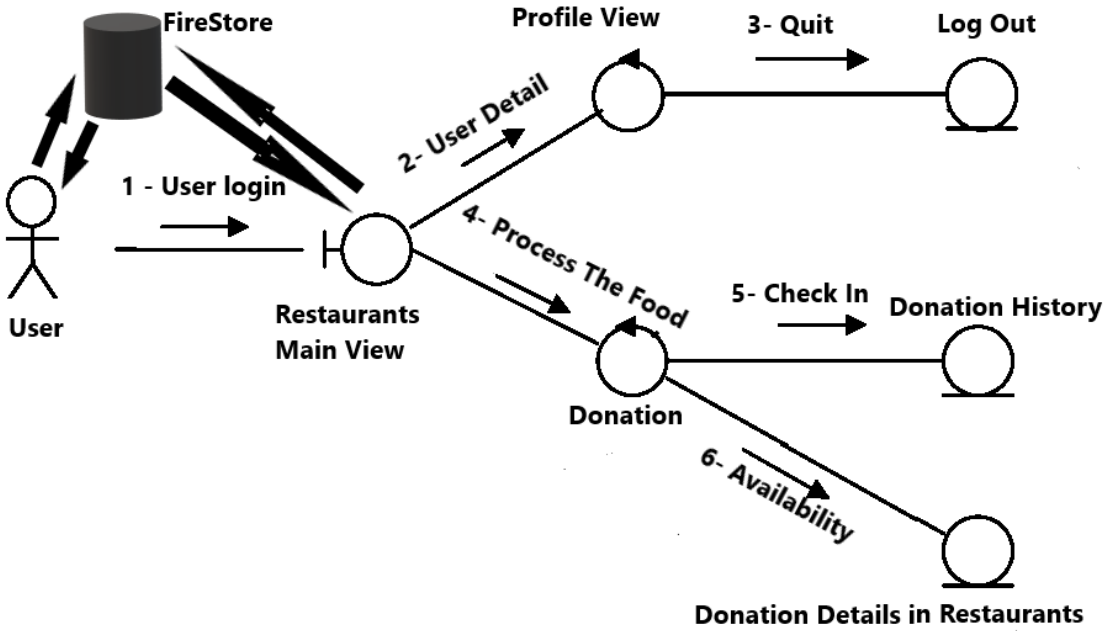
  - ✓ Login: Transition from Logged Out to Logged In.
  - ✓ View Profile: Transition from Logged In to Viewing Profile.
  - ✓ View Restaurants: Transition from Logged In to Viewing Restaurants.
  - ✓ Select Restaurant: Transition from Viewing Restaurants to Selecting Restaurant.
  - ✓ Submit Donation: Transition from Viewing Donation Form to Submitting Donation.
  - ✓ View History: Transition from Logged In to Viewing History.

3.6. Collaboration Diagram

Collaboration diagram for the "Humble Feed" system, illustrating the interactions between objects in the system to facilitate food donations:

In this detailed collaboration diagram:

- ✓ The user first logs in, triggering the login(email, password) method.
- ✓ After login, the user can view a list of available restaurants using viewRestaurants().
- ✓ The getRestaurantList() method in the Restaurant object retrieves the list of restaurants from the Firebase database.
- ✓ The user can also view their profile information, update it, and view donation history, which involves interactions with the Firebase database and possibly the History object.
- ✓ The Firebase component is responsible for handling user authentication, retrieving and updating user profiles, and managing restaurant and donation data.
- ✓ The History object may interact with the Firebase component to retrieve and display donation history for the user.



Chapter 4: User Interface Design

4.1. Introduction

A user interface design consists of three main parts:
Page elements should be visualized on paper before building them in the computer. Just as you draw a site map to plan the site, use cartoons and storyboards to begin blocking out the site's appearance and navigational scheme.

1. Site maps
2. Storyboards
3. Navigational maps

4.2. Site Maps

A site map's main benefit is to give users an overview of the site's areas in a single glance by dedicating an entire page to a visualization of the information architecture. If designed well, this overview can include several levels of hierarchy, and yet not be so big that users lose their ability to grasp the map as a whole. Some of the site maps we studied stretched over six screens on a standard 800x600 monitor. This is much too much. We recommend keeping the site map short; it should be no more than two-and-a-half times the window size most common among your users.

The greatest failures in our study came from site maps that attempted to lure the user into a dynamically twisting and expanding view, rather than presenting a simple, static representation of the information architecture. The site map's goal is to give users a single overview of the information space. If users have to work to reveal different parts of the map, that benefit is lost.

Dynamic site maps are basically an alternative way of navigating through the information space using a set of non-standard interaction techniques. For example, one site used a hyperbolic tree, where users had to click and drag clusters of links around the screen to expand areas of interest. Nobody could do this well.

A site map should not be a navigational challenge of its own. It should be a map.

As we have found again and again, users hated non-standard user interfaces that forced them to learn a special way of doing things for the sake of a single website. Site maps should be simple, compact layouts of links, and they should show everything in a single view.

If your site is large and complex, it is a good idea to include and index or site map that provides an outline of our site's content. This concept is something that we are all familiar with in the print world and it is very useful on the Web. An index is usually an alphabetical listing of key words that link to the appropriate pages in the site or it can be more like a table of contents. A site map is a graphical representation of the site's content. It doesn't usually have as much detailed information as the index has.

4.3. Story boards

A storyboard is a sequence of single images, each of which represents a distinct event or narrative. It is also a visual representation of the script illustrating the interaction between the user and the machine. It can also be imagined as a film in visual-outline form.
A storyboard can be used in two ways. It describes the task, which are a series of images showing the user, environment and the machine. It also describes the interface, which represent series of screen images indicating the user's representation and the computer's response and work out interaction details when asking, "what happens next?" It also shows interaction sequence at a glance and helps develop usage scenarios to help develop tools & tasks.
All this can be done to construct a visual & verbal sequence that illustrates the interaction. Consider …

- Environment -- where system is used
- Visual cues -- what user can see
- Audible cues -- what user can hear
- Tactile cues -- what user can touch
- User input -- how the user communicates to the machine
- Machine output -- how the machine responds to the user
- User's emotions -- how user perceives and responds to the interaction
- Technology -- what technology is involved in performing the task
- Quality of experience -- what benefit is perceived

