TABLE OF CONTENT

1.0 INTRODUCTION..... 3

1.1 Problem Statement..... 3

1.2 Objectives..... 3

1.3 Scope of the System..... 3

2.0 REQUIREMENT SPECIFICATION..... 4

2.1 Functional Requirements..... 4

2.2 Non-Functional Requirements..... 4

3.0 USE CASE DESIGN..... 5

3.1 Use Case Diagram..... 5

3.2 Use Case Description..... 5

4.0 USER INTERFACE DESIGN..... 6

4.1 Wireframes..... 6

4.2 Navigation Flow..... 6

5.0 SYSTEM DESIGN..... 7

5.1 Class Diagram..... 7

5.2 Object-Oriented Design..... 7

6.0 EVENT HANDLING PLAN..... 8

6.1 List of UI Events..... 8

6.2 Event Flow Example..... 8

7.0 DATA STORAGE STRATEGY..... 9

7.1 Data Structure..... 9

7.2 Validation..... 9

8.0 SYSTEM ARCHITECTURE..... 10

8.1 Architecture Pattern..... 10

8.2 Component Breakdown..... 10

9.0 TOOLS AND PLANNING..... 11

9.1 Tools Used..... 11

9.2 Timeline..... 11

10.0 SUMMARY..... 12

10.1 Team Roles & Contribution..... 12

10.2 Final Thoughts..... 12

1.0 INTRODUCTION

1.1 Problem Statement

With the rapid expansion of the food delivery industry it has placed significant pressure on customer service teams, who must manage a growing volume of service requests daily.The daily challenges faced by these customer service teams often include inaccurate orders, delayed deliveries, payment issues, and account-related problems. By using unmodernised customer support systems it often lacks the responsiveness and intelligence required to handle such inquiries effectively and in real time. Therefore, there is a need for a desktop-based support application powered by AI to provide immediate, accurate assistance and streamline issue tracking and resolution processes.

1.2 Objectives

The food delivery app supports

1.3 Scope of the System

The food delivery support app helps provide customer support when handling the food delivery process. Within its scope, the system has included many important support functionalities such as user registration and login, to ensure that users can access the platform. The users will also be able to create, submit and track their tickets related to their food delivery. The system also provides a FAQ section for customers to find the answer needed to their common queries. A chatbox is also available to assist customers by responding to their questions in real-time. In addition, role- based access control is created to differentiate between customers and admin to ensure that only authorized users can only perform specific tasks given. Unfortunately the system excludes core food delivery operations. It does not cover the actual food ordering process, payment or checkout functionalities or integration with delivery food drivers, all these aspects are out of this scope of this support- focused application.

2.0 REQUIREMENT SPECIFICATION

The requirements of this system are gathered by observing the existing food delivery support systems and team's discussion regarding the needs of admins and customers in a food delivery support app.

2.1 Functional Requirements

ID	Functionality	Description
FR01	Register Account	Allows customers to sign up using their name, email, and password.
FR02	Login	Enables both customers and admins to log in with valid credentials.
FR03	Submit Ticket	Allows customers to report issues by selecting order and describing the problem. They can upload images for proof.
FR04	View Ticket Status	Enables customers to track their submitted ticket's status and admin response.
FR05	Respond to Ticket	Allows admins to read and reply to tickets, and update their status.
FR06	Manage FAQs	Allows admins to add, edit, or delete FAQ entries.
FR07	Access FAQs	Customers can access the frequently asked questions.
FR08	Change Password	Enables customers to change their password.
FR09	Provide Feedback	Lets customers give rating to the support after their issues are resolved.
FR10	Use Chatbot	Customers can interact with chatbot for automatic reply.
FR11	Receive Notifications	Allows customers to receive notifications regarding their ticket status.
FR12	Manage Automated Respond	Allows admin to add, edit, or delete automated replies shown to the customer.

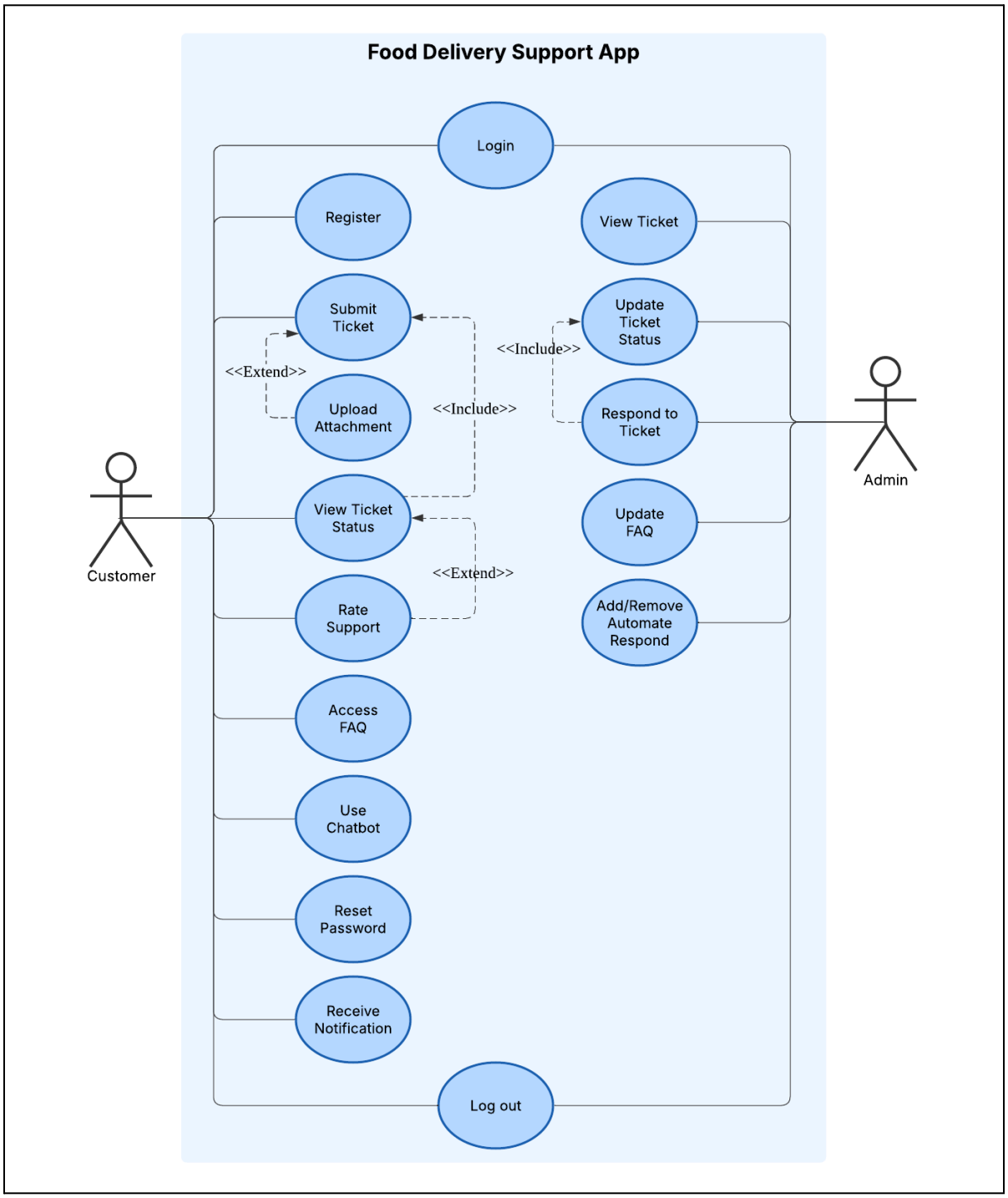
2.2 Non-Functional Requirements

ID	Non Functional	Description
NFR01	Performance	The system should respond to users and the chatbox must respond in real time to support the customer.
NFR02	Security	Users will have to log in before updating or viewing tickets
NFR03	Reliability	The system will ensure data security and users will not lose data during chat sessions or ticket submissions
NFR04	Maintainability	The system should be easy to update and change from time to time
NFR05	Scalability	The app should be able to support large increasing numbers of customers and handle multiple customer chat sessions
NFR06	Availability	The system should be available 24/7 and in case of system downtime it should be restarted and supported
NFR07	Portability	The app should be able to work and perform on all web browsers and mobile devices
NFR08	Usability	Easy access for both admins and customers

3.0 USE CASE DESIGN

This section focuses on how users interact with our Food Delivery Support App. We created a use case diagram to help visualise the system's main functions and show what each type of user can do within the app.

3.1 Use Case Diagram



3.2 Use Case Overview

The table below describes each use case in detail, including the actors involved and its necessary condition.

Use Case ID	Use Case Name	Actor	Description	Precondition	Postcondition
UC01	Register	Customer	Customer registers a new account.	Not logged in	Account is created
UC02	Login	Customer /Admin	User logs in with an email and password.	Account exists	Access granted
UC03	Submit Ticket	Customer	Customer submits a support ticket.	Logged in	Ticket is saved
UC04	Upload Attachment	Customer	Customer attaches a file to support the ticket.	Submitting ticket	Attachment saved
UC05	View Ticket Status	Customer	Customer checks the status of submitted ticket(s).	At least one ticket made	Status is displayed
UC06	Rate Support	Customer	Customers give feedback and rating on the support received.	Ticket resolved	Feedback is saved
UC07	Access FAQ	Customer	Customer browses the list of FAQs.	Logged in	FAQ list is displayed
UC08	Use Chatbot	Customer	Customer interacts with chatbot for help.	FAQ available	Bot returns response
UC09	Reset Password	Customer	Users reset their password via profile management.	Logged in	Password updated
UC10	Receive Notification	Customer	Customers receive system notifications related to ticket updates.	Ticket status updated	Notification received
UC11	View Ticket	Admin	Admin views all submitted support tickets.	Logged in as admin	Ticket list shown
UC12	Update Ticket Status	Admin	Admin updates ticket status (Pending, In Progress, Resolved).	Ticket exists	Status updated

UC13	Respond to Ticket	Admin	Admin writes a response to a customer's ticket.	Ticket exists	Response recorded
UC14	Update FAQ	Admin	Admin updates the list of FAQs.	Logged in as admin	FAQ updated
UC15	Add/Remove Automate Respond	Admin	Admin manages automated replies linked to chatbot.	Logged in as admin	Automated response updated
UC16	Log Out	Customer /Admin	User logs out of the system securely.	Logged in	Session ended

4.0 USER INTERFACE DESIGN

In our app, to give a smooth and enjoyable user experience, we have been meticulous to adhere to the principles of user interface (UI) design. UI design strives to give the look and feel of an application that guides user interaction. We ensured users could easily navigate the system, accomplish work quickly, and work intuitively with an aim of simplicity and usability. We effectively combined stable visual organization, perfect fonts, logical navigation, and quick responsiveness to create an easy-to-work-with space that is totally functional on every system.

4.1 Wireframes

The following wireframes illustrate the initial system's user interface design as a visual representation of the planned layout, navigation structure, and major interactive components. The design sought to create an intuitive, user-centered, and efficient interface that would enable easy interaction by both customer and admin users. All wireframes reflect strict adherence to usability principles, consistent visual information, and clear information hierarchy to enhance overall user experience.

The wireframes include the below basic pages:

- Login page

  This is the primary way users can sign in to their account providing user name/email and password as depicted in Figure.X.

- Sign up

  This page allows for a new user account creation.Both login and sign up pages in Figure.X provide two options upfront: to either log in to an existing account or sign up for one.

- Reset Password

  This page in Figure.X provide the user with account recovery mechanism in case forgetting the password.

- Manage Profile

  The page in Figure.X allows users to manage their personal information such as name, email, password, and other relevant details.

