YumYard

Cuisine Popular and Festivals

Software Requirements Specification

Version <1.0>

Submitted in Partial Fulfillment for the Award of Degree of Bachelor of Technology in Information Technology from Rajasthan Technical University, Kota

MENTOR:

Dr. S.R. Dogiwal

(Dept. of Information Technology)

COORDINATOR:

Dr. Priyanka Yadav

(Dept. of Information Technology)

SUBMITTED BY:

Asha Sharma (21ESKIT024)

DEPARTMENT OF INFORMATION TECHNOLOGY

SWAMI KESHWANAND INSTITUTE OF TECHNOLOGY, MANAGEMENT & GRAMOTHAN

Ramnagaria (Jagatpura), Jaipur – 302017

SESSION 2024-25

Contents

1 Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 1

1.4 References 2

1.5 Technologies to be Used....2

1.6 Overview 2

2 Literature Survey 3

2.1 Food Support Systems 3

2.2 Mobile Applications in Food Delivery....3

2.3 Research Papers and Publications 3

2.4 Conclusion 3

3 Specific Requirements 4

3.1 Functional Requirements....4

3.2 Non-Functional Requirements 5

3.3 Hardware Requirements 5

3.4 Software Requirements 5

3.5 Agile Methodology....5

3.6 Business Process Model....6

3.7 Supplementary Requirements 6

4 System Architecture 7

4.1 Client-Server Architecture 7

4.2 Communications Interfaces.... 8

5 Overall Description 9

5.1 Product Feature 9

5.2 ER Diagram 10

5.3 Class Diagram 11

5.4 Use-case Model Survey 12

5.5 Behavior Diagrams....12

5.5.1 Sequence Diagram 12

5.5.2 Activity Diagram 13

5.6 Assumptions and Dependencies....14

5.6.1 Assumptions 14
5.6.2 Dependencies 14
6 Supporting Information 15
6.1 List Of Figures 15
7 Conclusion and Future Scope 16
7.1 Conclusion 16
7.2 Future Scope 17
8 Concerns/Queries/Doubts if any 18

1. Introduction

1.1 Purpose

The primary objective of this SRS is to establish a clear and unified understanding among stakeholders, including developers, designers, project managers, domain experts in Festivals , foods, and end-users, regarding the functionalities, constraints, and objectives of the " Yum Yard " platform. By defining and documenting these requirements, this document aims to guide the development team in creating a robust, user-centric application that addresses the needs of users and enhances Ordering practices.

1.2 Scope

The Food Delivery Festival Website will provide a user-friendly platform where customers can explore various food offerings associated with different festivals, allowing for online ordering. The system will incorporate user authentication, a shopping cart feature, and a responsive design suitable for various devices.

1.3 Definitions, Acronyms and Abbreviations

- **SRS:** Software Requirements Specification
- **API:** Application Programming Interface
- **UI**: User Interface
- **UX:** User Experience
- **Dashboard**: A visual display of food data and statistics within the application

1.4 References

- **Django Documentation:** https://docs.djangoproject.com/en/5.1/
- **React Documentation:** https://reactjs.org/docs/getting-started.html

1.5 Technologies to be used

The "Yum Yard" platform utilizes cutting-edge technologies and frameworks to deliver its functionalities, including:

- **Frontend:** React.js, Redux for state management.
- **Authentication:** JWT Authentication for user login and management.
- **Backend:** Django Rest Framework.
- **Database:** SQLite.
- **Deployment:** Firebase Hosting for frontend deployment.

1.6 Overview

This SRS document is structured systematically into distinct sections, each detailing specific facets of the "Yum Yard" platform. It includes an executive summary, system overview, detailed descriptions of functional and non-functional requirements, user interfaces, system constraints, assumptions, dependencies, and technical specifications. Additionally, it incorporates diagrams, use cases, and mockups to provide a comprehensive understanding of the system's architecture and functionality.

2. Literature survey

1. Food Support Systems

The survey focused on existing food support systems, both digital and traditional, analyzing their functionalities, user interfaces, and features to understand their impact on food sourcing and delivery. Notable platforms reviewed include local food marketplaces and farm-to-table initiatives. By collaborating with these systems, our food delivery website aims to ensure that we source ingredients from responsible and sustainable suppliers, benefiting both local producers and our customers.

