"GeoHawk" - GPS BASED HAWKER MAPPING SYSTEM

A Project Report

Submitted in fulfilment of the

Requirements for the award of the degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Atharva Manjrekar

TYIT-22

Under the esteemed guidance of

Ms. Harshala Chaudhari

Head of Department

DEPARTMENT OF INFORMATION TECHNOLOGY

BHAVAN'S COLLEGE

(Affiliated to University of Mumbai)

ANDHERI (W), MUMBAI - 400058

MAHARASHTRA

YEAR 2024 - 2025

BHARTIYA VIDYA BHAVAN'S

M.M. COLLEGE OF ARTS, N.M. INSTITUTE OF SCIENCE

H.R.J. COLLEGE OF COMMERCE

BHAVAN'S COLLEGE, ANDHERI (W)

MUMBAI, MAHARASHTRA – 400 058

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled, "**GPS Based Hawker Mapping System**", is bona fide work of **Mr. Atharva Manjrekar** bearing Seat No: (**TYIT22**) submitted in fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide **Coordinator**

External Examiner

Date : **College Seal**

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PNR NO: **Roll No: TYIT22**

l. Name of the Student

Mr. Atharva Manjrekar

2. Title of the Project
GPS Based Hawker Mapping System

3. Name of the Guide
Ms. Harshala Chaudhari

4. Teaching experience of the Guide

---

5. Is this your first Submission? **Yes ☐** **No ☐**

Signature of the Student **Signature of the Guide**

Date: ________________ **Date: ______________**

Signature of the Coordinator

Date: ________________

ABSTRACT

This project focuses on creating a **location-based vendor mapping system** that allows users to discover and interact with nearby vendors using **geolocation services**. The system is designed with ease of use, ensuring that both customers and vendors benefit from its features while maintaining simplicity to suit the technical comfort level of local Indian vendors.

The **primary goal** of the application is to bridge the gap between vendors and customers by providing an easy-to-use platform that connects users with vendors based on geographic proximity. Users can **search for vendors** around their location, get directions, and make purchases directly at the vendor's shop. Unlike typical e-commerce systems, this application does not support online ordering or delivery; instead, it emphasizes physical interaction, where users travel to the vendor's location after viewing the vendor's profile, products, and reviews.

Discount Incentives are provided for customers who choose to pay through online methods such as **UPI** (Unified Payments Interface), while **cash payments** are also accepted but without any discounts. By incentivizing UPI payments, the system encourages modern, cashless transactions, while still catering to customers who prefer cash.

The project's **target audience** includes both **urban users** seeking convenience in finding nearby vendors and **local vendors** who wish to enhance their visibility and customer base without needing advanced technological knowledge. The platform also fosters the growth of **small businesses** by providing an easy-to-manage interface that enhances customer engagement through ratings and online payments.

The system is built using modern **web technologies** for the mobile app and backend. The **mobile app** handles user interactions and communicates with the backend, which processes user requests, vendor information, payments, and search functionalities. The system will also integrate with **third-party payment gateways** for UPI transactions and use **mapping APIs** for real-time location tracking and routing.

ACKNOWLEDGEMENT

I would like to extend my heartfelt thanks and deepest gratitude to my project guide, **Ms. Harshala Chaudhari**, for her invaluable guidance and support throughout the project. Her timely advice, encouragement, and insightful feedback greatly contributed to the successful completion of this work. I am truly grateful for her patience and dedication, which ensured that I stayed on track and met all deadlines.

Additionally, I would like to express my appreciation to the faculty of the Information Technology department for their constant encouragement, technical assistance, and unwavering support throughout this journey. Their contributions were essential in helping me overcome challenges and accomplish the project's objectives.

Lastly, I acknowledge the role of every teacher and staff member who provided help in shaping the project. Without their encouragement and input, this project would not have been completed to its full potential.

DECLARATION

I hereby declare that the project entitled **"GPS-Based Hawker Mapping System"**, undertaken at **Bhavans College, Andheri** in the Department of Information Technology, is my original work and has not been submitted, in part or whole, to any other university or institution for the award of any degree, diploma, or certification.

I affirm that the project is solely my own work and has been prepared to the best of my knowledge and capability. No part of this project has been plagiarized or reused from any other source for academic submission elsewhere. All efforts have been made to maintain academic integrity throughout the course of this work.