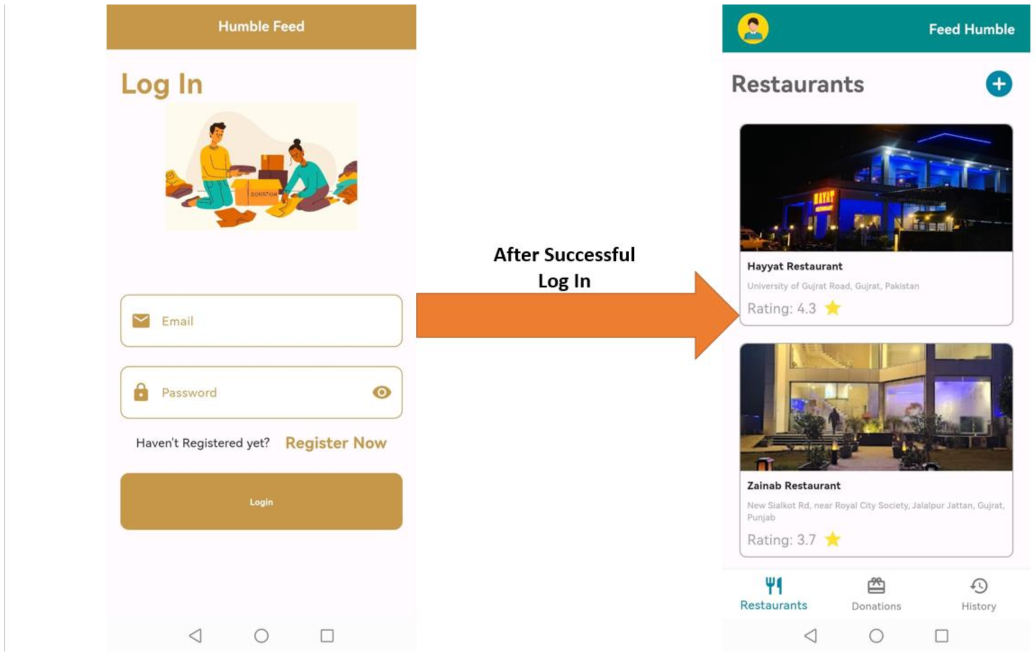
HOW?

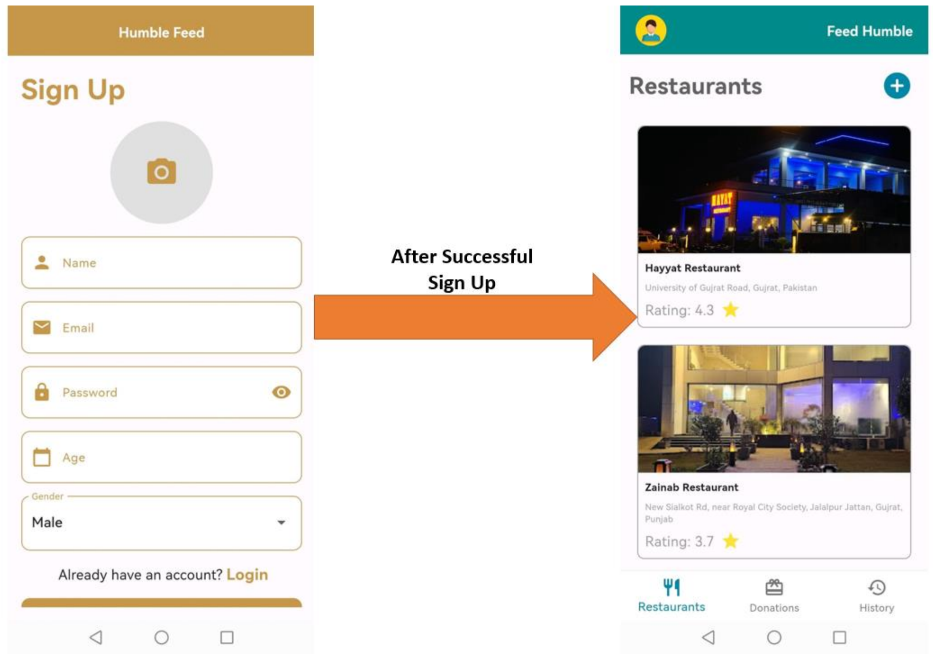
- Use a grid that puts the graphic representation above and the verbal description below
- Begin with loose thumbnail sketches and drawing for early design concepts. Refine with tighter drawings and screen designs for presentation and testing.
- Describe the interaction details and emotional responses verbally when no visual representation is effective
- Keep the medium loose and flexible in the conceptual design phase

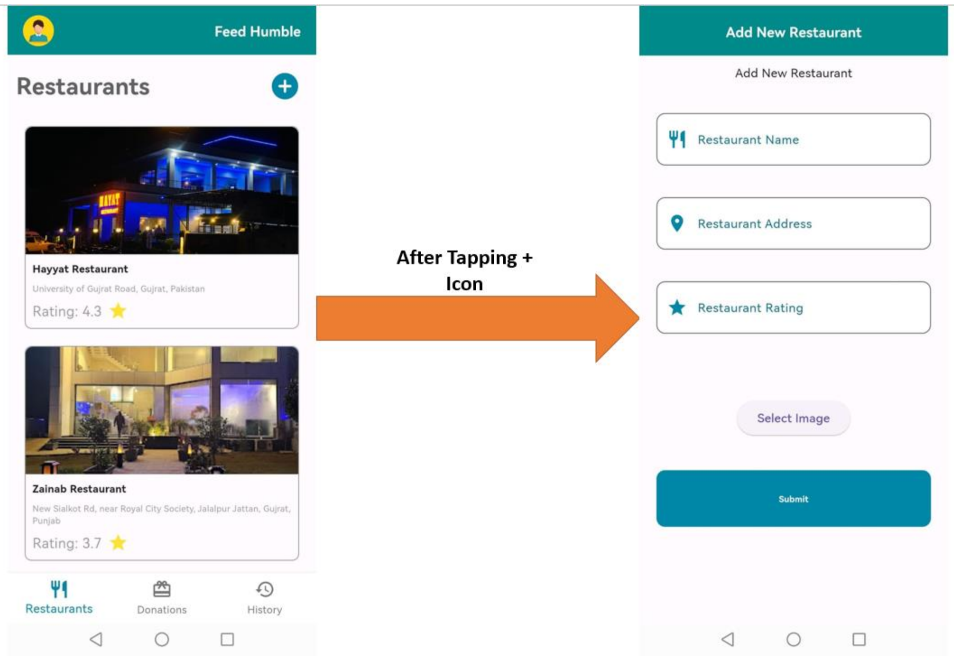
These are the detailed screens, which pictorially represent the complete view of the screens. There would be symbols representing the different elements of the screens and in the end an index that would detail the symbols. Sample is given on the next page.