- Logout

  As security and privacy features, our app provide logout option as in Figure.X offering protection against unauthorized access and misuse of accounts.

1. Customer Screens

A customer user has the personal dashboard or home screen upon which the user begins the journey within our app. The dashboard page illustrated in Figure 4.4 serves as a gateway to all the other screens and features. All screens within our app are accessible from the home screen one way or another.

By Figure 4.5, the customer can enter the details of issues and complaints and submit that to our app for processing. After sending an issue, the customer can view the ticket summary as shown in Figure 4.9. Additionally, the customer can monitor and check the status of the issues such as Figure 4.8. Our app informs of significant changes as in the Figure 4.10 page.

For the convenience of our customer, our application provides the FAQs regarding problems and issues that may be encountered at the customer side. Figure 4.6 is the page where answers and questions are presented to the customer. Customer instant support is also provided through a ChatBot as indicated by Figure 4.7.

FAQs and chatbots help to reduce waiting time, enhance customer experience, and make customer support easy.

2. Admin Screens

There is a specific admin dashboard where the admin begins managing the system activity. The dashboard page as seen in Figure 4.12 connects all the other admin screens and incorporates features. All the processes pertaining to ticket administration, FAQs, and customer support can be accessed using this master screen.

By using Figure 4.13, the admin can manage customer tickets by observing the list of tickets posted, filtering them, and selecting specific tickets to perform operations. Upon selecting a ticket, the admin can view its entire detail and perform the updates as shown in Figure 4.14.

The admin can modify the Frequently Asked Questions (FAQs) database from within the application. From Figure 4.15, existing FAQ entries are provided so customers can access up-to-date information. New FAQ entries can also be added using the page of Figure 4.16.

In order to ensure continuous improvement, the admin can review customer feedback and score the support performance using the screen shown in Figure 4.17. This helps them monitor support quality and enhance the customers' experience.

The individual wireframes for each of these pages are as follows.