This project fulfills the **partial requirements for the degree of Bachelor of Science (Information Technology)**, submitted as a part of the final semester curriculum. I have carried out this work as per the guidelines and timelines provided by Bhavans College, ensuring that the objectives of the project were met.

Through this declaration, I acknowledge the significance of academic honesty, and I recognize that any form of misconduct, duplication, or misrepresentation would be against the values upheld by my institution. Thus, this project is a reflection of my own learning, dedication, and understanding of the subject matter as required for the completion of my degree.

Name and Signature of the Student

Table of Content

Chapter 1
Introduction

1.1 Background

The ***GPS-Based Hawker Mapping System*** is a pioneering initiative designed to tackle the persistent challenges consumers face in locating hawkers and street vendors in urban environments. With rapid urbanization and population growth, the demand for efficient, location-based services has become increasingly crucial. Street vendors, who contribute significantly to the local economy by offering affordable goods and services, often remain invisible in the digital landscape, limiting their reach and business potential. Many of these vendors operate in bustling areas, yet lack the visibility that modern consumers expect.

To bridge this gap, the project harnesses the power of GPS technology to develop a user-friendly mobile application. This platform allows consumers to effortlessly find nearby hawkers based on their preferences, such as product type, price range, and user ratings. By integrating advanced geolocation services, the application not only enhances consumer convenience but also aims to increase vendor sales and customer satisfaction. The project aligns with the growing trend of mobile application usage and fosters the growth of local businesses, thus creating a mutually beneficial scenario for both consumers and hawkers.

The historical context of street vending in India illustrates its importance in urban commerce, where it provides livelihoods for millions while meeting the everyday needs of consumers. However, as urban dynamics shift, there is an urgent need for innovative solutions that promote inclusivity in the digital realm. This project addresses these pressing issues, ensuring that local vendors are recognized and integrated into the modern marketplace, thereby contributing to a more vibrant and accessible urban economy.

1.2 Objective

The primary objectives of the **GPS-Based Hawker Mapping System** are as follows:

- **Enhance Accessibility**: The project aims to provide a seamless platform for users to locate nearby hawkers based on their geographical location. By integrating advanced GPS technology, it seeks to significantly improve accessibility, ensuring that consumers can easily discover local vendors who might have previously been out of reach.
- **Promote Local Businesses**: This initiative is designed to empower street vendors by enhancing their visibility within the digital marketplace. By featuring them prominently on the platform, the project fosters local economic growth and supports the livelihoods of hawkers, ultimately helping to sustain and rejuvenate the urban informal economy.
- **User Engagement**: Developing a user-friendly interface is crucial for the project's success. The application will include features like vendor ratings, reviews, and feedback mechanisms to encourage consumer interaction. This engagement is essential for creating a vibrant community around local vendors, enhancing customer satisfaction, and fostering loyalty.
- **Facilitate Communication**: The application will serve as a bridge for effective communication between users and vendors. By enabling direct interactions through queries and feedback options, the system aims to build trust and transparency, allowing vendors to respond to customer needs and preferences more efficiently.
- **Encourage Cashless Transactions**: Recognizing the growing trend towards digital payments, the project will incorporate secure online payment options through UPI. This feature not only promotes safer transaction methods but also aligns with modern consumer preferences, reducing the reliance on cash and enhancing the overall shopping experience.
- **Gather Analytics**: A crucial aspect of the project is the ability to collect and analyze data on consumer preferences and vendor performance. By employing analytics tools, the system can refine services, tailor offerings to user needs, and provide valuable insights to vendors, helping them improve their businesses and adapt to market trends.

1.3 Purpose, Scope and Applicability

1.3.1 Purpose

The **HawkPoint: GPS-Based Vendor Mapping** system aims to address visibility issues faced by street vendors and the convenience demands of urban consumers. As cities grow larger, street hawkers remain an essential part of urban ecosystems, offering affordable products and services. However, the challenge for these vendors lies in their limited visibility and customer reach. This project seeks to empower street vendors by providing a digital platform that makes them easily discoverable through geolocation-based search functionality.

Consumers benefit from real-time access to vendor locations, helping them find specific vendors based on proximity, product categories, or preferences. Vendors, on the other hand, can enhance their reach, compete with larger stores, and increase foot traffic. The project also integrates modern payment solutions such as UPI, encouraging digital transactions and promoting safe and cashless economies. The platform aims to offer a practical, user-friendly interface to promote local commerce and empower the unorganized retail sector.