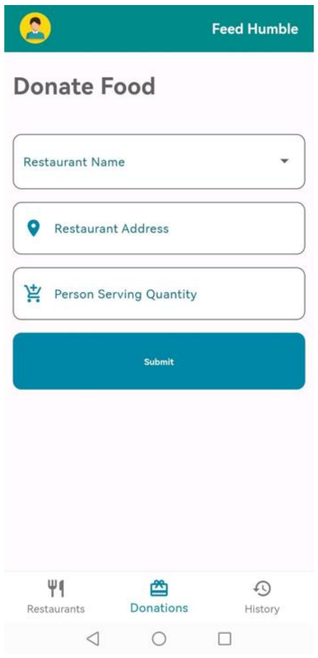
4.4. Navigational maps:

The next step is of navigational maps. In these maps, the storyboards are used as an input. The different display buttons or action buttons show the navigation from one screen to the other. In other words when one action button is pressed it would lead to other screens. This path and navigation would be shown.

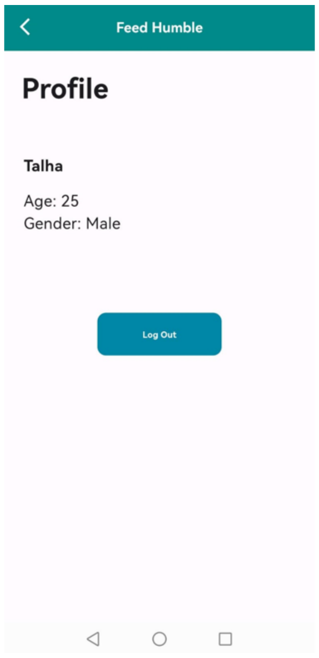



Humble Feed
Sign Up
Name
Email
Password
Age
Gender
Male
Already have an account? Login
After Successful Sign Up
Feed Humble
Restaurants
Hayyat Restaurant
University of Gujrat Road, Gujrat, Pakistan
Rating: 4.3
Zainab Restaurant
New Sialkot Rd, near Royal City Society, Jalalpur Jattan, Gujrat, Punjab
Rating: 3.7
Restaurants
Donations
History


Feed Humble
Restaurants
Hayyat Restaurant
University of Gujrat Road, Gujrat, Pakistan
Rating: 4.3
Zainab Restaurant
New Sialkot Rd, near Royal City Society, Jalalpur Jattan, Gujrat, Punjab
Rating: 3.7
Restaurants
Donations
History
After Tapping + Icon
Add New Restaurant
Add New Restaurant
Restaurant Name
Restaurant Address
Restaurant Rating
Select Image
Submit



To add Food to Donate



Profile View

Chapter 5: Software Testing

5.1 Introduction:

This deliverable is based on the IEEE standard of software testing i.e. IEEE SOFTWARE TEST DOCUMENTATION Std 829-1998. This standard describes a set of basic test documents that are associated with the dynamic aspects of software testing (i.e., the execution of procedures and code). The standard defines the purpose, outline, and content of each basic document. While the documents described in the standard focus on dynamic testing, several of them may be applicable to other testing activities (e.g., the test plan and test incident report may be used for design and code reviews). This standard may be applied to commercial, scientific, or military software that runs on any digital computer. Applicability is not restricted by the size, complexity, or criticality of the software. However, the standard does not specify any class of software to which it must be applied. The standard addresses the documentation of both initial development testing and the testing of subsequent software releases. For a particular software release, it may be applied to all phases of testing from module testing through user acceptance. However, since all of the basic test documents may not be useful in each test phase, the particular documents to be used in a phase are not specified. Each organization using the standard will need to specify the classes of software to which it applies and the specific documents required for a particular test phase.

The standard does not call for specific testing methodologies, approaches, techniques, facilities, or tools, and does not specify the documentation of their use. Additional test documentation may be required (e.g., code inspection checklists and reports). The standard also does not imply or impose specific methodologies for documentation control, configuration management, or quality assurance. Additional documentation (e.g., a quality assurance plan) may be needed depending on the particular methodologies used.

Following are standard artifacts, which must be included in this deliverable:

1. Test Plan
2. Test Design Specification
3. Test Case Specification
4. Test Procedure Specification
5. Test Item Transmittal Report
6. Test Log
7. Test Incident Report
8. Test Summary Report

5.2. Test plan

5.2.1. Purpose

In this test plan, we will check scope, approach, resources, and schedule of the testing activities for our Feed Humble application.

5.2.2. Outline

A test plan shall have the following structure:

a. Test plan identifier
b. Introduction
c. Test items
d. Features to be tested
e. Features not to be tested
f. Approach
g. Item pass/fail criteria
h. Suspension criteria and resumption requirements
i. Test deliverables
j. Testing tasks
k. Environmental needs
l. Responsibilities
m. Staffing and training needs
n. Schedule
o. Risks and contingencies
p. Approvals

Details on the content of each section are contained in the following sub-clauses.

5.2.2.1. Test plan identifier

The Test plan identifier for the Feed Humble Application is **FDAP-TP-2024-V1**

5.2.2.2. Introduction

Summarize the software items and software features to be tested. The need for each item and its history may be included. References to the following documents, when they exist, are required in the highest-level test plan:

a. Project authorization;
b. Project plan;
c. Quality assurance plan;
d. Configuration management plan;
e. Relevant policies;
f. Relevant standards.

Project authorization

The Project was authorized and approved in the context of being created for the Final Year Project.

a) **Project Plan**

The Goal of this project was to be created for the FYP and also to be used in the future.

b) **Quality assurance plan**

By adhering to industry best practices, and continuously monitoring performance, we aim to deliver a robust and seamless application.

c) **Configuration management plan**

We have tried level best to ensure that app's development and testing environments are consistent, controlled, and well-documented throughout the project lifecycle.

d) **Relevant policies**

We have applied relevant policies, best practices, and standards to ensure that our project aligns with established industry norms and ethical considerations.

e) **Relevant standards**

We have applied standards to ensure that our project follows established best practices and meets quality expectations.

5.2.2.3. Test items

By identifying the test items, we ensure a systematic and thorough assessment of the app's features and capabilities.