2. Mobile Applications in Food Delivery

An analysis of mobile applications designed for the food sector revealed prevalent features that enhance user experiences and increase adoption rates among consumers. Key aspects examined include menu browsing, order tracking, meal recommendations, and payment solutions. Our platform, built with React, leverages a responsive design to ensure a seamless ordering experience across devices.

3. Research Papers and Publications

Relevant research papers, publications, and articles related to food technology and delivery services were surveyed to gather insights into best practices. This research informs our use of technologies like Django for backend support and ensures that we implement effective strategies for sourcing, preparation, and distribution.

Conclusion

By leveraging technologies like React, DRF, and Redux, we are committed to creating an intuitive, fast, and reliable platform. Our goal is to enhance the quality of our offerings and bridge the gaps in the food delivery system, ensuring equitable access to delicious, fresh meals for all.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Authentication

- Users can register using email and password.
- JWT tokens are used to authenticate the user.

3.1.2 Food Browsing

- Users can explore food items categorized by festivals (e.g., Diwali, Christmas) also.
- Users can search food items.
- Each item will have detailed descriptions, images, and pricing.

3.1.3 Cart Functionality

- Users can add items to their cart.
- Users can modify item quantities and remove items from the cart.
- A summary of the cart will display total costs, including taxes and delivery fees.

3.1.4 Vendor Integration Functionality

- When users browse the item listings, each item displays information about the vendor offering it.
- If multiple vendors offer the same item, the listing will show nearby vendors and their availability.
- Users can view vendor details such as name, distance, and ratings (if available).

3.1.5 Checkout Process

- Users can enter delivery addresses and payment information.
- Django will store all these data in models.

3.2 Non-Functional Requirements

3.2.1 Performance

- The website should load within 3 seconds for optimal user experience.
- The system should handle at least 1,000 concurrent users..

3.2.2 Usability

- The UI must be intuitive, allowing easy navigation for all users.
- The design should be responsive, functioning seamlessly on both mobile and desktop devices

1.1.1 Security

- User data must be encrypted and securely stored in Django admin.
- Compliance with GDPR and other data protection regulations is required.

1.1.2 Reliability

- The system must maintain 99.9% uptime, with automatic failover capabilities.
- Regular backups of the database must be scheduled.

3.3 Hardware Requirements

- Firebase hosting will manage the infrastructure.
- Development machines should have modern web browsers and Node.js for local development

3.4 Software Requirements

The software requirements include:

- **Frontend:** React.js, Redux
- **Database:** SQLite
- **Authentication:** JWT Authentication
- **Development Tools:** Git

3.5 Agile Methodology

The project will adopt Agile methodologies, enabling iterative development cycles with regular feedback loops. Each sprint will be two weeks long, culminating in a review session to demonstrate progress and gather stakeholder input.

3.6 Business Process Model

- User registration and authentication.
- Browsing food items categorized by festival.
- Adding items to the shopping cart.
- Checkout and payment processing.
- Order confirmation .
- Vendor Registration

3.7 Supplementary Requirements

These are additional requirements that support the functionality and reliability of the system:

- Data Backup and Recovery: Regular backups and a disaster recovery plan to ensure data protection in case of system failures.
- User Training and Support: Comprehensive training sessions for users, along with a help desk or support team to address issues.
- Documentation: Detailed documentation, including user manuals, technical specifications, and maintenance guidelines for easy reference.

4. System Architecture

4.1 Client-Server Architecture

The Cuisine Popular and festivals uses a client-server architecture to ensure efficient, centralized data processing and secure access across multiple user devices. This architecture provides a clear separation of responsibilities between the client side (frontend interface) and the server side (backend processing and database management).

- **Client Side (Frontend)**: The client side is built with HTML, CSS, and React, providing an intuitive user interface accessible through web browsers on desktops, tablets, and mobile devices.
- **Server Side (Backend)**: The backend, built using Django handles business logic, authentication, and data processing. It includes routes for various system functionalities, such as login, record entry, and case management, and interacts with the database to store, retrieve, and manage order records.
- **Database**: The SQLite database stores structured data, including food records, user information. It ensures data consistency, reliability, and easy retrieval.
- **Security and Role-Based Access Control**: Role-based access control restricts data access according to user roles, ensuring that only authorized personnel can view or modify sensitive information.