The project supports the broader movement towards digital inclusivity, aligning with government initiatives like Digital India by enabling local street vendors—who might not be tech-savvy—to participate in the digital economy.

1.3.2 Scope

The system's scope is extensive, as it encompasses multiple layers of vendor-consumer interaction, with future expansion possibilities. It operates across the following key areas:

- **User Interface and Experience**: The system's design focuses on an easy-to-use interface. Users can log in, search for nearby vendors, filter by product categories, and receive step-by-step navigation using integrated GPS features. They also benefit from vendor ratings and reviews, offering transparency in quality.
- **Vendor Profiles**: Each vendor is mapped with geolocation data, enabling accurate and real-time discovery by users. Vendor profiles contain information on their goods or services, which are constantly updated.
- **Payment Integration**: The app integrates secure UPI payment services, offering users the option to pay digitally and avail discounts. Users paying with cash can still benefit from using the platform for discovery, but without promotional incentives.

- **Vendor Registration and Management**: Vendors can register on the platform, update their location, and communicate with consumers. Although they won't offer delivery services, they still benefit from online visibility.
- **Rating and Review System**: After visiting vendors, users can leave ratings and feedback, helping others make informed choices and providing vendors with the opportunity to improve their service.
- **Data Analytics and Insights**: The platform collects anonymized data on user preferences, search trends, and vendor performance. This data can be used to offer insights to vendors and city planners on consumer behavior and market trends.
- **Scalability**: While the platform starts in urban settings, it has the potential to expand to smaller towns, supporting diverse communities. Future versions of the app may offer advanced features such as vendor categorization, time-based search optimization, and multilingual interfaces to cater to non-English speakers in various regions.
- **Safety and Security**: User data is protected with encryption standards, and secure payments ensure financial information is not compromised.

1.3.3 Applicability

The **HawkPoint** system has extensive applicability in various domains, from consumer convenience to local economic empowerment and urban planning. Its primary applications include:

- **Urban Consumers**: Individuals living in cities can easily locate and engage with street vendors, whether they need fresh produce, snacks, or services like tailoring. The app eliminates the traditional hassle of searching for hawkers, creating convenience for users.
- **Street Vendors**: The platform is a boon for vendors who rely on foot traffic but struggle with visibility. Hawkers can increase their customer base, connect with new clients, and enhance their business through digital channels, all without investing in costly storefronts.
- **Economic Impact**: The system promotes local commerce by increasing vendor visibility and attracting more customers. This digital initiative can contribute to uplifting the unorganized sector, which forms a significant part of the Indian economy.

- **Municipal and Urban Planning**: City administrations can use the aggregated data from the system to understand hawking trends, monitor vendor locations, and even regulate hawker zoning or street markets. This helps in better urban planning and service provision.
- **Sustainability**: The platform promotes a low-resource business model, reducing the need for large storefronts or excessive infrastructure, contributing to sustainable urban growth.
- **Digital Inclusivity**: The app is an example of digital inclusivity, as it brings street vendors into the e-commerce space without requiring them to navigate complex technology. This aligns with government policies aimed at digital empowerment for all, including the unorganized sector.
- **Tourism and Hospitality**: Visitors and tourists in unfamiliar cities can use the app to find local hawkers and experience authentic street food or handicrafts, offering a cultural benefit to the platform's applicability.

1.4 Achievements

The development of the **GeoHawk: GPS-Based Vendor Mapping System** brought several valuable achievements for the student. Through the course of this project, the student gained technical knowledge in critical areas such as GPS integration, geolocation-based services, and mobile application development. They mastered creating user-friendly interfaces, learned how to manage real-time vendor data efficiently, and implemented a seamless backend system that supports user-vendor interaction.

Furthermore, this project contributed to empowering street vendors by increasing their visibility and accessibility to a larger audience. The system's ability to provide location-based mapping of vendors offers an innovative solution to the problem of finding hawkers, which has previously been largely dependent on chance. By bridging this gap between technology and the informal sector, the project promotes local businesses and enhances urban commerce in a digitally-driven world.

One of the project's standout achievements is the successful implementation of a cashless payment system via UPI, which encourages modern, secure financial transactions. This feature not only supports digital payment adoption but also ensures that users benefit from discounts when using the platform for transactions.