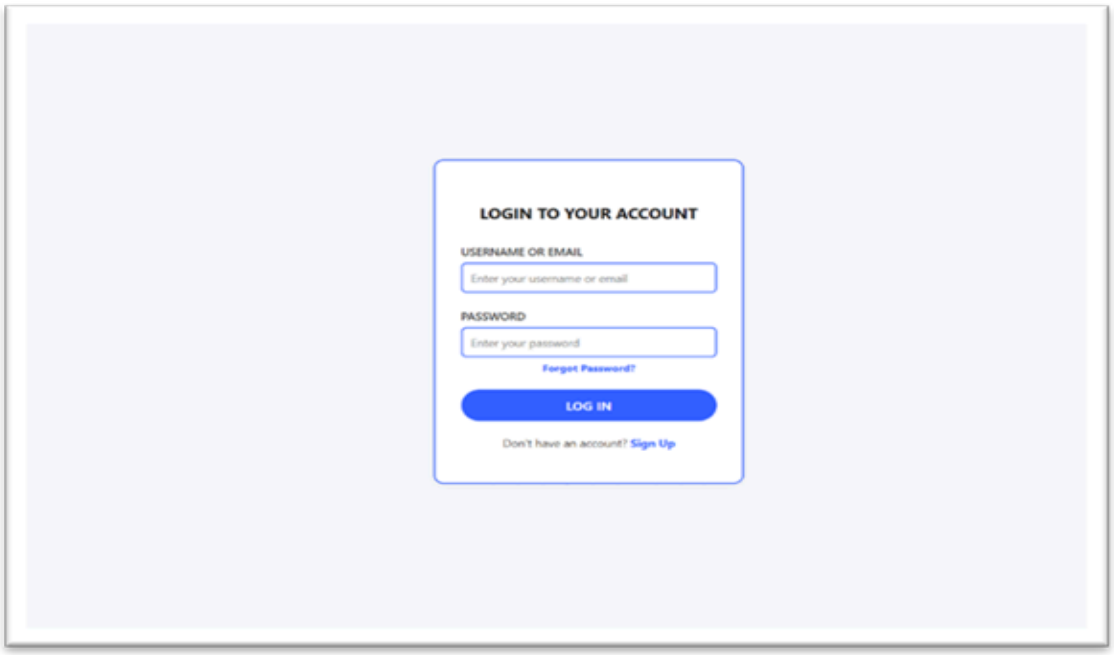


Figure 4.1: Wireframe of Login Page

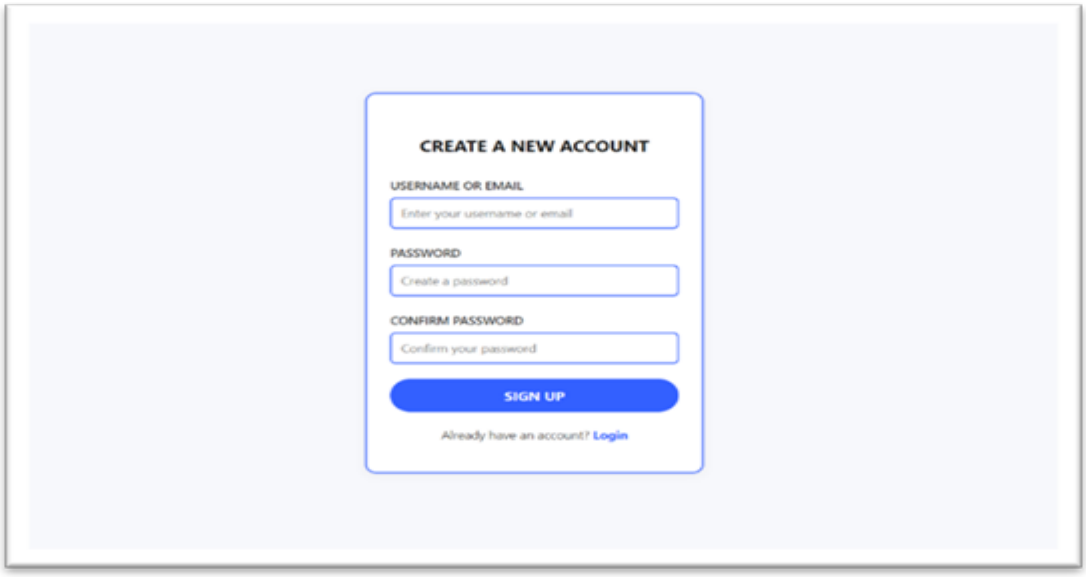


Figure 4.2: Wireframe of Sign Up Page

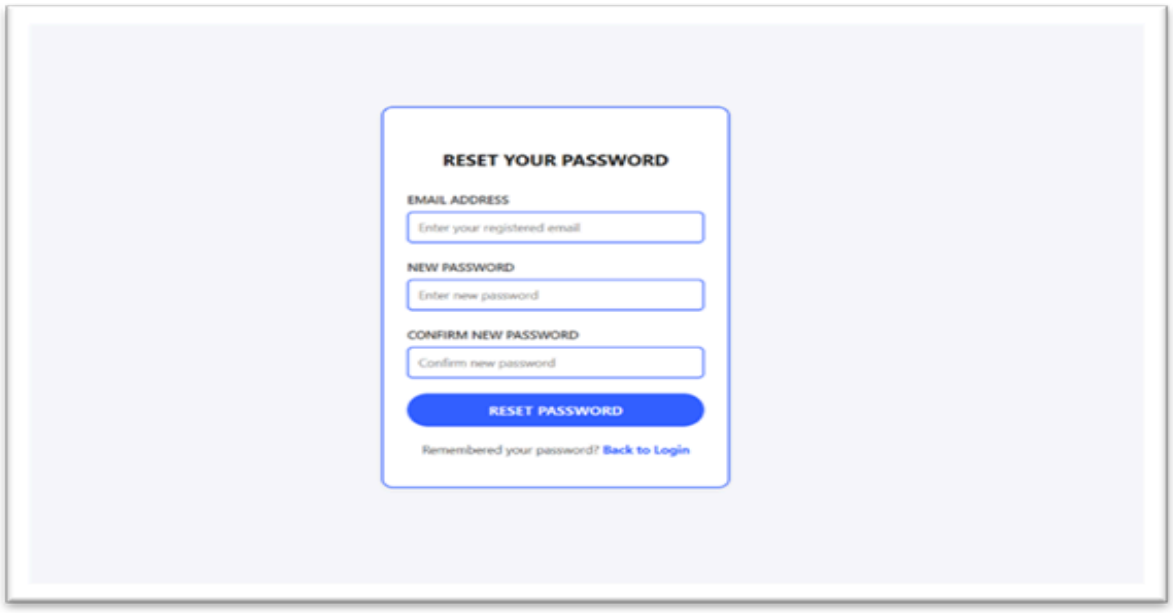


Figure 4.3: Wireframe of Reset Password Page

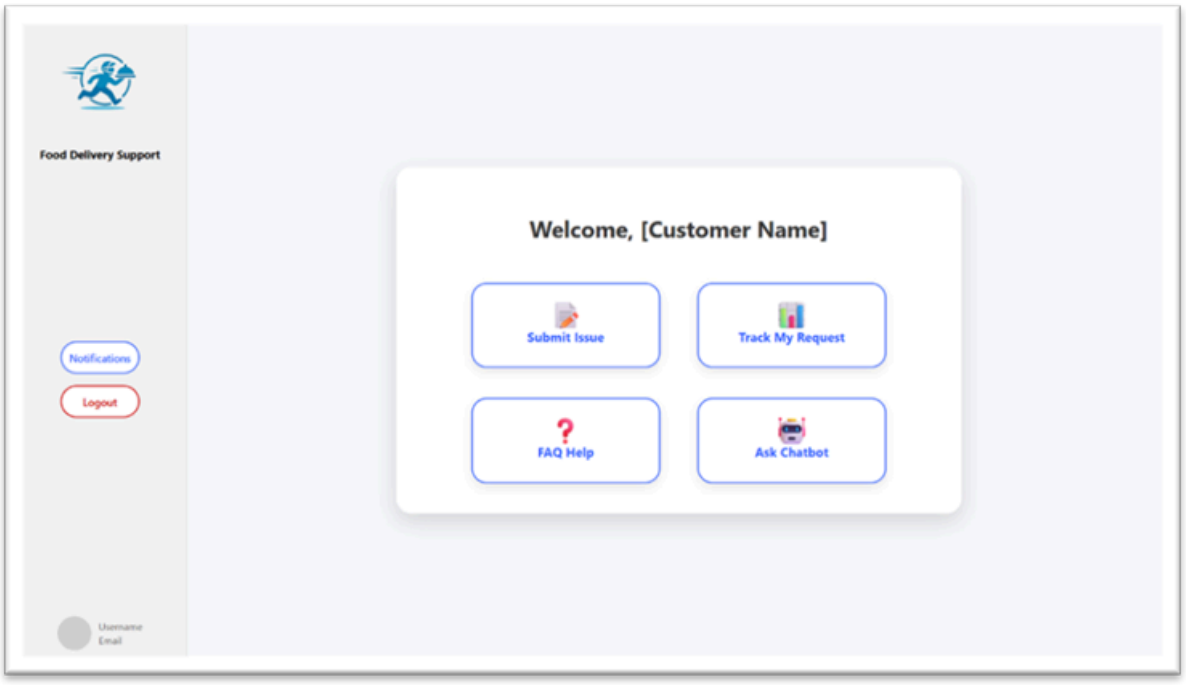


Figure 4.4: Wireframe of Customer Dashboard

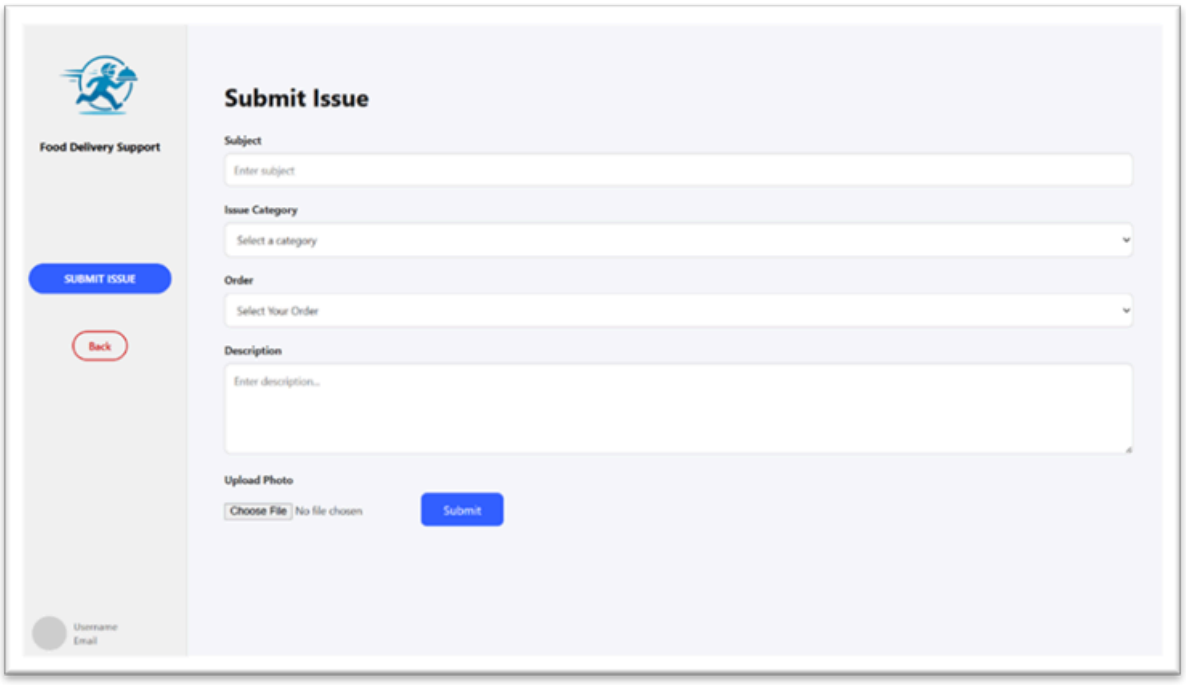


Figure 4.5: Wireframe of Submit Issue Page

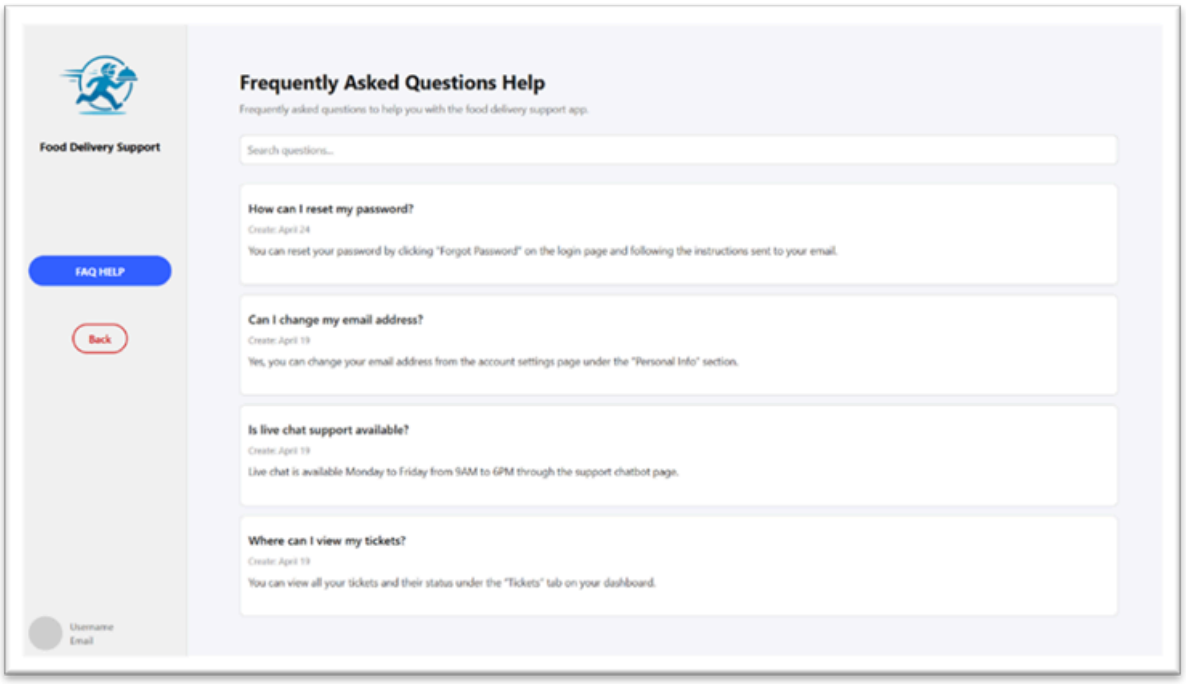


Figure 4.6: Wireframe of FAQ Help Page