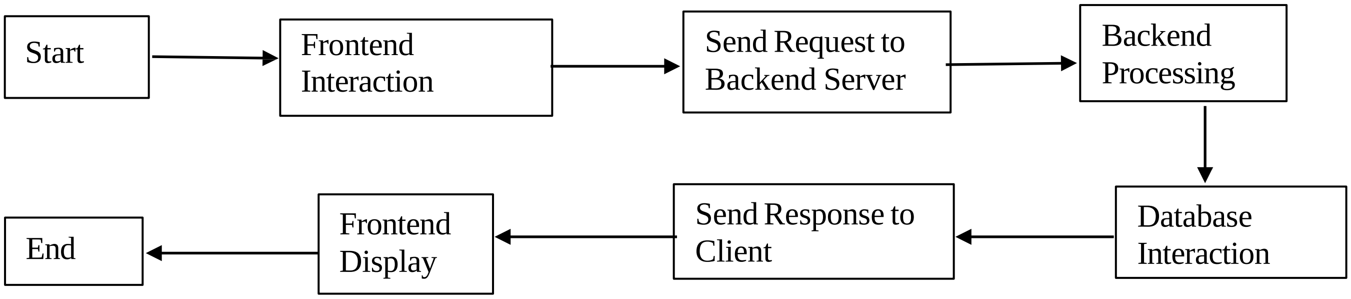


Figure 4.1: Client Server Architecture

4.2 Communication Interfaces

The communications interface**s** of the food delivery system facilitate secure, efficient data exchange between the client and server, as well as interoperability with other law enforcement systems. The primary communication protocols and interfaces are as follows:

- **HTTP/HTTPS Protocols**: The System uses HTTP/HTTPS protocols for secure client-server communication. All API requests from the client side to the server are sent over HTTPS to encrypt data in transit, protecting against unauthorized access or interception. HTTPS also ensures data integrity and authentication between the client and server.

- **RESTful API**: The backend server provides a RESTful API interface for handling requests from the client. The API follows standard REST principles for CRUD operations (Create, Read, Update, Delete), enabling consistent and predictable communication between the frontend and backend. The API includes endpoints for login, order management, search functionality, and statistical data retrieval, allowing smooth interaction and data exchange between client and server.

- **Authentication Interface**: The system uses JWT for user authentication. When users log in, they receive a JWT, which is included in each subsequent request for secure authentication. The server validates the token, ensuring only authenticated users access the system.

5. Overall Description

5.1 Product Feature

Mainly in backend have two apps one is users (for login,register) , second orders (for cart , cartItems , order , orderItems , Address) for the project foodsite

1. User Authentication and Management

- Sign Up / Sign In: Users can create an account and log in using email/password .
- **api/users/login** is the endpoint of api which allow successfull login .
- **api/users/register** for registering user in backend
- Authentication with JWT: JWT Authentication handles user sign-in and session management, ensuring secure access to user data.
- Profile Management: Users can update their personal information, delivery addresses, and payment methods.

2. Menu and Item Display

Endpoints:

- **api/orders/cart -**for viewing all cart details
- **api/orders/cartItems -** for viewing cart items of particular person
- **api/orders/items** – for all the items added
- **api/orders/cart/add** – for adding items in cart
- **api/orders/cart/decrease** – decreasing the quantity of item added in cart
- **api/orders/cart/item/<int:item_id>/** - for removing items from cart
- **api/orders/search** – for searching
- **api/orders/registers** – for registering vendor
- Menu Browsing: Users can view a list of food categories and items (e.g., pizza, sushi, burgers) with high-quality images, descriptions, and prices.
- Dynamic Filtering & Sorting: Allow users to filter food items by type, price, ratings, etc., and sort by popularity or price.

- Search Functionality: A search bar for users to easily find specific dishes or ingredients.
- Item Customization: Option to customize food items (e.g., add extra cheese, select spice level).

3. **Shopping Cart and Order Management**

Endpoints:

- **api/orders/place-order/ -** for placing order
- **api/orders/track-order/ -** for tracking order
- Add to Cart: Users can add items to their shopping cart with the option to modify quantities or remove items.
- Cart Persistence: Django backend can be used to save cart data, allowing users to retrieve their cart even if they log out or refresh the page.
- Order Summary: Show an overview of the order, including item details, quantities, and total price (with tax, delivery fee).