- User
- Place
- Category
- Food
- Consumers
- Available Food
- Donated Food
- Reports

5.2.2.4. Features to be tested

- Registration
- Login
- Dashboard
- Available Food
- Restaurants
- History
- Database

5.2.2.5. Features not to be tested

- Third Party Services
- Backward Compatibility
- Load Testing for Extreme Conditions
- Connectivity Failures

5.2.2.6. Approach

Our approach for testing features of this app is different for every feature. In Login & Registration we linked it to the Backend and if user enters incorrect data while login or doesn’t provides full information to create an account for registration the app will prompt him to enter full details and when login in to the app if he/she enters wrong info a message from backend will be displayed on the app that he/she has entered wrong details. While browsing the navigations we did all kinds of testing to ensure it doesn’t produce any error and wherever it did we handled the error to make the overall user experience better. Firebase was the best option to build the backend and it reduced our testing work as it handles it pretty much itself.

5.2.2.7. Item pass/fail criteria

Item	Passed	Failed
Registration	✓	
Login	✓	
Dashboard	✓	
Restaurants	✓	
Food	✓	
History	✓	
Donated Food	✓	
Reports	✓	
Database	✓	
Third Party Services		✗
Backward Compatibility		✗
Load Testing		✗
Connectivity Failures		✗

5.2.2.8. Suspension criteria and resumption requirements

- **Suspension Criteria**

➢ We suspended Testing activities under the following circumstances:
  - ✓ If critical defects are identified that significantly impact the functionality or usability of the app.
  - ✓ If external factors, such as network outages or system downtime, hinder the ability to conduct testing.

If there are significant changes to the app's scope, architecture, or requirements.

- **Resumption Requirements:**

➢ We resumed testing activities under the following conditions:
  - ✓ Once critical defects are addressed and verified as resolved, testing will be resumed.

When essential testing resources become available, activities will be resumed.

5.2.2.9. Test deliverables

Test Deliverables includes:

a. **Test plan:**
The test plan will detail our testing strategy for the Feed Humble App including objectives, scope, testing methodologies, and resource allocation. It will serve as a roadmap that guides our testing activities throughout the project.

b. **Test design specifications:**
Test design specifications provide a clear roadmap for executing tests and ensure that testing is comprehensive, repeatable, and aligned with project requirements.

c. **Test case specifications:**
We will document individual test cases and scenarios that cover various aspects of the app, such as user interactions. Each test case will outline inputs, expected outcomes, and the steps to replicate user actions.

d. **Test procedure specifications:**
Test procedure specifications provide step-by-step instructions for testers to perform tests accurately and consistently.

e. **Test item transmittal reports:**
These reports serve as formal documentation to communicate the status of test items and their associated materials.

f. **Test logs:**
Test logs serve as a chronological record of test activities, outcomes, and observations.

g. **Test incident reports:**
Test Incident Reports document unexpected occurrences, defects, and anomalies encountered during testing.

h. **Test summary reports:**
Test Summary Reports that will provide an overview of the testing activities, outcomes, and achievements for the Feed Humble App.

5.2.2.10. Testing tasks

- Test Planning
- Requirements Analysis
- Test Case Design
- Test Data Preparation
- Test Environment Setup
- Test Execution

5.2.2.11. Environmental needs

- A Windows or Mac Device with minimum 8GB Ram.
- Flutter SDK & Android SDK
- Java Development Kit
- VS Code or Android Studio
- XAMPP Server
- Gradle
- GIT
- PHP Composer
- Internet Connection
- Web Browser (Chrome, Edge etc.)

5.2.2.12. Responsibilities

Each member is responsible for requirements, designing, preparing documentation, developing and executing the system properly.

5.2.2.13 Staffing and training needs

Each member was able to do adequate staffing and fulfilling training needs.

5.2.2.14. Schedule

Additional test milestones will take more than 15 days.

5.2.2.15. Risks and contingencies

- **Risk Identification:**

✓ The scope of the app's functionalities might expand during testing.
✓ Insufficient staffing or lack of testing resources.
✓ Reliance on external APIs might introduce technical complexities.
✓ Incomplete or unclear requirements could lead to misunderstandings.

- **Contingencies:**

✓ Maintain clear communication with supervisors to manage scope changes
✓ Manage resources efficiently and consider training team members to solve Insufficient staffing
✓ Identify backup solutions to solve technical dependencies.

Engage with supervisors and project team members to clarify requirements

5.2.2.16 Approvals

Approved by: **Mr. Muhammad Jabbar**

Signature: ____________________

5.3. Test design specification

5.3.1. Purpose

The Purpose of this Test design specification is to prescribe scope, approach, resources, and schedule of the testing activities. We will identify the items being tested, the features to be tested, the testing tasks to be performed, the personnel responsible for each task, and the risks associated with this plan.

5.3.2. Outline

A test plan shall have the following structure:

a. Test plan identifier;
b. Introduction;
c. Test items;
d. Features to be tested;
e. Features not to be tested;
f. Approach;
g. Item pass/fail criteria;
h. Suspension criteria and resumption requirements;
i. Test deliverables;
j. Testing tasks;
k. Environmental needs;
l. Responsibilities;
m. Staffing and training needs;
n. Schedule;
o. Risks and contingencies;
p. Approvals.

Details on the content of each section are contained in the following sub-clauses.

5.3.2.1 Test plan identifier

Test plan identifier of this app is **FDAP-TP-2024-V1**.

5.3.2.2. Introduction

A Test Plan Identifier includes:

a. **Project authorization:**
The Project was authorized and approved in the context of being created for the Final Year Project.

b. **Project plan:**
The Goal of this project was to be created for the FYP and also to be used in the future.

c. **Quality assurance plan:**
By adhering to industry best practices, and continuously monitoring performance, we aim to deliver a robust and seamless application.

d. **Configuration management plan:**
We have tried level best to ensure that app's development and testing environments are consistent, controlled, and well-documented throughout the project lifecycle.

e. **Relevant policies:**
We have applied relevant policies, best practices, and standards to ensure that our project aligns with established industry norms and ethical considerations.

f. **Relevant standards.**
We have applied standards to ensure that our project follows established best practices and meets quality expectations.

5.3.2.3. Test items

By identifying the test items, we ensure a systematic and thorough assessment of the app's features and capabilities.

- User
- Restaurants
- Category
- Available Food
- History
- Database
- Reports

5.3.2.4. Features to be tested

- Registration
- Login
- Dashboard
- Restaurants
- Available Food
- History
- Database

5.3.2.5. Features not to be tested

- Third Party Services
- Backward Compatibility
- Load Testing for Extreme Conditions
- Connectivity Failures

5.3.2.6. Approach

Our approach for testing features of this app is different for every feature. In Login & Registration we linked it to the Backend and if user enters incorrect data while login or doesn’t provides full information to create an account for registration the app will prompt him to enter full details and when login in to the app if he/she enters wrong info a message from backend will be displayed on the app that he/she has entered wrong details. While browsing the navigations we did all kinds of testing to ensure it doesn’t produce any error and wherever it did we handled the error to make the overall user experience better. Firebase was the best option to build the admin panel and it reduced our testing work as it handles it pretty much itself.