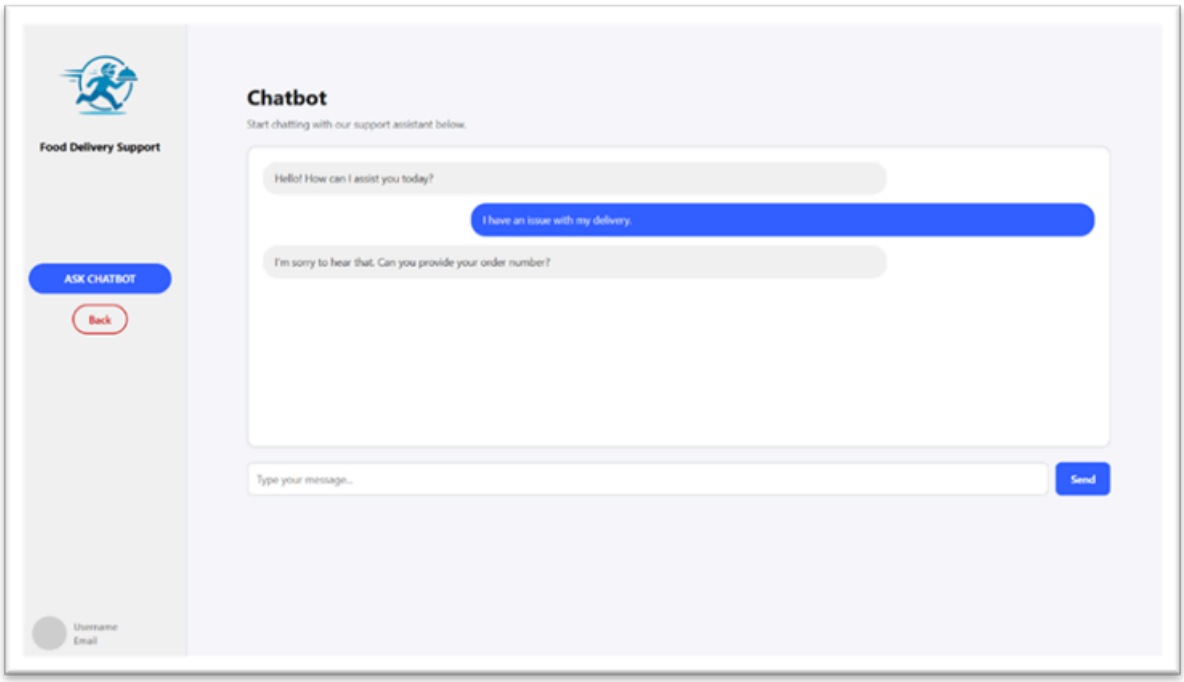


Figure 4.7: Wireframe of Chatbot Page

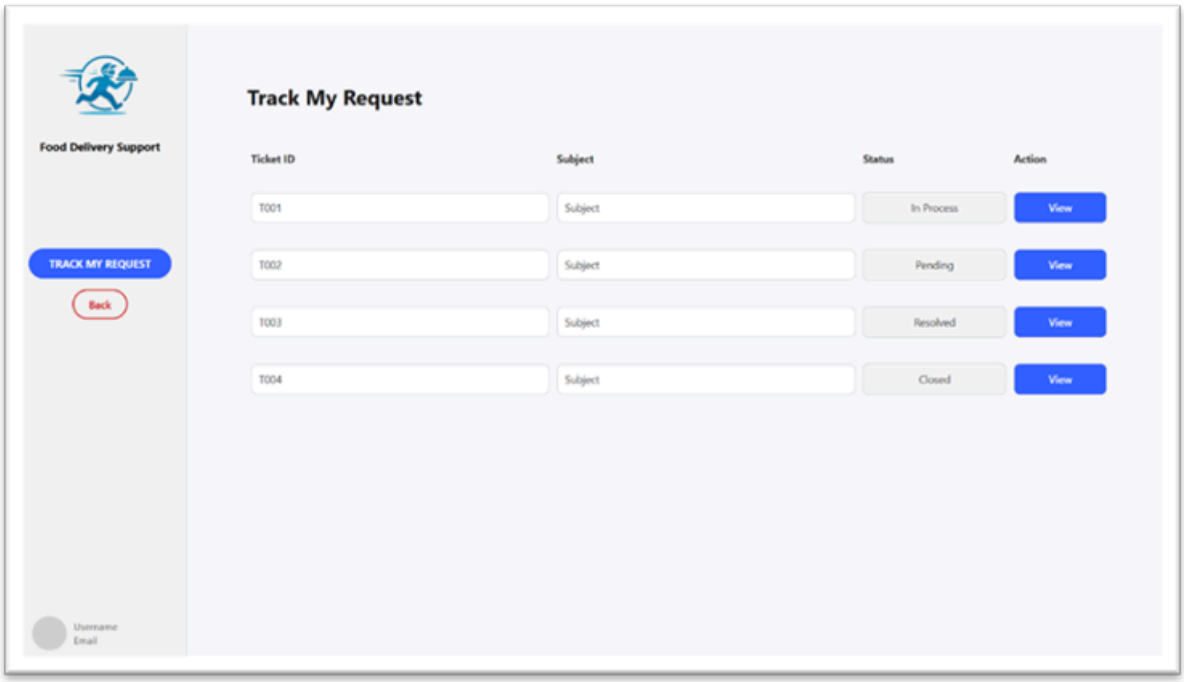


Figure 4.8: Wireframe of Track My Request Page

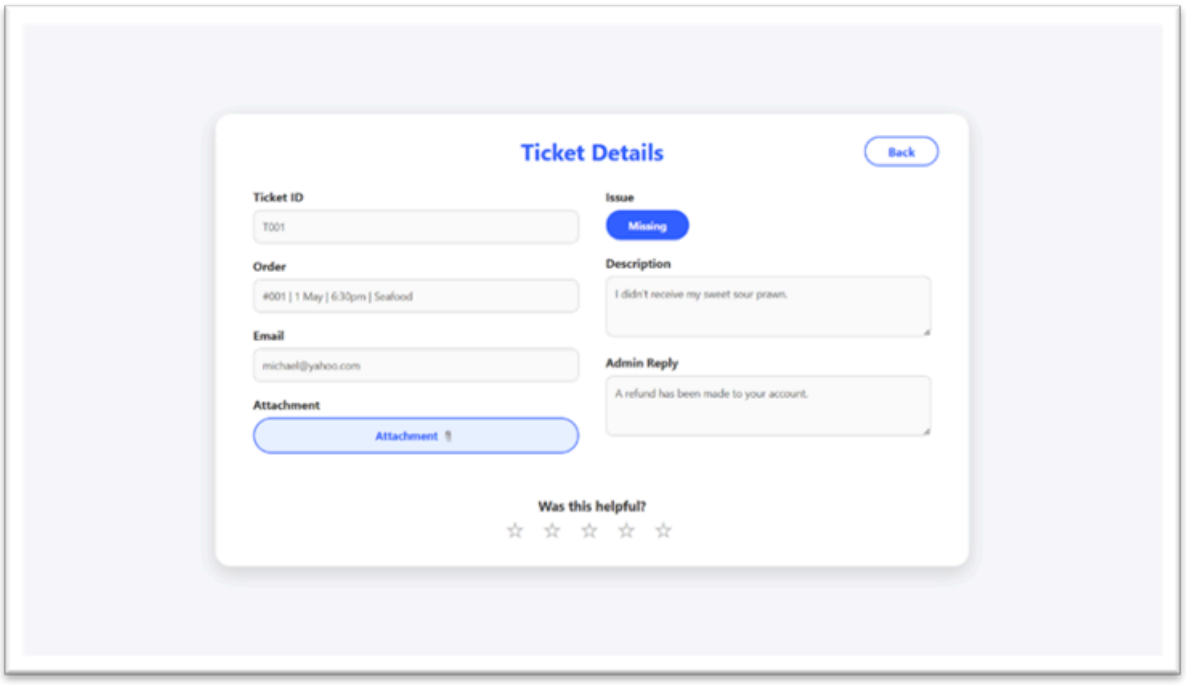


Figure 4.9: Wireframe of Ticket Details

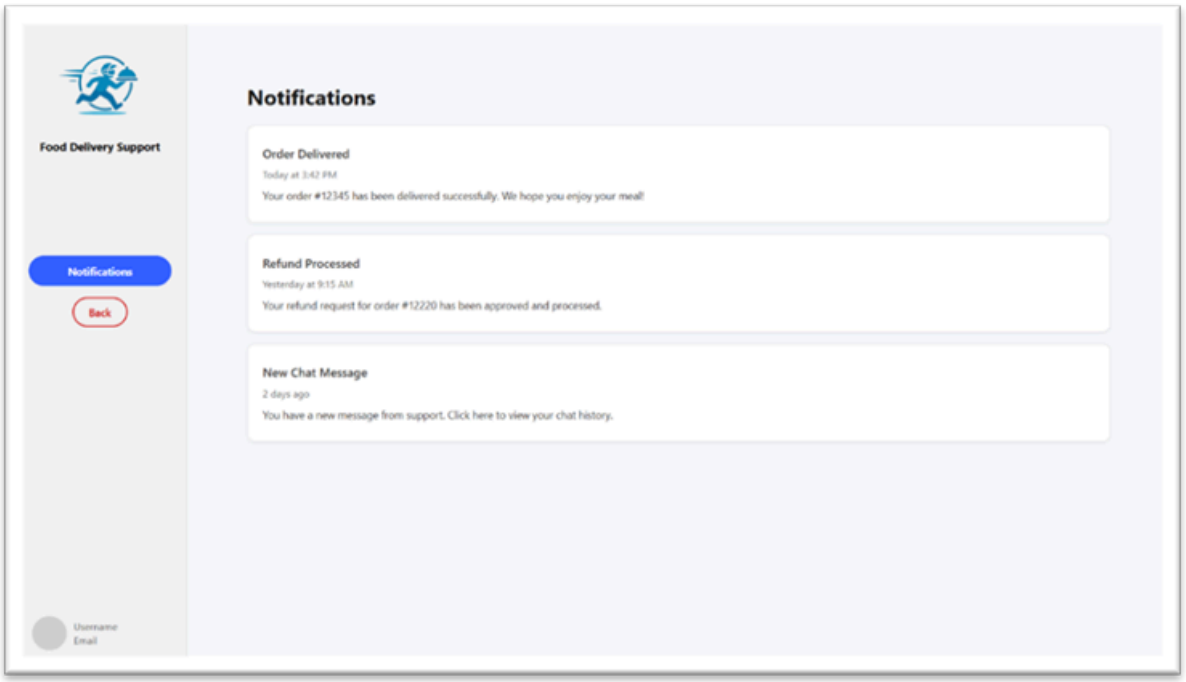


Figure 4.10: Wireframe of Notifications Page

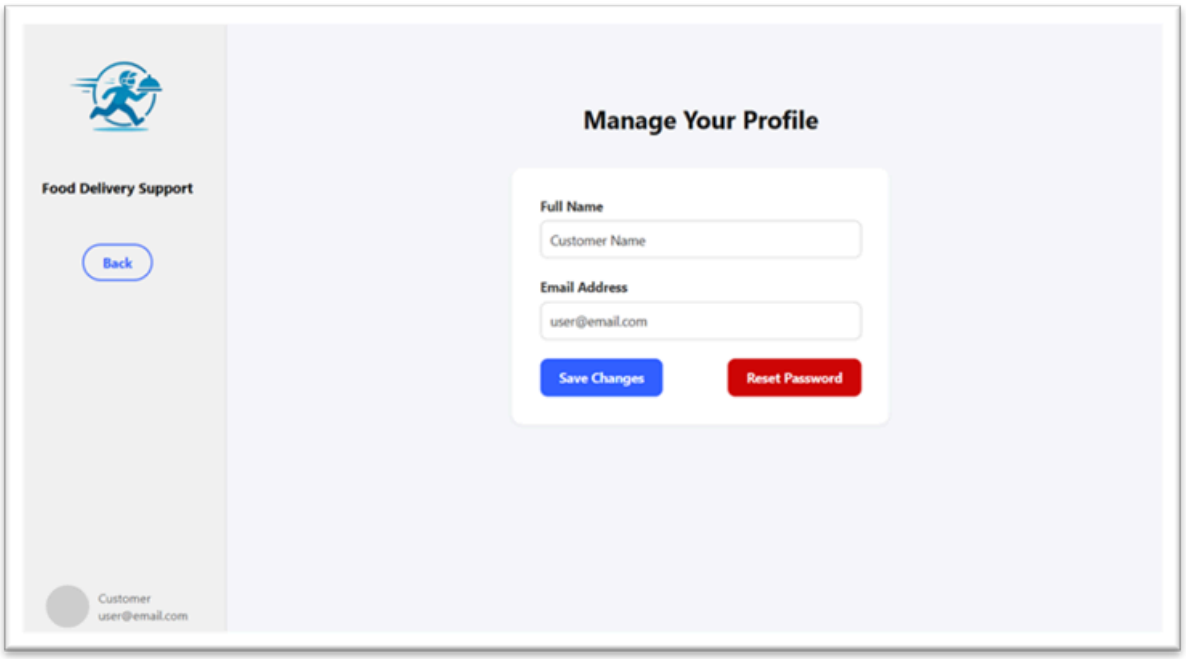


Figure 4.11: Wireframe of Manage Profile Page

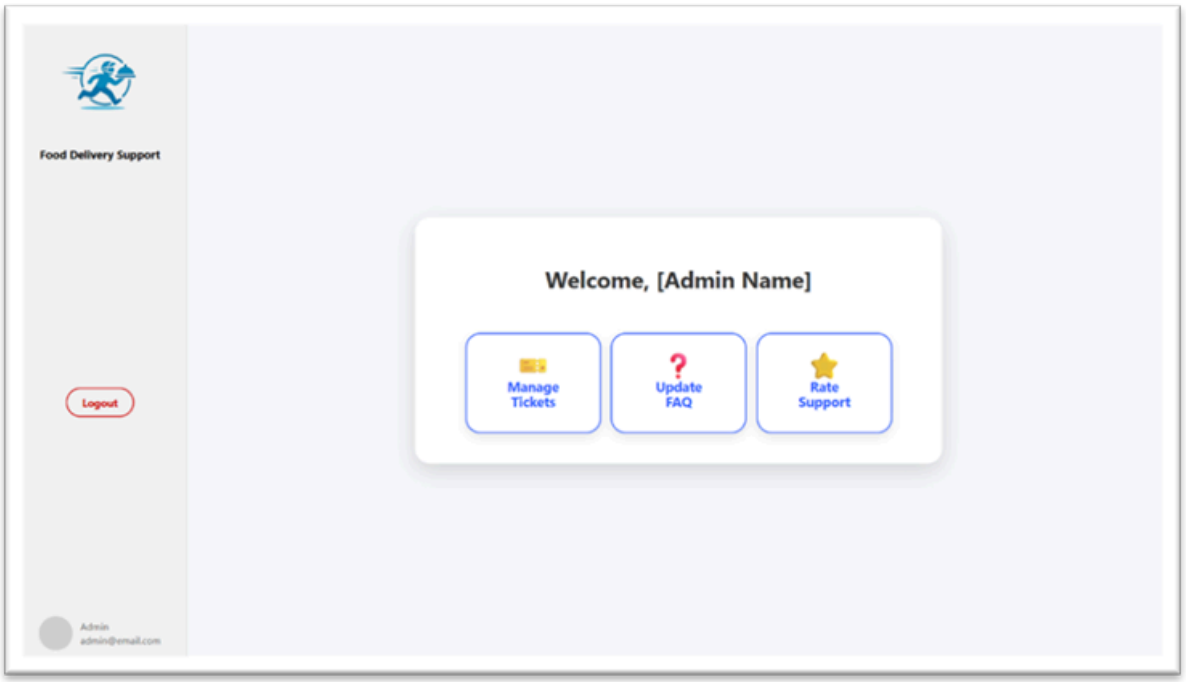


Figure 4.12: Wireframe of Admin Dashboard