The goals set at the beginning of the project were not only fully achieved but, in some instances, exceeded. Besides helping users locate hawkers, the project integrated user engagement features such as vendor ratings and reviews, contributing further to consumer satisfaction and vendor accountability. Data collection for consumer preferences and vendor performance tracking also allows future optimization of services, positioning this project as a forward-thinking solution in urban commerce.

1.5 Organization

The report is systematically organized into several chapters to provide a comprehensive overview of the GPS-Based Hawker Mapping System project:

- **Chapter 1: Introduction** - This chapter sets the stage for the project by detailing the background context, objectives, and significance. It also outlines the purpose, scope, applicability, and overall achievements of the project, laying a strong foundation for the reader's understanding.
- **Chapter 2: Survey of Technologies** - In this chapter, the existing systems relevant to the project are examined, alongside the applicable technologies that will be leveraged to implement the solution. This includes an exploration of GPS technology and its integration into mobile applications.
- **Chapter 3: Requirements Gathering and Planning** - This section delves into the specifics of the problem being addressed, followed by detailed requirements specifications. It also includes planning and scheduling methodologies, software requirements necessary for development, preliminary product descriptions, and the conceptual model that guides the project's direction.
- **Chapter 4: System Design** - The design chapter outlines the architecture of the system, including basic modules, data design elements such as schema design and data integrity constraints, and procedural design encompassing logic diagrams and algorithms. Additionally, user interface design considerations and security issues are discussed, along with test case design to ensure functionality and reliability.

This structured organization enhances clarity, guiding the reader through the intricate details of the project while ensuring that each aspect is thoroughly covered for an in-depth understanding of the overall implementation and significance of the GPS-Based Hawker Mapping System.

Chapter 2
Survey Of Technologies

2.1 Existing System

The current landscape for finding street vendors primarily relies on conventional methods, where consumers physically search for shops in their vicinity. This often involves navigating through local markets or busy streets, a process that can be not only time-consuming but also frustrating and inefficient. Many consumers may remain unaware of the variety of vendors available in their area, leading to missed opportunities for both consumers and vendors alike.

The lack of an integrated application means that customers often depend on word-of-mouth recommendations or outdated printed directories, which fail to provide real-time information about vendor locations, offerings, or hours of operation. This gap in access to information hinders the consumer's ability to make informed decisions, limiting their choices and overall experience.

Moreover, many street vendors do not maintain a digital presence, which significantly diminishes their visibility and potential customer base. This absence in the digital landscape impacts their ability to attract new customers, as they are often overshadowed by established retail stores and modern shopping platforms that have embraced technology. In an age where mobile technology and instant access to information are ubiquitous, the lack of an application designed specifically for hawker mapping creates a noticeable gap in urban commerce.

This traditional approach does not align with the needs and preferences of modern consumers who prioritize convenience and efficiency in their shopping experiences. Many people now turn to mobile apps for quick solutions, seeking instant information and seamless interactions. The existing system's limitations underscore the pressing necessity for an innovative solution that leverages technology to more effectively connect consumers with street vendors. By addressing these challenges, a GPS-based hawker mapping application can enhance visibility, accessibility, and customer engagement, ultimately benefiting both consumers and vendors in the ever-evolving urban landscape.

2.2 Applicable Technologies

To address the challenges posed by the existing system, the GPS-Based Hawker Mapping System will employ several modern technologies that enhance user experience and operational efficiency:

- **GPS Technology**: Central to the application, GPS technology will provide real-time location tracking, allowing users to find nearby vendors quickly and easily. This technology will also facilitate route navigation, ensuring that consumers can efficiently reach their chosen hawker.
- **Mobile Application Development Frameworks**: The use of cross-platform frameworks like Flutter or React Native will enable the development of an application that is accessible on both Android and iOS devices. This inclusivity ensures a wider reach among potential users, catering to diverse smartphone users.
- **Backend Technologies**: A robust backend infrastructure using frameworks such as Node.js or Django will manage the data flow within the application. This includes handling user registrations, storing vendor information, processing transactions, and maintaining a secure database that protects user privacy and data integrity.
- **Geolocation APIs**: Integration with services like Google Maps API will enhance the mapping features of the application, providing accurate location data, maps, and additional contextual information about vendors.
- **Payment Gateway Integration**: The implementation of secure payment gateways, particularly those supporting UPI (Unified Payments Interface), will facilitate cashless transactions. This feature encourages users to make purchases through the app, thereby enhancing vendor sales and streamlining payment processes.
- **Database Technology**: The application will utilize relational databases such as MySQL or PostgreSQL to store and manage user, vendor, and transaction data securely. This technology will ensure efficient data retrieval and manipulation while maintaining data integrity and supporting complex queries. The choice of database technology will also facilitate scalability, enabling the application to handle an increasing number of users and transactions seamlessly.