5.3.2.7. Item pass/fail criteria

Item	Passed	Failed
Registration	✓	
Login	✓	
Dashboard	✓	
Restaurants	✓	
Food	✓	
History	✓	
Donated Food	✓	
Reports	✓	
Database	✓	
Third Party Services		✗
Backward Compatibility		✗
Load Testing		✗
Connectivity Failures		✗

5.3.2.8. Suspension criteria and resumption requirements

- **Suspension Criteria**

- ➢ We suspended Testing activities under the following circumstances:
  - ✓ If critical defects are identified that significantly impact the functionality or usability of the app.
  - ✓ If external factors, such as network outages or system downtime, hinder the ability to conduct testing.

If there are significant changes to the app's scope, architecture, or requirements.

- **Resumption Requirements:**

- ➢ We resumed testing activities under the following conditions:
  - ✓ Once critical defects are addressed and verified as resolved, testing will be resumed.

When essential testing resources become available, activities will be resumed.

5.3.2.9. Test deliverables

Test Deliverables includes:

i. **Test plan:**
The test plan will detail our testing strategy for the Feed Humble App including objectives, scope, testing methodologies, and resource allocation. It will serve as a roadmap that guides our testing activities throughout the project.

j. **Test design specifications:**
Test design specifications provide a clear roadmap for executing tests and ensure that testing is comprehensive, repeatable, and aligned with project requirements.

k. **Test case specifications:**
We will document individual test cases and scenarios that cover various aspects of the app, such as user interactions. Each test case will outline inputs, expected outcomes, and the steps to replicate user actions.

l. **Test procedure specifications:**
Test procedure specifications provide step-by-step instructions for testers to perform tests accurately and consistently.

m. **Test item transmittal reports:**
These reports serve as formal documentation to communicate the status of test items and their associated materials.

n. **Test logs:**
Test logs serve as a chronological record of test activities, outcomes, and observations.

o. **Test incident reports:**
Test Incident Reports document unexpected occurrences, defects, and anomalies encountered during testing.

p. **Test summary reports:**
Test Summary Reports that will provide an overview of the testing activities, outcomes, and achievements for the Feed Humble App.

5.2.2.10. Testing tasks

- Test Planning
- Requirements Analysis
- Test Case Design
- Test Data Preparation
- Test Environment Setup
- Test Execution

5.3.2.11. Environmental needs

- A Windows or Mac Device with minimum 8GB Ram.
- Flutter SDK & Android SDK
- Java Development Kit
- VS Code or Android Studio
- XAMPP Server
- Gradle
- GIT
- PHP Composer
- Internet Connection
- Web Browser (Chrome, Edge etc.)

5.3.2.12. Responsibilities

Each member is responsible for requirements, designing, preparing documentation, developing and executing the system properly.

5.3.2.13 Staffing and training needs

Each member was able to do adequate staffing and fulfilling training needs.

5.3.2.14. Schedule

Additional test milestones will take more than 15 days.

5.3.2.15. Risks and contingencies

Risk Identification:

- The scope of the app's functionalities might expand during testing.
- Insufficient staffing or lack of testing resources.
- Reliance on external APIs might introduce technical complexities.
- Incomplete or unclear requirements could lead to misunderstandings.

Contingencies:

- Maintain clear communication with supervisors to manage scope changes
- Manage resources efficiently and consider training team members to solve Insufficient staffing
- Identify backup solutions to solve technical dependencies.

Engage with supervisors and project team members to clarify requirements

5.3.2.16 Approvals

Approved by: **Mr. Muhammad Jabbar**

Signature: ____________________

5.4. Test Case Specification

5.4.1. Purpose

The Purpose of this Test design specification is to prescribe scope, approach, resources, and schedule of the testing activities. We will identify the items being tested, the features to be tested, the testing tasks to be performed, the personnel responsible for each task, and the risks associated with this plan.

5.4.2. Outline

Our test plan includes the following outline.

a. Test plan identifier;
b. Introduction;
c. Test items;
d. Features to be tested;
e. Features not to be tested;
f. Approach;
g. Item pass/fail criteria;
h. Suspension criteria and resumption requirements;
i. Test deliverables;
j. Testing tasks;
k. Environmental needs;
l. Responsibilities;
m. Staffing and training needs;
n. Schedule;
o. Risks and contingencies;
p. Approvals.

Details on the content of each section are contained in the following sub-clauses.

5.4.2.1 Test plan identifier

Test plan identifier of this app is **FDAP-TP-2024-V1**

5.4.2.2 Test items

By identifying the test items, we ensure a systematic and thorough assessment of the app's features and capabilities.

- User
- Restaurants
- Category
- Available Food
- History
- Database
- Reports

5.4.2.3. Input specifications

Sr#	Test Items Input	Test case identifier
1	User enters valid details (email, name, password, phone)	TC1
2	User enter valid details (username, password)	TC2
3	User check for available food.	TC3
4	User receive food.	TC4
5	Administrator enters valid details (username, password)	TC5
6	User's information	TC6
7	App interacts with the database to place food.	TC7

5.4.2.4. Output specifications

Sr#	Test Items Output	Test case identifier
1	User registration is successful.	TC1
2	User login is successful.	TC2
3	Available Food display	TC3
4	Restaurants displayed	TC4
5	Administrator successfully logs in and is redirected to the admin panel dashboard.	TC5
6	Add Food to donate	TC6
7	Food details are successfully stored in the database.	TC7

5.4.2.5. Environmental needs

5.4.2.5.1. Hardware

Sr#	Hardware Required	Test case identifier
1	Computer or laptop or mobile	TC1
2	Computer or laptop or mobile	TC2
3	Computer or laptop or mobile	TC3
4	Computer or laptop or mobile	TC4
5	Computer or laptop or mobile	TC5
6	Computer or laptop or mobile	TC6
7	Computer or laptop or mobile	TC7

5.4.2.5.2. Software

- A Windows or Mac Device with minimum 8GB Ram.
- Flutter SDK & Android SDK
- Java Development Kit
- VS Code or Android Studio
- XAMPP Server
- Gradle
- GIT
- PHP Composer
- Internet Connection
- Web Browser (Chrome, Edge etc.)

5.4.2.5.3. Other

There were no requirements for unique facility needs or specially trained personnel.