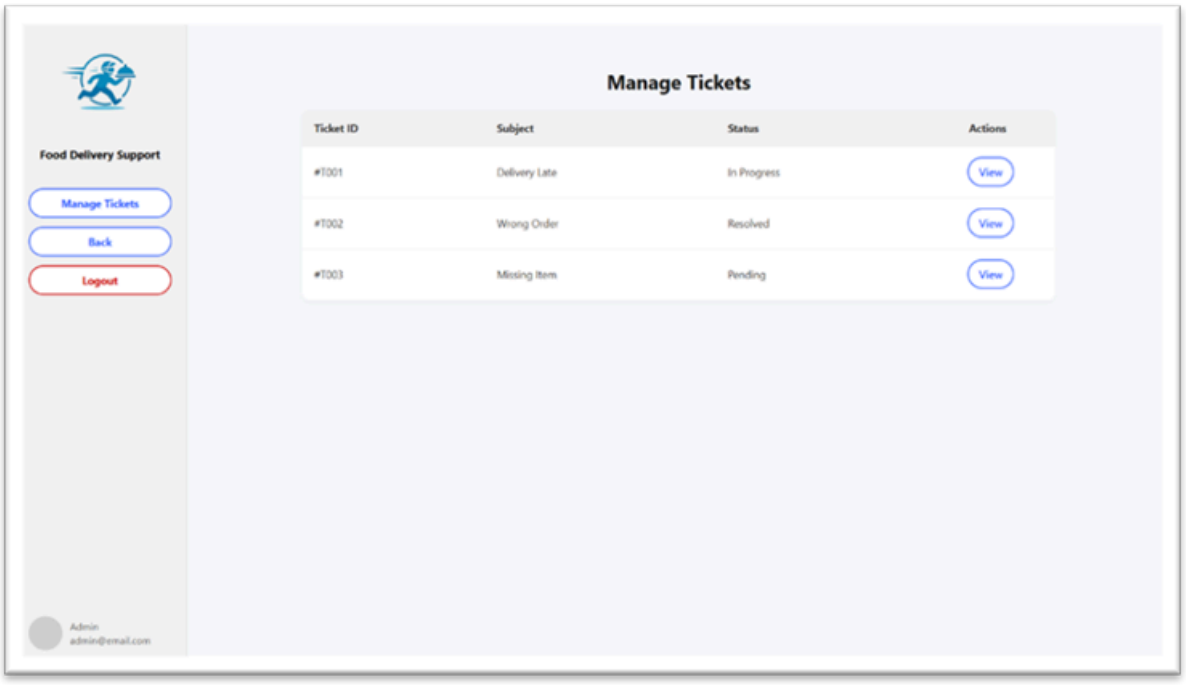


Figure 4.13: Wireframe of Manage Tickets Page

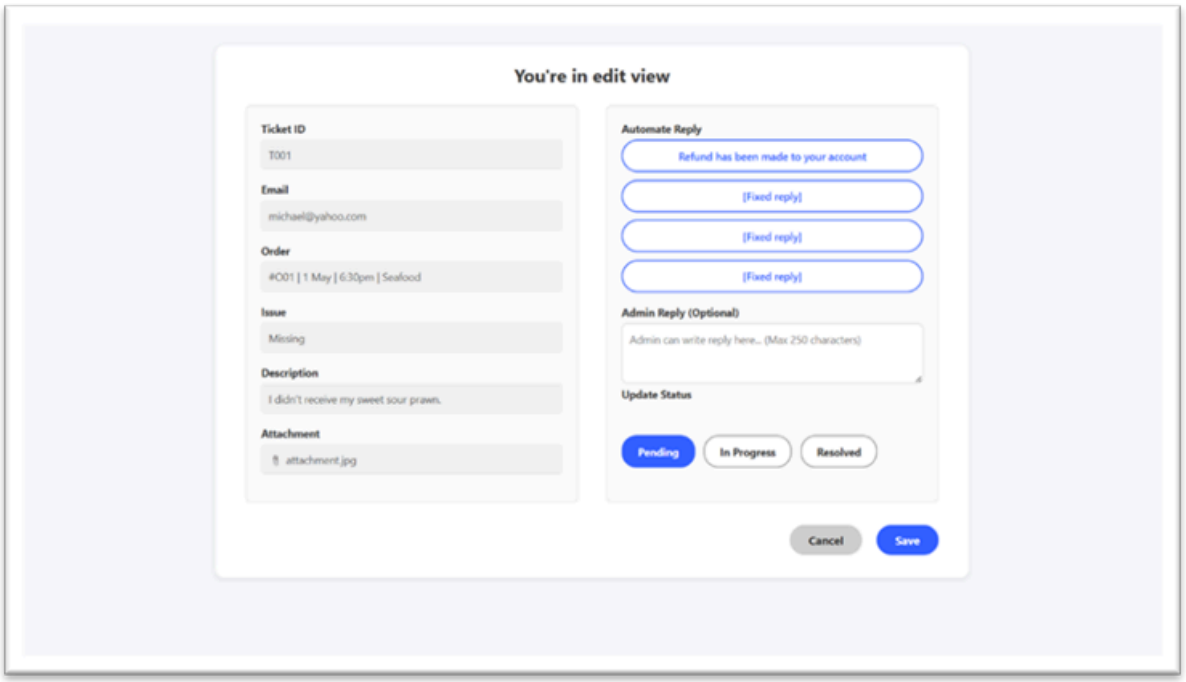


Figure 4.14: Wireframe of Manage Tickets View / Update Page

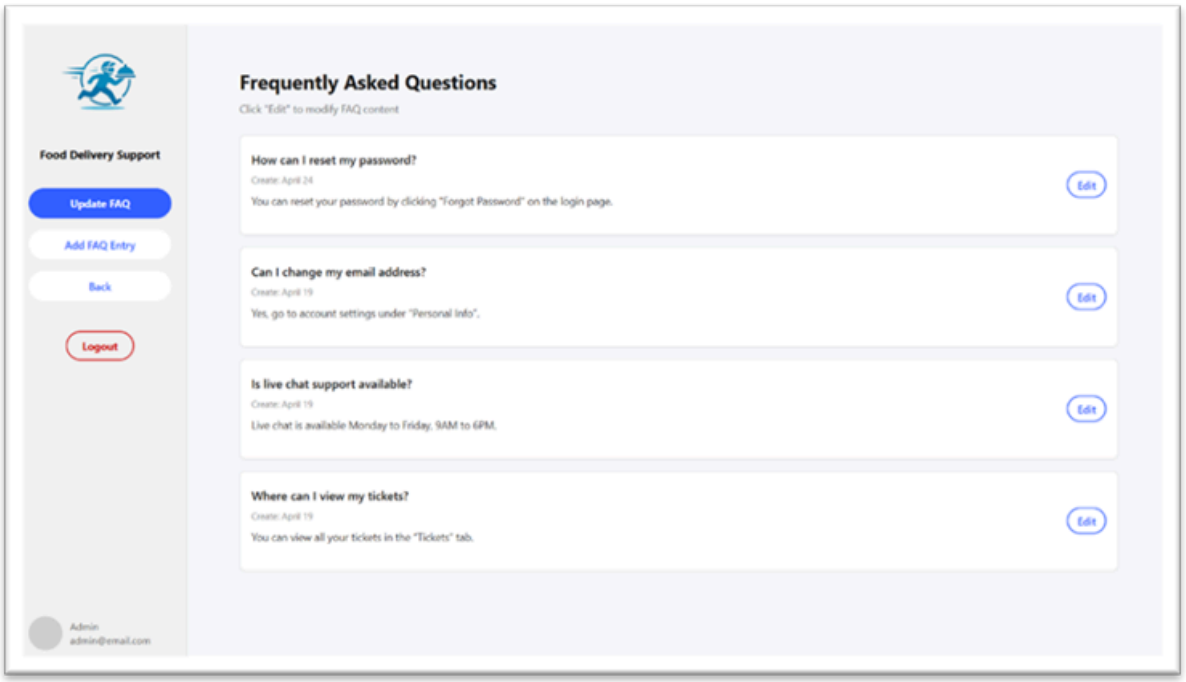


Figure 4.15: Wireframe of Update FAQ Page

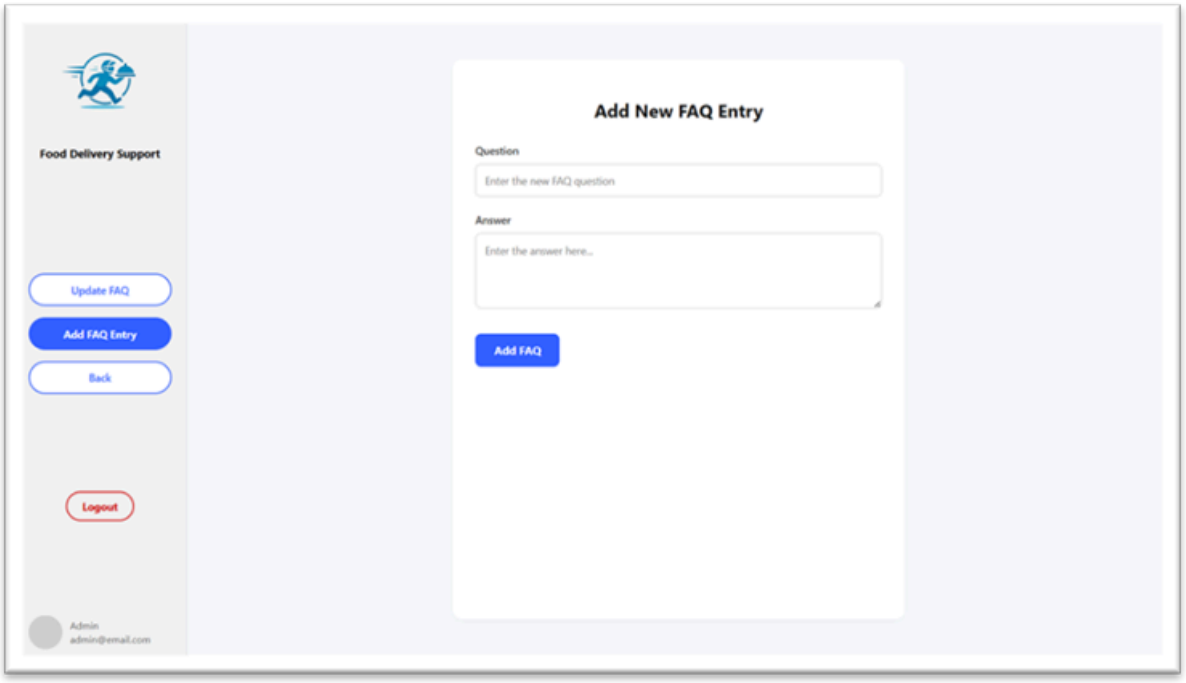


Figure 4.16: Wireframe of Add New FAQ Entry Page

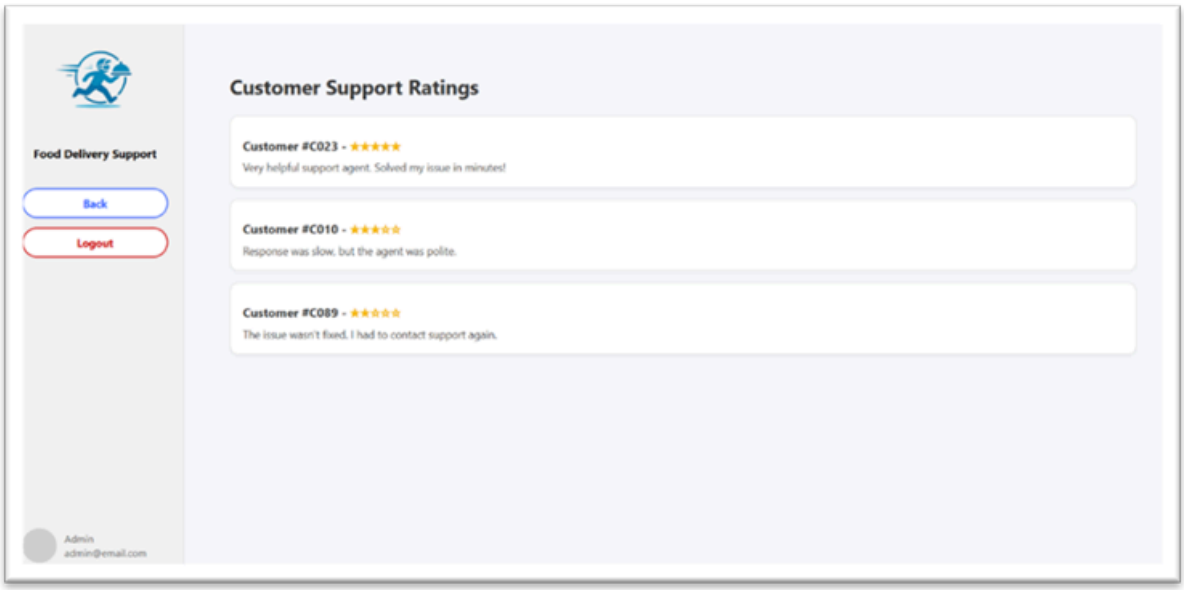


Figure 4.17: Wireframe of Rate Support Page

4.2 Navigation Flow

Customer Flow:

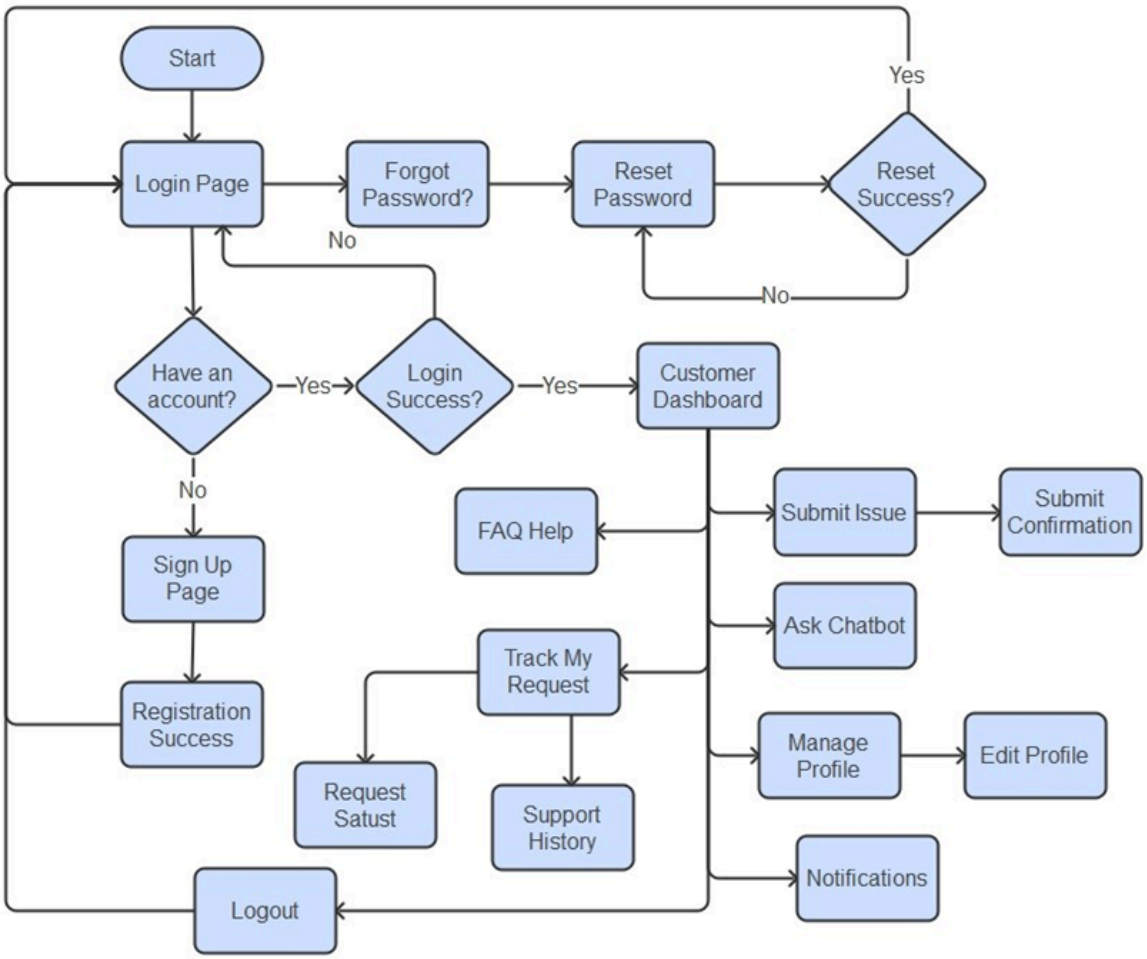


Figure 4.18: Customer Navigation Flow Diagram

Customer flow navigation is the path and structure customers follow to use and access different functionalities of our application. The flow includes processes like account registration, login, password retrieval, request submission, and support tracking, all aimed at an easy and smooth-to-use experience.

What follows is an outline of key elements and considerations:

1. Starting Point:

    - The flow begins with an initial state directing the user to the login page.

2. Login Process:

- If the user is already registered, login.
- Upon successful login, redirect the user to the Customer Dashboard.
- If a login fails, the user can choose to reset the password or try again.

3. Account Registration:
    - If the user has not registered yet, they may navigate to the Sign-Up page.
    - Upon successful registration, redirect the user to the Customer Dashboard.
4. Password Reset:
    - If the user clicks on "Forgot Password," they are taken to the reset password page.
    - After successful reset, the user can go back to the login page and try to login again.
5. Customer Dashboard:
    - Acts as the central point of customer activity.

    From this point, users can:

    - Submit issues (with submission confirmation).
    - Request the chatbot for help.
    - Visit FAQ and help pages.
    - Monitor requests and see support history.
    - Edit profile settings (including editing profile and see notifications).
    - See request status.
6. Logout Option:
    - Available at any time to safely end the customer session.

Admin Flow:

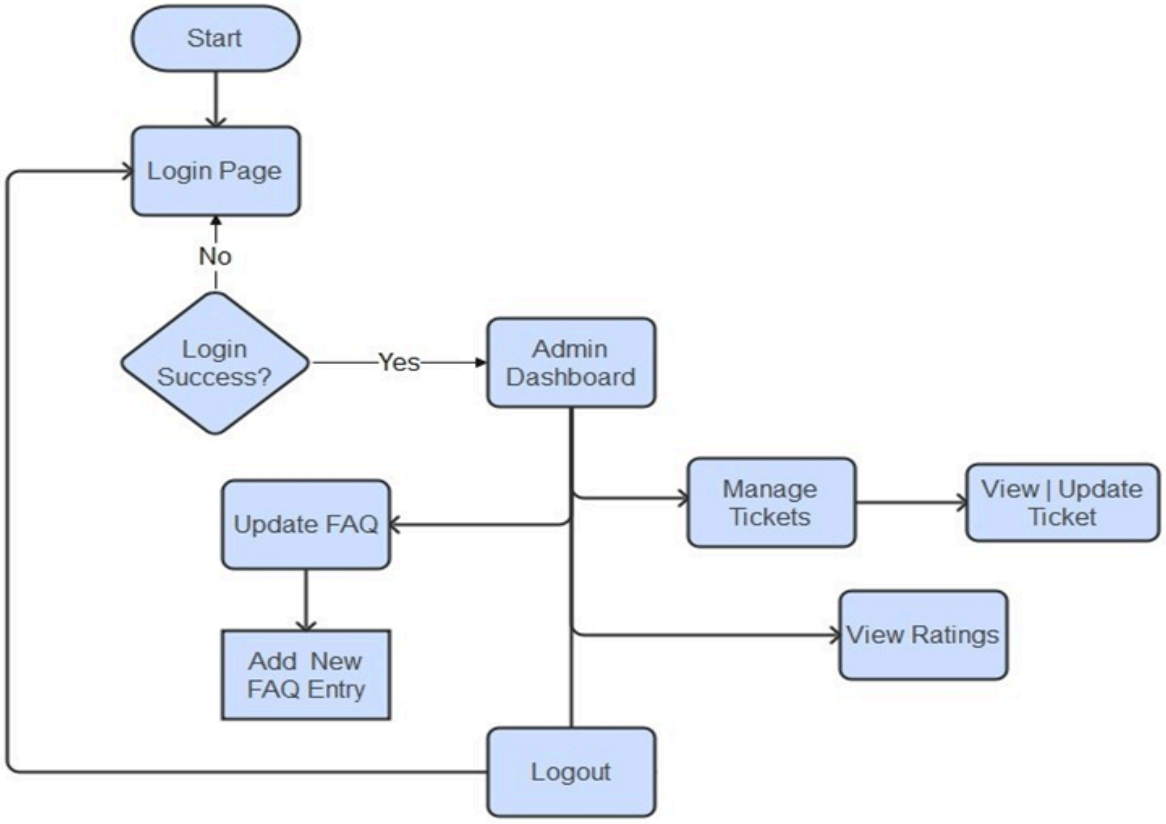


Figure 4.19: Admin Navigation Flow Diagram

Admin navigation flow refers to the structure and pathways users take to manage and administer different aspects of our application. The flow includes features like a dashboard, navigation menu and search functionality all designed to facilitate efficient administrative tasks.

Here's a breakdown of key elements and considerations:

1. Dashboard:

    - Lists available administrative sections or modules as clickable buttons (e.g., manage issues and tickets, update FAQ, and view ratings of support service).
    - Can be the entry point for navigating to other administrative sections.

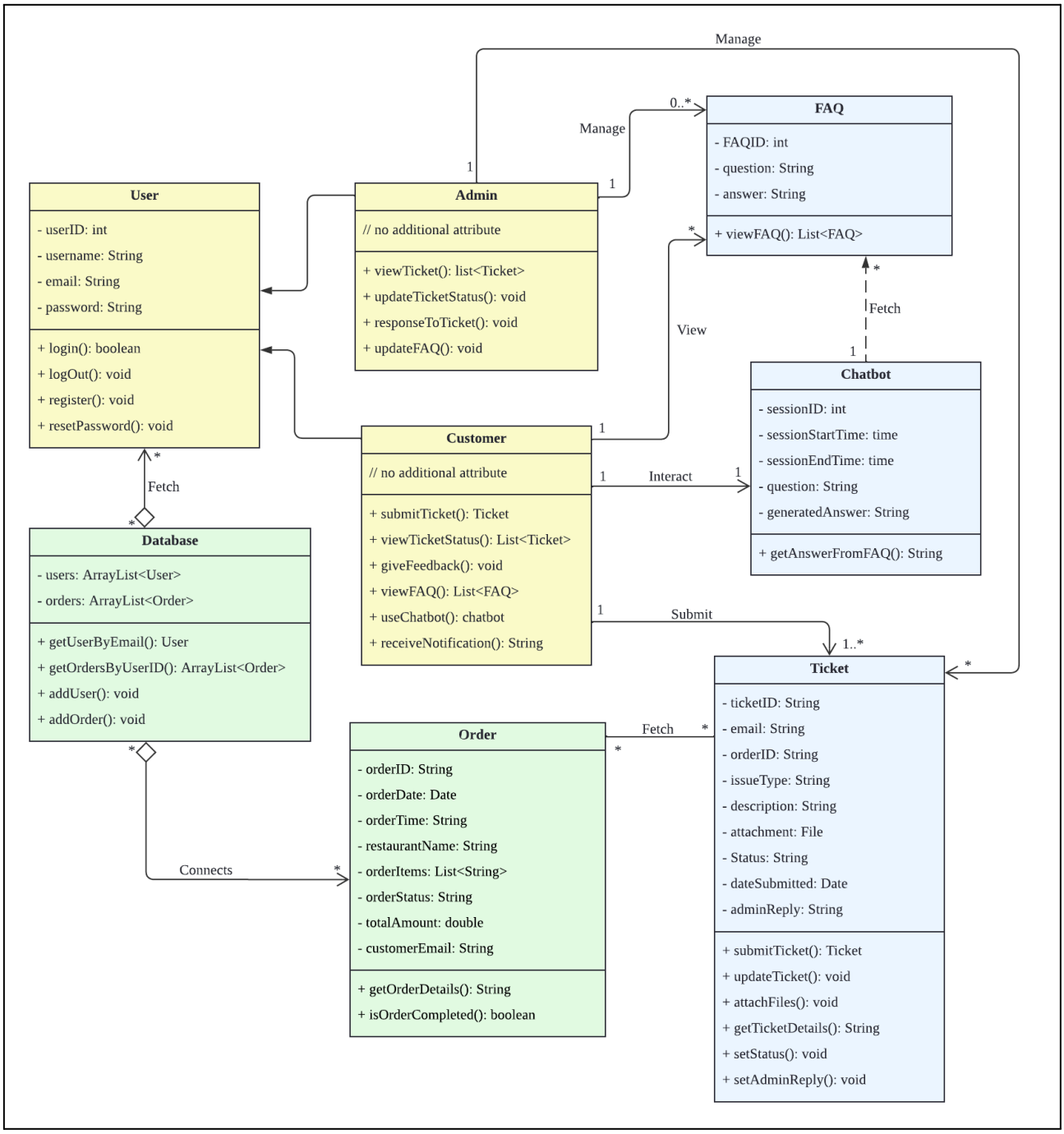
2. Navigation Menu:

    - Organized hierarchically based on the currently displayed page.
    - Lists other available administrative sections when the user navigates to a specific page. For example when the admin selects the FAQ option from the dashboard, the navigation menu will display the options related to the FAQ management, and a back button to go back to the dashboard.
    - Helps users easily locate and access specific areas of the application.
    - The logout option is available at hand in the navigation menu at all times.