5.2 ER Diagram

The Entity-Relationship (ER) Diagram for the Food ordering site represents relationships between entities such as Customer, Menu , Orders and Bank payments etc**.**

- **Entities:**
  - User: Attributes include userID, name, role, and contactInfo.
  - Order Record: Attributes include orderID, CustomerName,total.
  - Food record: Attributes include IsPrepared, IsDelivered, Remarks,Quantity.
  - Menu: Attributes include MenuID,Price ,Availability.
- **Relationships:**
  - User-Orders Record: One-to-many, with users authorized to add/update records.
  - Order-Food Record: many-to-many, associating orders with food records.

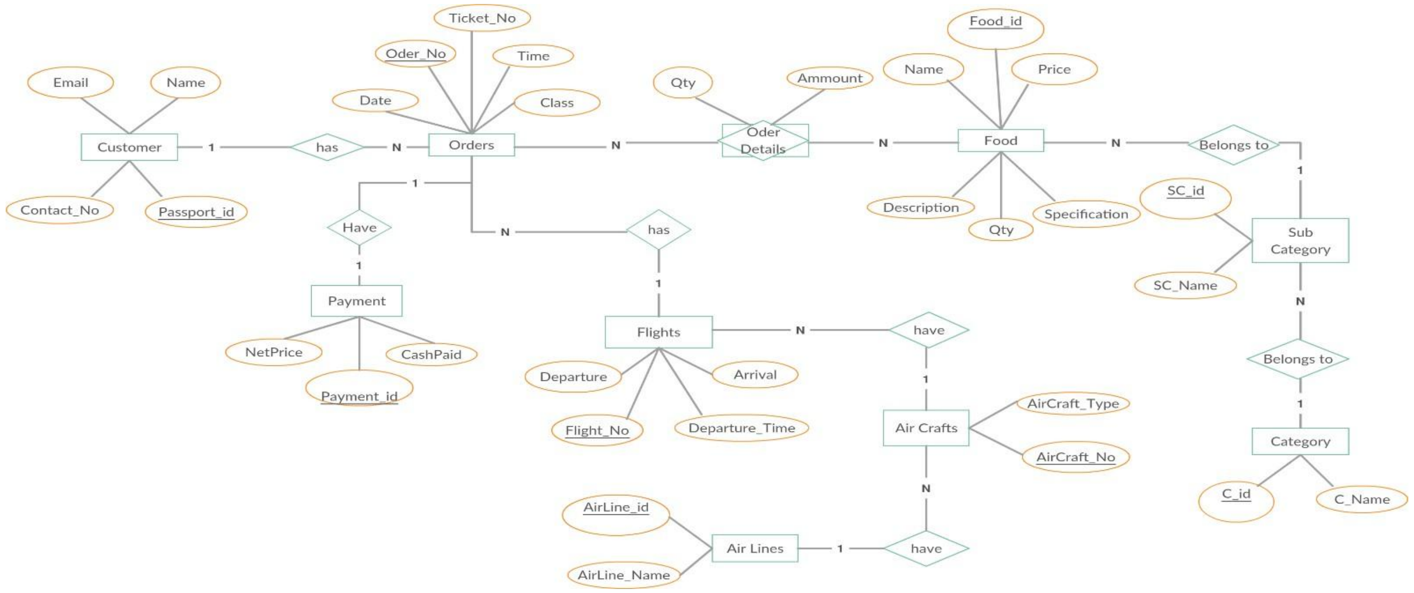


Fig 5.2.1: ER Diagram

5.3 Class Diagram

The **Class Diagram** outlines the structure and interactions of main classes in the Food delivery system:

- **Classes**:
  - User: Attributes include userID, name, role, and contactInfo.
  - Order Record: Attributes include orderID, CustomerName,total.
  - Food record: Attributes include IsPrepared, IsDelivered, Remarks,Quantity.
  - Menu: Attributes include MenuID,Price ,Availability.
- **Associations**:
  - User interacts with Menu Record and Order.

- **Methods**:
  - User methods include Send_Order(), Receive_Order().
  - MenuRecord methods include UpdateMenu(), DeleteMenu(),GetMenu().
  - Order methods include Confirm(), Cancel().