5.4.2.6. Special procedural requirements
There were no special constraints on the test procedures that execute this test case.

5.4.2.7. Inter case dependencies

Sr#	Test case identifier	Test case Dependency
1	TC1	
2	TC2	TC1
3	TC3	TC2
4	TC4	TC3
5	TC5	
6	TC6	
7	TC7	

5.5. Test procedure specification

5.5.1. Purpose

The purpose of test procedure specification is to specify the steps for executing a set of test cases.

5.5.2 Outline

A test procedure specification shall have the following structure:

a. Test procedure specification identifier
b. Purpose
c. Special requirements
d. Procedure steps

5.5.2.1. Test procedure specification identifier

The unique identifier assigned to this test procedure specification is TPS-001

5.5.2.2. Purpose

The purpose of the Test Procedure Specification in our app is to provide a clear and structured guide. It ensures consistent and thorough testing of the app's functionalities, leading to higher quality and reliability.

5.5.2.3. Special requirements

- A Windows or Mac Device with minimum 8GB Ram.
- Flutter SDK & Android SDK
- Java Development Kit
- VS Code or Android Studio
- XAMPP Server
- Gradle
- GIT
- PHP Composer
- Internet Connection
- Web Browser (Chrome, Edge etc.)

5.5.2.4. Procedure steps

Include the steps in 8.5.2.4.1. through 8.5.2.4.10 as applicable.

5.5.2.4.1. Log

The log section of our app provides a record of what actions were taken, what outcomes were observed, and any issues or observations made during the test.

5.5.2.4.2. Set up

The set of actions necessary to prepare for execution of the procedure were:

- A Laptop with Flutter installed.
- An editor like Android Studio or VS Code
- A Physical or Virtual Emulator

5.5.2.4.3. Start

- On the Application Side, Open the editor and navigate to Flutter project
- Open a terminal and do ‘Flutter run’ this will start the compilation of the project.
- After Successful running of the App now you can begin Testing

5.5.2.4.4. Proceed

While running the app make sure to have an internet connection otherwise the app will not receive the data from Database.

5.5.2.4.5. Measure

Test Measurements are done through Logs e.g. Print Statements. These Statements help us to identify the issue and resolve it.

5.5.2.4.6. Shut down

To shut down the app we would have to kill the emulator and also run ‘Q’ short for quit to shut down the app.

5.5.2.4.7. Restart

To restart we would run ‘Shift+R’ which will restart the compilation.

5.5.2.4.8. Stop

To stop it we would run ‘q’ and close the emulator also we’ll stop the development server running at localhost.

5.5.2.4.9. Wrap up

To Wrap up, we’ll quit the compilation, stop the development server and close everything.

5.5.2.4..10. Contingencies

- If during execution, the app doesn't get data from database we'll use Log or Print Statement to find the issue and resolve it.
- If any API doesn't work, we'll use Post Man to test it.
- If the app crashes, we'll restart it.

5.6. Test item transmittal report

5.6.1. Purpose

The purpose of test items being transmitted includes the person responsible for each item, its physical location, and its status.

5.6.2. Outline

A test item transmittal report shall have the following structure:

a. Transmittal report identifier
b. Transmitted items
c. Location
d. Status
e. Approvals

5.6.2.1. Transmittal report identifier

The unique identifier assigned to test item transmittal report is TR-001.

5.6.2.2. Transmitted items

There weren't any physical test items that were transmitted for Testing. All the testing happened in this device with all the necessary tools like Postman. Postman software helped us to see whether the API was working or not. Other type of code and user interface testing was done with Print Statements.

5.6.2.3. Location

There weren't any physical test items that were transmitted anywhere for Testing.

5.6.2.4. Status

All test items were inside this windows device and there weren't any physical items that were transmitted anywhere.

5.6.2.5. Approvals

Approved by: **Mr. Muhammad Jabbar**

Signature: ____________________

5.7. Test log

5.7.1. Purpose

The purpose of this Test Log is to provide a chronological record of relevant details about the execution of tests.

5.7.2. Outline

A test log shall have the following structure:

a. Test log identifier;
b. Description;
c. Activity and event entries.

5.7.2.1. Test log identifier

Specify the unique identifier assigned to this test log.

5.7.2.2. Description

Test Items	Test Case ID	Software	Hardware
Registration	TC1	Flutter, VSCode or Android Studio, Firebase	Core i5 Processor, 8GB Ram
Login	TC2	Flutter, VSCode or Android Studio, Firebase	Core i5 Processor, 8GB Ram
Available Food	TC3	Flutter, VSCode or Android Studio, Firebase	Core i5 Processor, 8GB Ram
Donation History	TC4	Flutter, VSCode or Android Studio, Firebase	Core i5 Processor, 8GB Ram
Restaurants	TC5	Flutter, VSCode or Android Studio, Firebase	Core i5 Processor, 8GB Ram
Donated Food	TC6	Flutter, VSCode or Android Studio, Firebase	Core i5 Processor, 8GB Ram
Database	TC7	Flutter, VSCode or Android Studio, Firebase	Core i5 Processor, 8GB Ram

5.7.2.3. Activity and event entries

Test Items	Test Case ID	Start Date	End Date
Registration	TC1	04-07-2024	04-07-2024
Login	TC2	04-07-2024	04-07-2024
Available Food	TC3	05-07-2024	05-07-2024
Donation History	TC4	05-07-2024	06-07-2024
Restaurants	TC5	06-07-2024	08-07-2024
Donated Food	TC6	08-07-2024	10-07-2024
Database	TC7	10-07-2024	11-07-2024

5.7.2.3.1. Execution description

Sr#	Test case identifier	Test case Dependency
1	TC1	
2	TC2	TC1
3	TC3	TC2
4	TC4	TC3
5	TC5	
6	TC6	
7	TC7	

5.7.2.3.2. Procedure results

For each execution, record the visually observable results (e.g., error messages generated, aborts, and requests for operator action). Also record the location of any output (e.g., reel number). Record the successful or unsuccessful execution of the test.

5.7.2.3.3. Environmental information

- A Windows or Mac Device with minimum 8GB Ram.
- Flutter SDK & Android SDK
- Java Development Kit
- VS Code or Android Studio
- XAMPP Server
- Gradle
- GIT
- PHP Composer
- Internet Connection
- Web Browser (Chrome, Edge etc.)

5.7.2.3.4. Anomalous events

There were errors that were solved on the way but we didn’t encounter any major anomalous event.

5.7.2.3.5. Incident report identifiers

We recorded the identifier of each test incident report.