Chapter 3
Requirement & Analysis

3.1 Problem Definition

The **GPS-Based Hawker Mapping System** addresses the difficulty consumers face in finding street vendors in urban settings, where traditional methods like word-of-mouth or walking through busy streets are inefficient and outdated. This project seeks to bridge the gap between consumers and vendors by leveraging GPS technology to provide real-time data on vendor locations, operating hours, and product availability, thus enhancing convenience and boosting vendor visibility.

Problems and Solutions:

1. **Limited Consumer Awareness**:

- **Problem**: Consumers are unaware of nearby street vendors, relying on outdated methods like word-of-mouth to discover them.
- **Solution**: The app creates an organized system, showing nearby vendors based on user location.

2. **Inefficient Search Mechanism**:

- **Problem**: Searching for vendors by physically navigating markets is time-consuming, especially in densely populated areas.
- **Solution**: Integrating GPS-based searches to locate vendors within a user's proximity, cutting down search time.

3. **Absence of Real-Time Information**:

- **Problem**: Consumers lack real-time updates on vendor availability, leading to missed opportunities.
- **Solution**: The app provides real-time data on vendor locations, operating hours, and products, ensuring up-to-date information.

4. **Vendor Visibility Challenges**:

- **Problem**: Vendors lack digital presence, which limits their ability to attract customers.
- **Solution**: The app gives vendors a platform to showcase their offerings and reach more customers.

5. **Inability to Compare Vendors**:

- **Problem**: Consumers can't compare vendors in terms of pricing, quality, or services.
- **Solution**: The app introduces a comparison feature that displays vendor ratings, prices, and customer reviews, enabling informed decisions.

6. **No Customer Loyalty Programs**:

- **Problem**: Street vendors lack loyalty programs to encourage repeat business.
- **Solution**: Implement a digital loyalty program that rewards customers for using the app for repeated purchases.

7. **Lack of Payment Flexibility**:

- **Problem**: Many vendors rely solely on cash transactions, limiting consumer options.
- **Solution**: UPI and digital wallet payment integration in the app promotes cashless transactions, increasing payment flexibility.

8. **Unorganized Vendor Operations**:

- **Problem**: Vendors often operate without consistent schedules, making them unreliable for customers.
- **Solution**: Vendors can update their working hours within the app, allowing consumers to plan visits around vendor availability.

9. **Lack of Route Assistance**:

- **Problem**: Consumers may struggle to find vendors, especially in crowded or unfamiliar areas.
- **Solution**: The app provides GPS navigation to guide users directly to the vendor's location, simplifying the search process.

10. **Vendor Language Barriers**:

- **Problem**: Some vendors face language challenges in communicating with a diverse customer base.
- **Solution**: The app can offer multilingual support, catering to users and vendors in various languages, breaking communication barriers.

Sub-Problems

1. **Limited Consumer Awareness**: Without a system in place, consumers often have no idea which vendors are nearby or what they offer.
2. **Inefficient Search Mechanism**: Manually searching for vendors by physically walking through streets can be highly inefficient, especially in densely populated areas.
3. **Absence of Real-Time Information**: Consumers don't have up-to-date knowledge about vendor availability, hours, or location.
4. **Vendor Visibility Challenges**: Vendors without a digital presence miss out on customers, limiting their ability to grow their business.
5. **Lack of Consumer Feedback System**: Without a platform for customer reviews and feedback, vendors have no structured way to improve their services or attract more customers.
6. **Limited Payment Options**: The current cash-only nature of vendor operations makes transactions inconvenient for modern users who prefer cashless, digital payments.
7. **Navigation Difficulties**: Consumers often face challenges in finding vendor locations due to the disorganized nature of street markets.