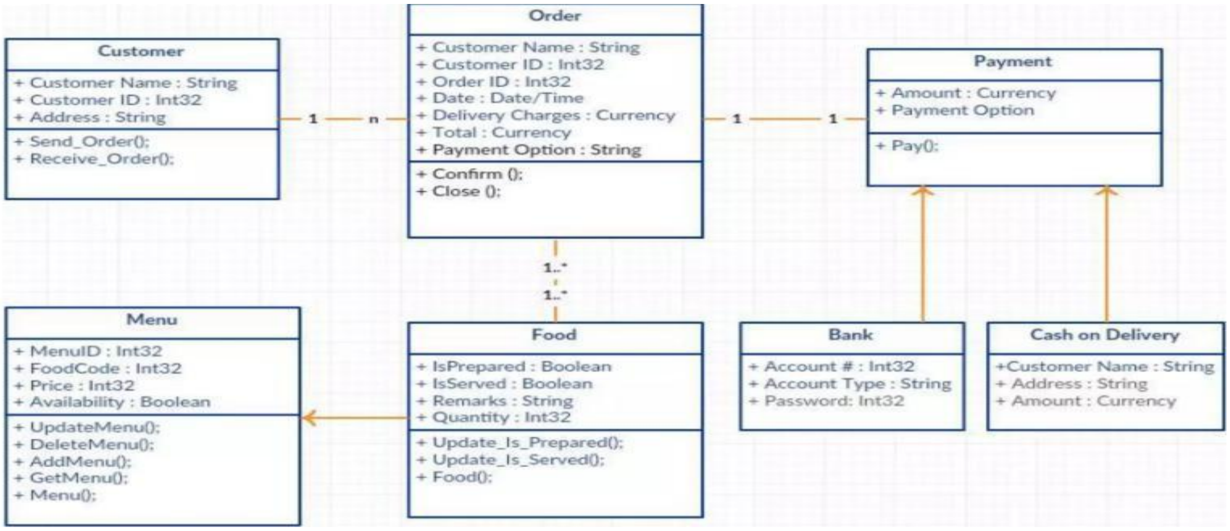


Fig 5.3.1: Class Diagram

5.4 Use-case Model Survey

The Use-Case Model Survey includes primary user actions in site:

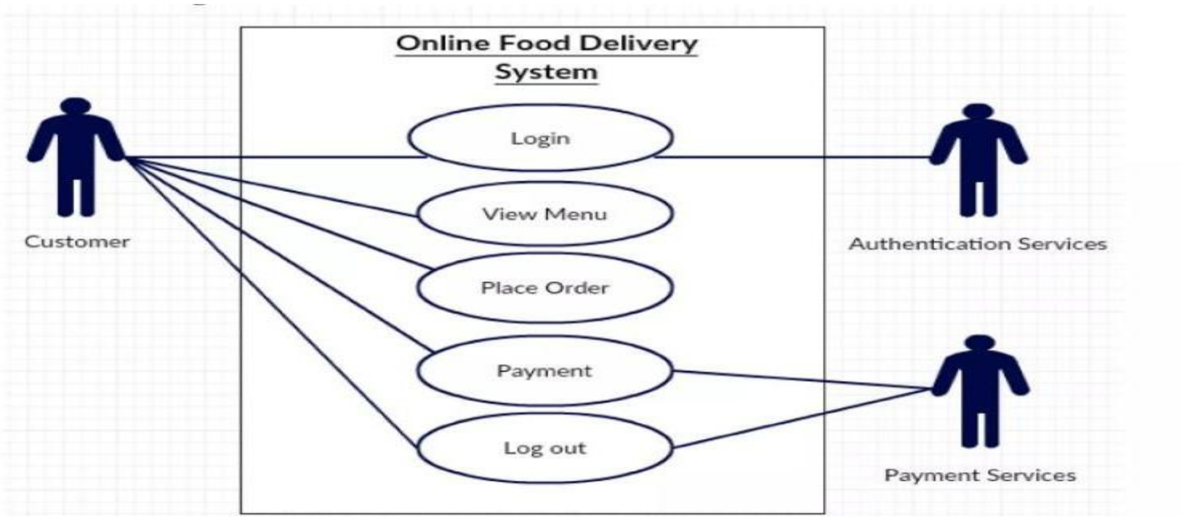
- **Login and Authentication**: Users authenticate to access features.
- **Manage Order Records**: Users add, update, and view orders.
- 

Fig 5.4.1: Use Case Diagram

5.5 Behavior Diagrams

5.5.1 Sequence Diagram

Login Process Sequence:

1. User initiates login by submitting credentials.
2. System validates credentials against the database.

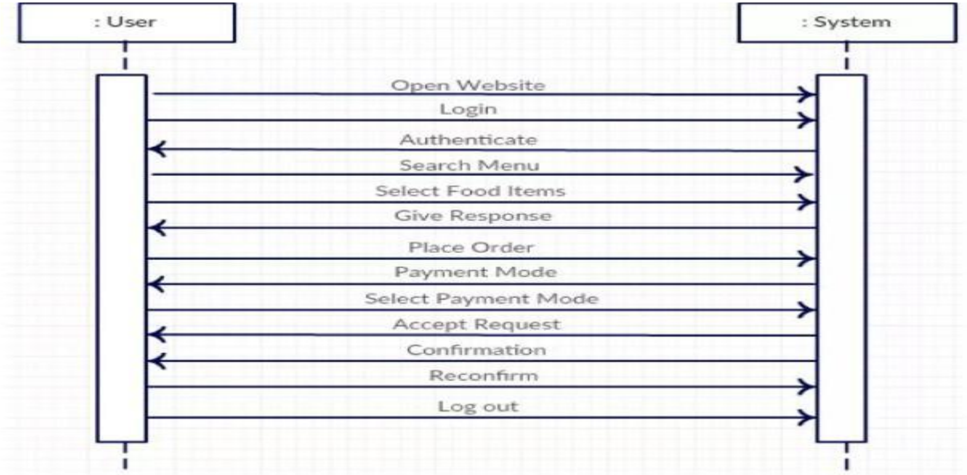


Fig 5.5.1: Sequence Diagram

➢ 5.5.2 Activity Diagram

Record Entry:

1. Start.
2. User selects "Display Menu".
3. Enter details and submit.

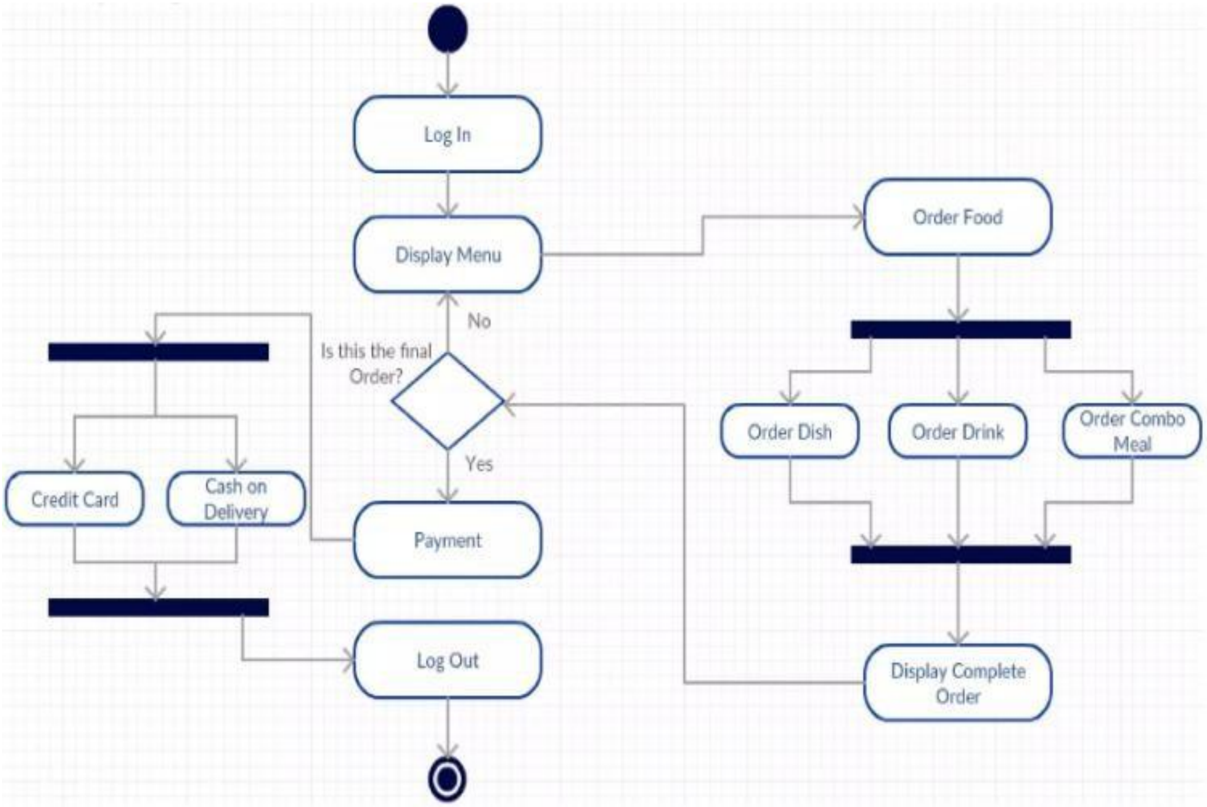


Fig5.5.2: Activity Diagram

5.6 Assumptions and Dependencies

1. Technical Feasibility Assumptions:

- **Hardware Infrastructure:** The assumption is that the necessary hardware infrastructure, including servers and network components, will be available and capable of supporting the platform's functionalities.
- **Software Dependencies:** The project assumes access to essential software components, such as Django for backend support, React for frontend development, and Redux for state management, all configured correctly to ensure smooth operation.
- **Real-time Data Integration:** Assumes the successful integration of real-time data features, enabling instant updates on order status and menu availability**.**

2. Subsystems or Component Availability:

- **Availability of APIs:** The project relies on various APIs for delivery tracking, and menu management, assuming their consistent availability and access to ensure the platform's functionality.
- **Database System:** Assumes uninterrupted access to SQLite for data storage and retrieval, facilitating real-time updates and user interactions.

3. Project-Related Assumptions:

- **User Adoption:** Assumes a favorable adoption rate among the target user base (customers and restaurant partners) for the mobile and web platforms, leading to active engagement and usage.

Data Accuracy: Assumes the accuracy and reliability of data received from user inputs, external APIs, and real-time updates from Backend.

- **4. Dependencies on External Factors:**

  - **Internet Connectivity:** Dependencies exist on stable internet connectivity for users to access the platform’s features and real-time data updates.
  - **Regulatory Compliance:** The project assumes adherence to food safety regulations and standards, including data privacy laws, to ensure compliance

6. Supporting Information

List Of Figures

Client-Server Architecture 7

Entity Relationship Diagram 10

Class Diagram 11

Use Case Diagram..12

Sequence Diagram 12

Activity Diagram 13

7.Conclusion and Future Scope

○ Conclusion

1. Summary of Achievements:

- **Accomplishments:** Summarize the successful implementations within the food delivery project.
- **Key Objectives:** Highlight how the project has addressed its main goals, such as enhancing user experience and streamlining food ordering.
- **Challenges Overcome:** Discuss challenges faced during development and the strategies used to overcome them.

2. Impact:

- **Benefits to Users:** Explain how the platform positively affects users by simplifying food ordering, improving access to various cuisines, and enhancing delivery efficiency.
- **Contribution to Local Businesses:** Discuss the project's role in supporting local restaurants and food providers by offering a digital platform for their services.
-

○ Future Scope

1. Potential Enhancements:

- **New Features:** Outline potential additional functionalities, such as personalized recommendations, loyalty programs, and real-time chat support.
- **Technological Upgrades:** Discuss opportunities for tech improvements to enhance user experience, like incorporating progressive web app features.
- **Scalability Considerations:** Address plans for scalability to accommodate a growing user base and expanding to new geographical areas.

2. **Research and Development:**

    - **AI Integration:** Explore possibilities for integrating AI algorithms for personalized recommendations and demand forecasting.

    - **Mobile App Enhancement:** Consider developing a mobile application optimized for iOS and Android to broaden accessibility.

    -

3. **Community Engagement and Partnerships:**

    - **Collaborations:** Discuss potential partnerships with local restaurants, food suppliers, and delivery services to expand offerings.

    - **Community Growth:** Highlight strategies to create a vibrant user community, including reviews, ratings, and social sharing features.

4. **Conclusion of Future Scope:**

    - **Vision and Direction:** Sum up the future direction of the food delivery project, expressing the vision for growth, improvements, and continued support for local food ecosystems.

8. Concerns / Queries / Doubts if any:

Project-related Queries:

- How can we effectively integrate real-time order tracking into our system?

- What methods can be employed to ensure seamless multi-language support in the platform?

Technological Queries:

- Are there any known challenges in implementing specific functionalities using Django ?

- How can we optimize UI development using React for web applications?

- How can we ensure compatibility and reliability while integrating various payment gateway?