5.8. Test incident report

5.8.1. Purpose

The purpose of Test Incident Report is to document any event that occurs during the testing process that requires investigation.

5.8.2. Outline

A test incident report shall have the following structure:

a. Test incident report identifier
b. Summary
c. Incident description
d. Impact

Details on the content of each section are contained in the following sub clauses.

5.8.2.1. Test incident report identifier

The unique identifier assigned to this test is TIC-001

5.8.2.2. Summary

There were errors that were solved on the way but we didn’t encounter any major anomalous event.

5.9. Test summary report

5.9.1. Purpose

The purpose of this Test summary report to summarize the results of the designated testing activities and to provide evaluations based on these results.

5.9.2. Outline

A test summary report shall have the following structure:

a. Test summary report identifier
b. Summary
c. Variances
d. Comprehensive assessment
e. Summary of results
f. Evaluation
g. Summary of activities
h. Approvals

Details on the content of each section are contained in the following sub clauses.

5.9.2.1. Test summary report identifier

The unique identifier assigned to this test summary report is TSR-001

5.9.2.2. Summary

All modules and functionalities of the Car Parking App are tested. All test items were tested in Windows Operating System with necessary software e.g. Flutter SDK, VS Code or Android Studio, Firebase.

5.9.2.3. Variances

There are no variances in the test items from their design specifications.

5.9.2.4. Comprehensiveness assessment

There were errors that were solved on the way but we didn't encounter any major anomalous event.

5.9.2.5. Summary of results

Every test item is working fine and every module is performing its required function.

5.9.2.6. Evaluation

Test Items	Test Case ID	Passed	Failed
Registration	TC1	✓	
Login	TC2	✓	
Available Food	TC3	✓	
Donation History	TC4	✓	
Restaurants	TC5	✓	
Donated Food	TC6	✓	
Database	TC7	✓	

5.9.2.7. Summary of activities

Test Items	Test Case ID	Total Staff	Total Time
Registration	TC1	2	1 Day
Login	TC2	2	1 Day
Available Food	TC3	2	1 Day
Donation History	TC4	2	2 Days
Restaurants	TC5	2	2 Days
Donated Food	TC6	2	7 Days
Database	TC7	2	1 Days

5.9.2.8. Approvals

Approved by: **Mr. Muhammad Jabbar**

Signature: ______________________

Chapter 7: Results

7.1 Result

The **Humble Feed** project successfully achieved its primary goals by providing a user-friendly platform that facilitates food donations from restaurants and individuals to those in need. Here are the key results observed from the development and initial deployment of the app:

7.1.1 User Authentication and Profile Management:

- Implemented a secure and efficient user authentication system using Firebase.
- Users can easily sign up, log in, and manage their profiles.

7.1.2 Restaurant Listings and Food Availability:

- Displayed a comprehensive list of participating restaurants with detailed information, including name, address, and rating.
- Enabled users to view food items available for donation, with details about the quantity and status of the food.

7.1.3 Donation Process:

- Simplified the process for donors to submit information about food donations, including the restaurant name, address, and quantity of food.
- Successfully tracked donations and updated the availability status of food in real-time.

7.1.4 Donation and Receipt History:

- Developed a history feature that allows users to view a detailed record of their donations and food received.
- Enhanced transparency and accountability by maintaining an accurate history of all transactions.

7.1.5 User Interface and Navigation:

- Designed an intuitive and easy-to-navigate interface using Flutter, with three main pages accessible through a navigation bar: Restaurants (home page), Donation, and History.
- Ensured a seamless user experience with smooth transitions between different sections of the app.

7.1.6 Firebase Integration:

- Leveraged Firebase for real-time database management, ensuring fast and reliable data storage and retrieval.
- Used Firebase authentication for secure user management.

7.2 User Feedback

The initial feedback from users has been positive, highlighting the app's ease of use and effectiveness in facilitating food donations. Key points from user feedback include:

7.2.1 Ease of Use:

- Users found the app easy to navigate and appreciated the simple and straightforward design.

7.2.2 Effectiveness:

- Users noted that the app effectively connected them with restaurants willing to donate food, making it easier to provide help to those in need.

7.2.3 Transparency:

- The history feature was particularly appreciated for its role in maintaining transparency and accountability in the donation process.

7.3 Future Enhancements Based on Results

Based on the initial results and user feedback, several enhancements have been identified to further improve the "Humble Feed" app:

7.3.1 Notification System:

- ➢ Implement real-time notifications to keep users informed about new donations, urgent needs, and updates on their contributions.

7.3.2 Expanded Partnerships:

- ➢ Increase collaboration with more restaurants, grocery stores, and non-profit organizations to expand the reach and impact of the app.

7.3.3 User Engagement and Incentives:

- ➢ Introduce features to recognize and reward active donors, such as a points system or badges, to encourage continued participation.

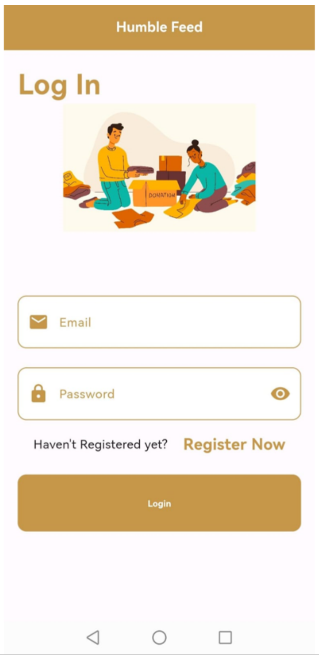
7.3.4 Enhanced Data Analytics:

- ➢ Implement analytics to track and analyze donation trends, user activity, and high-demand areas to better serve the community.

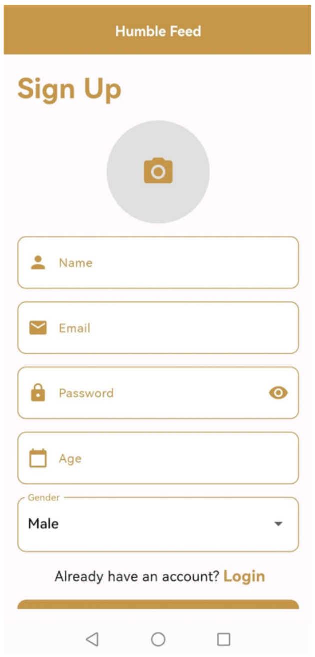
By addressing these areas, "Humble Feed" can continue to evolve and provide even greater support to those in need, ultimately contributing to a more sustainable and connected community.