3.2 Requirement Specification

The **Requirements Specification** for the GPS-Based Hawker Mapping System describes the essential functions of the system, focusing on the system's operations and what it needs to achieve. This phase highlights the requirements without going into the technical implementation details. Key components include:

1. Functional Requirements

- **User Features**:
  - **Vendor Search**: Users can search for vendors based on proximity using GPS functionality. The app will automatically identify the user's location and suggest nearby vendors within a selected radius. This search can also be filtered by vendor type (e.g., food, clothes) or specific items (e.g., fresh fruits).

  - **Vendor Information**: Users can view vendor details, such as the type of products, pricing range, vendor location, hours of operation, and customer reviews. This includes real-time updates about vendor availability, products, and promotions.
  - **Feedback and Reviews**: Users can rate and review vendors based on their experience, helping other users make informed choices. This system will enable consumers to provide constructive feedback, enhancing the vendor's reputation and improving customer service.
  - **Digital Payments**: Integration with digital payment methods like UPI, wallets, and cards, allowing users to make seamless cashless transactions with vendors, promoting a modern, contactless payment culture.
  - **Navigation Assistance**: Users will receive step-by-step GPS directions to a vendor's location. This feature will allow users to locate vendors easily, particularly in large, crowded urban markets where finding specific vendors can be difficult.
- **Hawker Features**:
  - **Hawker Registration**: Vendors can sign up and create a digital profile in the system. They will need to provide details such as business type, location, products/services offered, and operating hours. Vendors without technical expertise can also have their profiles set up through an assisted onboarding process.
  - **Updating Availability**: Vendors can update their availability or temporarily deactivate their profile when not operating. This will help users find currently active vendors and avoid unnecessary trips.
  - **Payment Setup**: Vendors can link UPI or other digital wallets to their accounts, enabling them to accept payments digitally.
  - **Hawker Analytics**: Vendors can view basic analytics such as customer feedback, product preferences, and popular times, which will help them adjust offerings and schedules to maximize revenue.

2. Non-Functional Requirements

- **Performance**: The system should be responsive and provide real-time updates regarding vendor availability and location.
- **Security**: Both vendor and customer information should be securely handled. Secure transactions and personal data protection are essential to maintaining trust and user confidence.
- **Usability**: The app should be easy to navigate for both tech-savvy and non-tech-savvy users. This includes providing a simple UI for consumers and vendors alike.

3. System Constraints

- **Device Compatibility**: The app must function across various platforms, including Android and iOS.
- **Network Dependence**: The functionality depends heavily on internet access and reliable GPS signals for real-time data.
- **Scalability**: The system should be designed to accommodate an expanding number of vendors and users as the app grows in popularity.

3.3 Planning and Scheduling

Planning and Scheduling for the GPS-Based Hawker Mapping System is essential for organizing tasks and ensuring timely execution. The project begins with breaking down tasks into manageable objectives, considering critical factors like time management, resource allocation, and technological constraints. This structured planning ensures that each phase is clearly defined, with specific goals and deadlines. Allocating the right team members to individual tasks and ensuring the availability of necessary software tools will streamline development. Additionally, budgetary and time constraints will be monitored to keep the project on track.

Requirement Analysis:

In this phase, the focus is on identifying the needs of consumers searching for vendors and understanding vendor expectations. Integrating essential APIs like Google Maps for geolocation and UPI for payment processing is crucial. The team will conduct user surveys and vendor interviews to gather comprehensive requirements, which will help clarify the app's features. A key milestone will be the finalization of requirement specifications, ensuring both user and technical needs are clearly defined.

System Design:

System design involves creating both user interfaces (UI) and backend structures. The goal is to develop an intuitive UI/UX for consumers while maintaining a straightforward interface for vendors. This phase includes designing database schemas to manage vendor data, such as locations and ratings, as well as developing algorithms for proximity-based vendor searches. Key tasks involve creating wireframes and documenting the system architecture, with design approvals being a critical milestone before moving to development.

Development:

Development is organized into frontend and backend tasks, with work structured into manageable sprints. The frontend focuses on building a mobile-friendly interface, including vendor listings and maps. Meanwhile, the backend will establish a scalable database and integrate geolocation services, creating APIs for real-time interactions. Each module is assigned a sprint of 2-3 weeks, ensuring timely completion. Key milestones include the successful development of each module within its designated sprint.

Testing:

The testing phase encompasses unit testing, integration testing, and user acceptance testing (UAT). Unit testing verifies that each module functions correctly in isolation, while integration testing ensures that all system components work together seamlessly. UAT involves real users to validate that the app meets their needs effectively. Test cases will be developed for each module, with a major milestone being the successful passage of all tests before proceeding to deployment.