5.0 SYSTEM DESIGN

The system design of this project includes object-oriented design and class diagrams that model the key entities and their relationships. The function of a class diagram and object-oriented design is to visually and structurally represent the system's components, showing how objects interact and relate to one another. Together, they provide a clear roadmap for developers to implement the system effectively.

5.1 Class Diagram



5.2 Object-Oriented Design

The class diagram above represents an object oriented design for food delivery customer support management systems.

5.2.1 Classes Relationships and Design Highlights

5.2.1.1 User

User class is important to identify users based on their role and thus comes with their own service functionality. They have attributes that consist of userID, username, password and email. They can either login(), logout(), resetPassword() and register(). They act as supertype to Admin and Customer.

5.2.1.2 Customer

If the User is a Customer, they apply inheritance where they will not have any attributes as they are a subtype of the User which is a supertype. In terms of tickets, they can submitTicket() where each submitted ticket will have their own ID, viewTicketStatus() for each ticket, giveFeedback() on the usage of the app, viewFAQ() section, receiveNotification() for any activity based on their preferences and useChatbot() in case they have any inquiries and need quick answers to. Methods such as viewTicketStatus() or submitTicket() could be overridden for role specific behaviour if needed.

5.2.1.3 Admin

If the User is an Admin, they also apply inheritance where they will not have any attributes as they are a subtype of the User which is a supertype. Their main role is to manage tickets such as responseToTickets(), updateTicketStatus() for customers to keep track of the progress, updateFAQ(), and viewTicket() to make reports.

5.2.1.4 Ticket

Ticket that are submitted by Customers have their own unique ticketID and are based on orderID and the userID that submits the ticket. The additional attribute of ticket that are important includes issueType (account/order/delivery), description so that customer can give a further detailed complaints, dateSubmitted as proof, status to make sure the customer is aware of the progress made, and adminReply to ease communication between the Customer and Admin. For each Ticket, an image or file can be added to attachment to guide the complaint, and the details of the ticket can be viewed by both the customer and admin. Methods include attachFiles(), getTicketDetails(), setStatus() and setAdminReply().

5.2.1.5 FAQ

Customers can view the FAQ page if they have questions about anything that is related to the app or how things work within. This page stores the predefined questions for customers self service. While admins are responsible to keep them updated. Each FAQ will have their own unique FAQID, and each will contain one question and the answer. The only method for this class is viewFAQ().

5.2.1.6 Chatbot

If the customer could not find the answer to their question, or did not understand the answer and requires a little bit more elaboration, they can access the Chatbot. The chatbot will document the sessionID, the time when sessionStart and sessionEnd, the question given/prompted and finally the generatedAnswer replied. The only method for this class is getAnswerFromFAQ(). They offer automated customer support by matching questions to relevant FAQs. The modular design supports easy extension which ensures scalability

5.2.1.7 Database

The database will fetch data from the User and connects it with the Order. It will getUserByEmail(), and getOrderByUserID() like playing match cards. Databases are crucial to track the tickets relation to orders in order to have a more systematic ticket system sorting. Design highlight of this class is encapsulation where all the data and behaviour of each class are discreet to their own class.

5.2.1.8 Order

Order is basically the food order made by the customer. They include important attributes like orderID, orderDate, orderTime, retsuarantName, orderItems, orderStatus, totalAmount, and customerEmail. They have the option to getOrderDetails() to determine if the orderCompleted(). These attributes are crucial to be cross checked by the admin based on the tickets submitted. This ensures the separation and concerns on data since each class has their own distinct responsibility.

6.0 EVENT HANDLING PLAN

This section explains how the system responds to the various events brought on by user activities inside the application. This is a comprehensive example of an event flow as well as a list of user interface events

6.1 List of UI Events

The system's primary user interface (UI) events are compiled in the following table:

Event Name	Event Trigger	Trigger From	Screen	Use Cass	JavaFX Implementation	Handler Functions
loginEvent	User clicks "Login	Client-side	LoginPage	UC01	LoginButton. setOnAction(...)	handleLogin()
registerEvent	Register	Client-side	RegisterPage	UC02	registerButton. setOnAction(...)	handleRegister()
submitTicketEvent	User clicks "Submit	Client-side	SubmitTicketPage	UC03	submitButton. setOnAction(...)	handleSubmitTicket()
viewTicketsEvent	View Tickets	Client-side	ViewTicketsPage	UC04	viewTicketsButton. setOnAction(...)	handleViewTickets()
ticketDetailsEvent	User clicks ticket for details	Client-side	TicketDetailsPage	UC04	detailsButton. setOnAction(...)	handleViewTicketDetails()
adminReplyEvent	Admin clicks "Reply	Client-side	TicketDetailsPage	UC08	replyButton. setOnAction(...)	handleAdminReply()

6.2 Event Flow Example

Send in the ticket along with the rating and attachment.

The way the system responds to user activities when a support ticket is submitted, an attachment is uploaded, and feedback is provided is demonstrated by the event flow that follows:

1. Handling Login Events

    - After entering their login information, the user hits the login button.
    - A LoginEvent is generated.

The user is authenticated by the system:

- The user is taken to the Customer Dashboard if it is successful.
- The system shows the error message "Invalid credentials" if authentication is unsuccessful.

2. Event Handling Ticket Submission

    - The user enters the issue details on the "Submit Ticket" page.
    - The SubmitEvent is triggered when the submit button is clicked.

The ticket is processed and saved by the system:

- "Your issue has been submitted successfully" is the confirmation message that appears if it is successful.
- The system displays an error notice if the submission is unsuccessful.

3. Managing Attachment Upload Events

    - To upload supplementary documents, the user selects the attachment button.
    - The event onAttachmentOpen() is called.

The file upload is processed by the system:

- The attachment is connected to the ticket if it is successful.

- The error message "File upload failed" appears if the upload is unsuccessful.

4. Handling Rating Submission Events

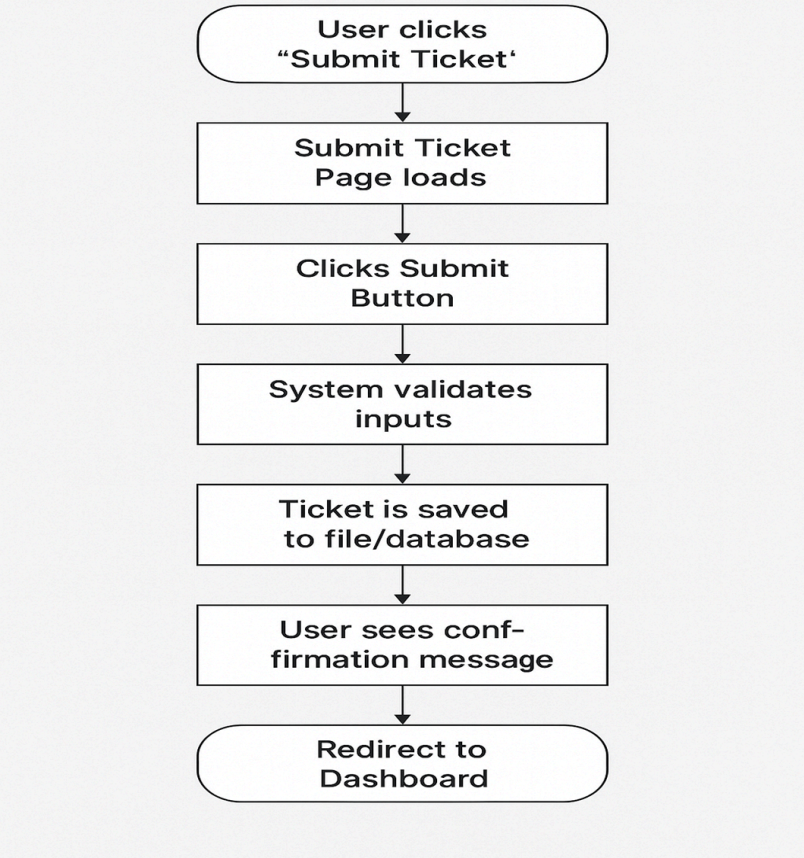
- Once the issue has been resolved, the user provides feedback.
- The event onRatingSubmit() is activated.

The feedback is saved by the system:

- If it works, the message "Thank you for your feedback" appears.
- An error message appears if saving is unsuccessful.

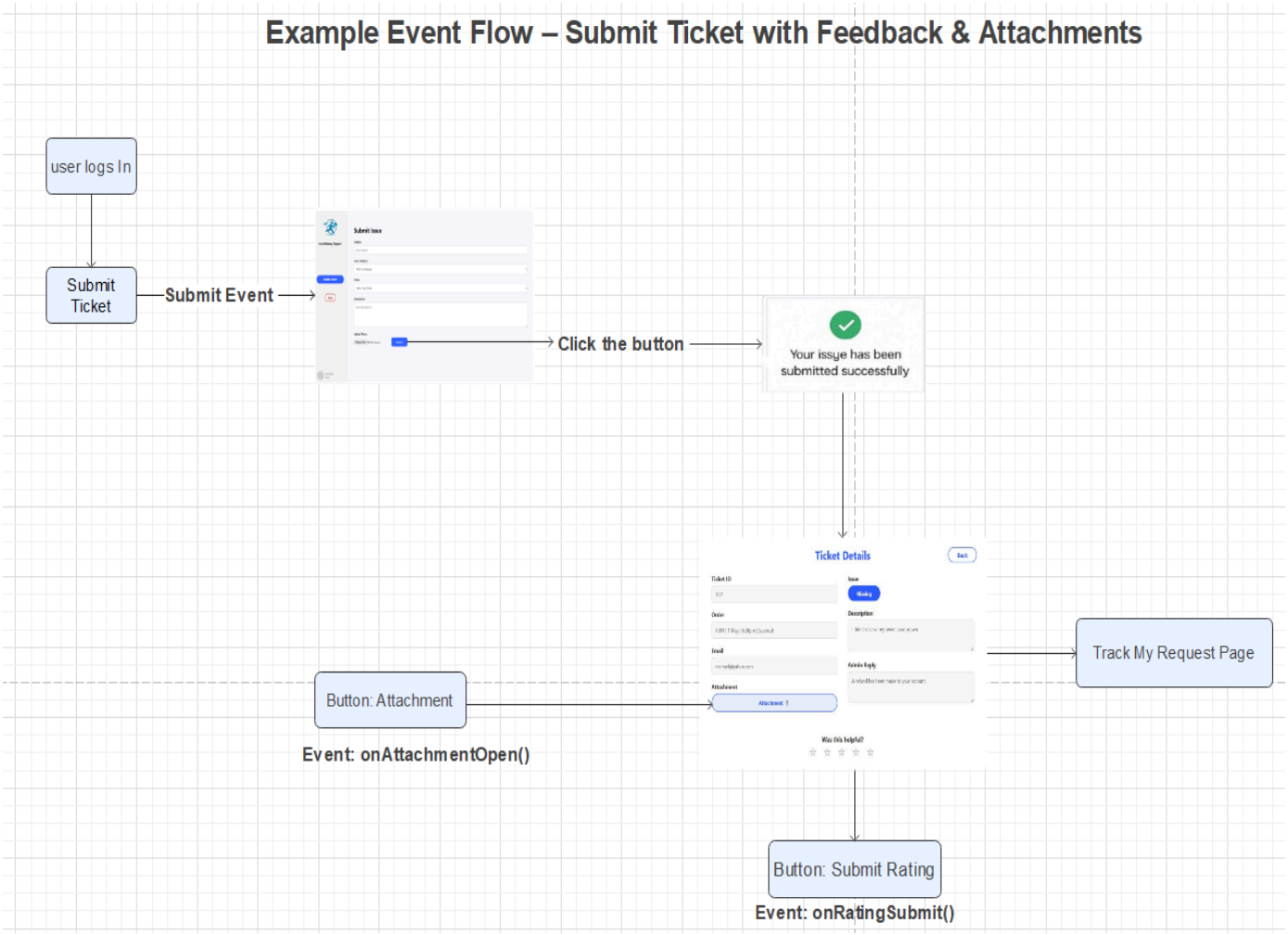
Appendix: Submit Ticket - Event Flow (Visual)

This diagram illustrates the flow of the 'Submit Ticket' process visually.