Chapter 8: User Manual

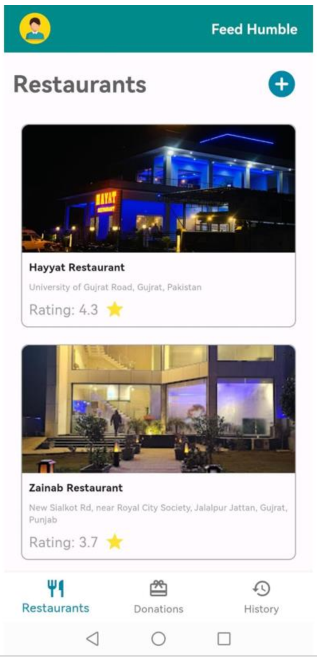
Log In View



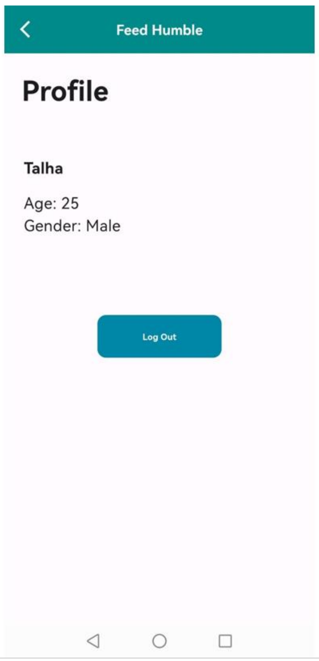
Sign Up View



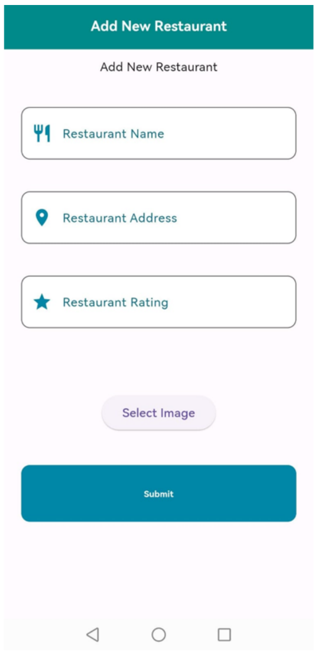
Dashboard View


Feed Humble
Restaurants
Hayyat Restaurant
University of Gujrat Road, Gujrat, Pakistan
Rating: 4.3
Zainab Restaurant
New Sialkot Rd, near Royal City Society, Jalalpur Jattan, Gujrat, Punjab
Rating: 3.7
Restaurants
Donations
History

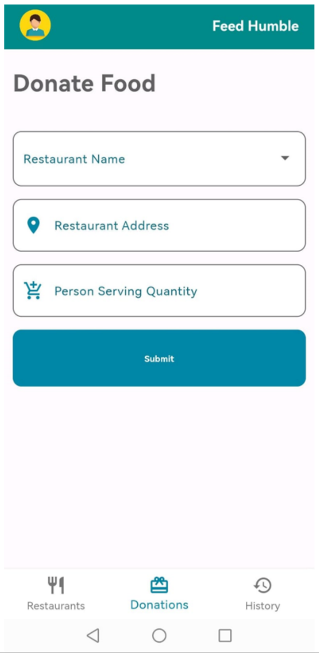
Profile View



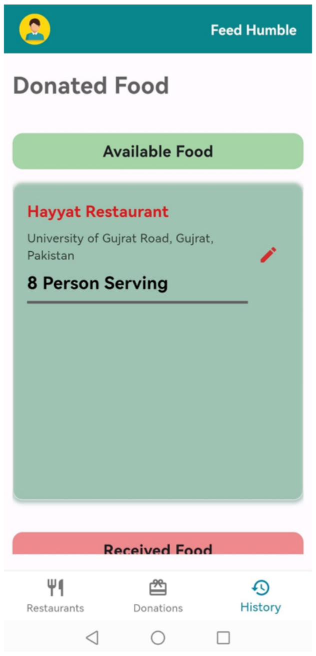
Add New View



Donate Food View



History View


Feed Humble

Donated Food

Available Food

Hayyat Restaurant

University of Gujrat Road, Gujrat, Pakistan

8 Person Serving

Received Food

Chapter 9: Conclusion and Future Work

The "Humble Feed" project aims to provide a platform that connects restaurants and donors with people in need, facilitating the donation of surplus food. Using Flutter for the front-end and Firebase as the backend, the app ensures a seamless and user-friendly experience. Key features include user authentication, restaurant listings, food donation tracking, and a history of donations received and given.

The application successfully addresses several critical aspects:

1) **User Authentication:**

   ✓ Ensures secure access and personalized experiences.

2) **Restaurant Listings:**

   ✓ Provides a comprehensive list of restaurants with detailed information.

3) **Food Donation Tracking:**

   ✓ Allows users to donate and track food donations easily.

4) **History Management:**

   ✓ Maintains a detailed history of donations, enhancing transparency and accountability.

Future Work:

While the "Humble Feed" project has achieved its primary objectives, there are several areas for future improvement and expansion:

1) **Enhanced User Profiles:**

   ✓ Add profile pictures and more detailed information.
   ✓ Implement social login options (e.g., Google, Facebook) for easier access.

2) **Geolocation Services:**

   ✓ Integrate geolocation to help users find the nearest restaurants or donation centers.
   ✓ Provide route and distance information for better planning.

3) **Advanced Analytics:**

- ✓ Implement data analytics to track donation trends, most active users, and high-demand areas.
- ✓ Provide insights and reports to users and administrators.

4) **Multilingual Support:**

- ✓ Offer the app in multiple languages to cater to a broader audience.

5) **Collaborations and Partnerships:**

- ✓ Establish partnerships with more restaurants, grocery stores, and non-profits to expand the reach and impact of the app.
- ✓ Implement a feature for organizations to register and manage their donation activities.

6) **User Feedback and Rating System:**

- ✓ Introduce a rating and feedback system for donors and recipients to ensure quality and trust in the donation process.

7) **Enhanced Security Measures:**

- ✓ Continuously improve security protocols to protect user data and prevent unauthorized access.

8) **Offline Functionality:**

- ✓ Develop offline capabilities to allow users to access certain features without an internet connection.

9) **Mobile Wallet Integration for Rewards:**

- ✓ While maintaining free services for donations, consider integrating a rewards system through mobile wallets for active donors, providing incentives and recognition for their contributions.

By focusing on these areas, "Humble Feed" can evolve into a more robust, user-friendly, and impactful platform, furthering its mission to alleviate hunger and promote food security through technology and community collaboration.