Appendix: Full System – Event Flow Diagram

This diagram represents the complete flow of user and admin interactions within the system, including login, ticket submission, response handling, chatbot usage, and profile management.



This section serves as the blueprint between design and development. It outlines exactly how each interaction will function and provides a structured plan for translating those interactions into working code.

7.0 DATA STORAGE STRATEGY

We use a simple data storage strategy that combines in-memory storage with file-based storage. ArrayLists store data while the system is running, and text files keep the data safe for future use. This method makes the system fast, reliable, and easy to manage.

7.1 Data Structure

We use two main ways to store data in our system: in-memory storage and file-based storage.

7.1.1 In-Memory Structures

While the program is running, we store data in ArrayList objects. These lists help us add, update, and get data quickly during use. The data we store includes:

Entity	Data Stored	Java Structure
Customers	Name, email, password	List<Customer>
Admins	Name, email, password	List<Admin>
Tickets	Subject, issue type, description, status, admin replies	List<Ticket>
FAQs	Question, answer	List<FAQ>
Chatbot History	Question asked, chatbot reply	List<ChatbotSession>
Notifications	Ticket updates, admin responses	List<Notification>
Activity Logs	Timestamps of user actions	List<ActivityLog>
Feedback	Rating, comments	List<Feedback>

7.1.2 File-Based Storage

We use .txt files to save data so we can load it again next time the app starts. Each type of data has its own file.

File Name	Stores	Format Example
customers.txt	Customer info	C001,sara,sara@gmail.com,passwordsara
admins.txt	Admin info	A001,Admin One,admin@support.com,securePass
tickets.txt	Ticket data	T001,C001,Missing item,Open,Admin reply here
faqs.txt	FAQs	How to cancel?,Go to My Orders and click Cancel
chatlogs.txt	Bot interactions	CL001,C001,"Where's my order?","It's on the way"
notifications.txt	Alerts	N001,C001,Your ticket has been resolved
activitylog.txt	Logs	C001,Submitted ticket,2025-06-16 10:00AM
feedback.txt	Ratings/comments	F001,C001,5,"Fast response!"

We use BufferedReader to read files and BufferedWriter to write to them in Java. Every time the user submits or updates data, it is written into the file.

7.2 Validation

Before saving any data, we check the input to make sure it is correct. This helps prevent errors and makes sure the system works well.

Form/Page	Fields	Validation Rules
Register Page	Name, Email, Password	- No empty fields - Email must include @ - Password min. 6 characters - Email must be unique
Login Page	Email, Password	- Email must exist - Password must match stored record
Submit Ticket	Subject, Issue Type, Description	- All fields required - Description min. 10 characters
Change Password	Current password, New password	- New password is not equal to current password - New password min. 6 characters
Feedback Page	Rating, Comment	- Rating must be 1–5 - Comment optional
FAQ Management	Question, Answer	- Both fields must not be empty

8.0 SYSTEM ARCHITECTURE

8.1 Architecture Pattern

Our system uses the Model-View-Controller (MVC) architecture pattern. This design helps us separate the system into three parts: Model, View, and Controller, so that the code is easier to manage, test, and update.

8.1.1 Model

The Model stores and manages the data of the application. It also includes the logic to update or get the data.

I. User (superclass)

II. Customer, Admin

III. Ticket

IV. FAQ

V. ChatbotSession

VI. Feedback, Notification, ActivityLog

.

These classes represent real data in the app, such as tickets, users, and FAQs. They also have methods to update or return data, like updateStatus() or getAnswer().

8.1.2 View

The View is the user interface (UI), what the user sees and interacts with. We designed our interface using JavaFX.

Examples of View pages in our system:

I. Login Page

II. Register Page

III. Submit Ticket Page

IV. Admin Dashboard

V. FAQ Page

VI. Chatbot Page

VII. User Profile Page

Each page is built using .fxml files in JavaFX and styled for easy navigation. The View gets information from the Model and displays it to the user.

8.1.3 Controller

The Controller handles user actions and connects the View and Model. When a user clicks a button or submits a form, the Controller receives that action and updates the Model or changes the View.

Example controllers in our system:

I. LoginController

II. TicketController

III. AdminController

IV. ChatbotController

For example, when a user clicks “Submit Ticket”, the TicketController checks the input, creates a new Ticket object, adds it to the list, and saves it to the file.

8.2 Component Breakdown

Component breakdown is done to list and explain each function of an individual component that makes up the whole user interface. The importance of doing this is to ensure that the developers understand the functionality of each component clearly.

a. Login Page
   - Email & password entry: For user Authentication
   - Login: Promt login
   - Forgot password: Incase user forgets their password/email and needs it reset.
   - Sign up: Option if users are first time users.

b. Sign up page
   - Email & password : User authentication
   - Confirm password: Ensure password entered is correct.
   - Sign up: Prompt registration
   - Log in: Prompt to login page again.

c. Reset Password page
   - Email Address: Original email address used during sign up.
   - New password: Make sure it is not the same as the old ones.
   - Confirm new password: For affirmation purpose
   - Reset Password: Prompt reset
   - Back to login: Prompt to login page

d. Customer dashboard
   - Sidebar menu: Consist of notification and log out option
   - Submit issue: Prompt to issue submission page
   - Track my request: Prompt to request tracking page
   - Ask Chatbot: Prompt to interact with chatbot
   - FAQ help: Display recent previous FAQ submitted by other users.

e. Submit issue page:
   - Sidebar menu: Options on issue submission and prompt customer dashboard.
   - Subject: Required as subject of ticket
   - Issue category: Dropdown menus for customer to select types of ticket
   - Order: Dropdown menus for customers to select which order.
   - Description: For customers to describe in detail on their ticket.
   - Choose file: Browse files to upload photo/document as proof.
   - Submit: Prompt submission

f. FAQ help page
    - Search bar: For customers to browse for relevant questions/topics.
    - Question table: Table with question, answer and date created.

g. Chatbot page
    - Conversation pane: Displays the interaction with chatbot
    - Message bar: For customers to send message to chatbot

h. Track my request page
    - Ticket ID: Displays Ticket ID
    - Subject: Displays ticket subject
    - Status: Displays current status of ticket
    - Action: Prompt to view details of the ticket

i. Ticket details page
    - TicketID: Displays ticket ID
    - Order: Display date, time and restaurant
    - Email: Displays email of user/ticket issuer
    - Attachment: Prompt to display supporting photo of ticket.
    - Issue: Display type of issue of ticket
    - Description: Display the further detail on issue
    - Admin reply: Displays the reply sent from admin to resolve issue
    - Rating bar: For users to rate whether the admin reply is useful.
    - Back: prompt Track my request page

j. Notifications page:
    - Tables: Displays time, subject and description of the notification

k. Manage profile page
    - Full name: Customers full name for identification
    - Email Address: new email address
    - Save changes: prompt to save the changes into the database
    - Reset password: prompt to reset password page

l. Admin dashboard
    - Sidebar menu: logout option
    - Manage tickets:Prompt to ticket management page
    - Update FAQ: prompt to FAQ updating page
    - Role support: Prompt to role support page

m. Manage tickets page
    - Tickets table: Displays ticketID, subject, status and actions
    - View: Prompt to see the description of ticket

n. View/Update page
    - Left pane: Displays details of ticket
    - Right pane: Automate reply prompt option
    - Admin reply: Text bar for further declaration/explanation on ticket
    - Update status: Options to refresh current ticket status
    - Cancel: Prompt to clear all changes made
    - Save: Prompt to save changes made.

o. Update FAQ
    - Tables to display questions and answer
    - Edit: Prompt to edit FAQ page

p. Add new FAQ entry page
    - Question: Text bar to fill in question
    - Answer: Text bar to fill in answer
    - Add FAQ: Prompt addition of entry into FAQ page

q. Customer support ratings page
    - Table: Displays userID, rating given and feedback description

9.0 TOOLS AND PLANNING

In this section, we share the tools we used throughout the development process, along with the planning timeline that helped us stay organised. Since we worked as a team, it was important to divide our tasks clearly and use tools that made collaboration easier. The timeline also gave us a clear view of what to focus on each week, from start to finish.

9.1 Tools Used

As a team, we used a combination of development, design, and documentation tools to help us build and present our Food Delivery Support Application smoothly. Each tool served a different purpose based on our individual roles and project needs.

Tool / Software	Purpose / Usage
Java	Used as the main programming language to implement the system logic.
JavaFX	To create the graphical user interface (GUI) for customer and admin features.
NetBeans IDE	Our team used NetBeans to write, run, and debug our JavaFX application efficiently.
Canva [tbc]	Helpful for designing simple and clean wireframes for each screen in our app.
Lucidchart	Used to create use case diagrams, class diagrams, and navigation flows.
Google Docs	We collaborated on our documentation in real-time, making it easier to edit and track contributions.

9.2 Timeline

The project was divided into two phases: Part A for documentation and Part B for system development. While the main focus early on was completing Part A before Week 9, some development work was also started in parallel to manage the timeline efficiently.

Week	Task / Milestone
Week 1	Topic selection and initial task distribution among group members.
Week 2	Drafted the system's functional and non-functional requirements.
Week 3	Use case tables and diagrams were prepared.
Week 4	Wireframes and navigation flow were designed.
Week 5	Completed the class diagram and object-oriented design.
Week 6	Final edits on documentation and internal review of written sections.
Week 7	Refined the documentation while beginning initial setup for JavaFX UI.
Week 8	Finalized the documentation and started implementing login and registration.
Week 9	Implementation of ticket system and admin features.
Week 10	Implementation of chatbot and FAQ.
Week 11	Continued testing, bug fixing, and UI refinement.
Week 12	Final review with client and application deployment.

10.0 SUMMARY

The food delivery support app is designed and created to help users resolve their food delivery related issues by allowing them to use and access all these features on the app. With the features created on the app it helps customers or users to ease their worries and concerns by providing a simple and easy way to report their issues.

10.1 Team Roles & Contribution

Name	Roles	Contribution
Nur Azra Zulaikha	Group Leader	Assigned team tasks and contributed to project overview and final touches.
Nur Laila Maisarah Bt Noorhelmee	Project Manager	Managed project document, handled use case and planning.
A'isyah Nabihah Marlehan	System Designer	Developed and set up database structure.
Alhamary Juri Dham O	UI/UX Designer	Designed wireframes in Figma and implemented JavaFX UI.
Alshamary Bader Dham O	UI/UX Designer	
Isabelle Tan Yuin	Requirement Analyst	Gathered user needs, and helped define functional & non-functional requirements.