Deployment:

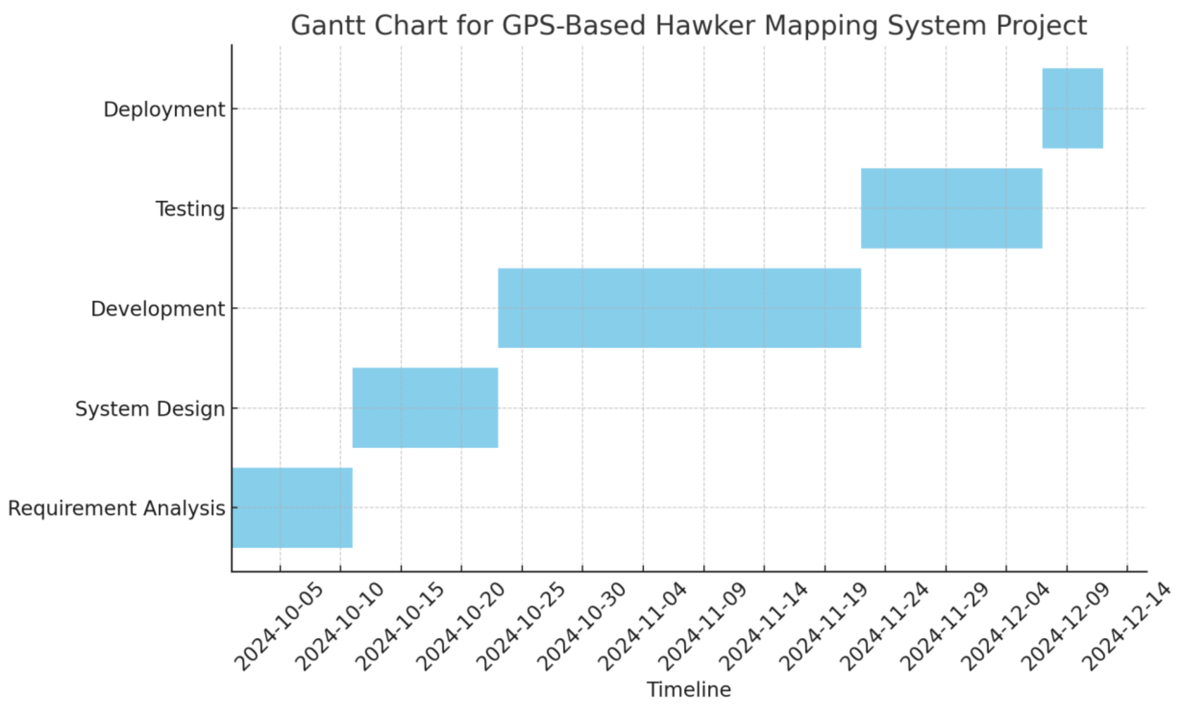
After testing is complete, the app will be deployed on a live server, followed by post-deployment monitoring to identify and fix any issues. This phase includes setting up the cloud infrastructure and monitoring app performance in real-world conditions. Continuous oversight will help address any bugs that arise after launch. The final milestone will be the successful launch of the application on major app stores, transitioning the project from development to active use.

The scheduling phase looks at the availability of resources, human expertise, and tools. Key resources include development team members, software tools (e.g., for coding, database management), testing platforms, and cloud infrastructure.

You can illustrate this through a **Gantt Chart**. This chart will visually represent tasks over time, showing the start and end dates of each task, dependencies between tasks, and resource allocation.

Example of Task Breakdown:

- Requirements Gathering: 2 weeks
- UI/UX Design: 3 weeks
- Backend Development: 4 weeks
- Frontend Development: 4 weeks
- Testing: 2 weeks
- Deployment: 1 week



3.4 Software Requirements

Software Requirements

- **Mobile Application Development**
  - **Development Framework:** React Native or Flutter for cross-platform support.
  - **Database:** Firebase Firestore for real-time data storage or PostgreSQL for relational data management.

- **Backend Development**
  - **Server:** Node.js or Python (Django/Flask) for handling API requests.
  - **Geolocation API:** Google Maps API or OpenStreetMap for mapping and location services.
- **Payment Gateway Integration**
  - **Payment Processing:** Razorpay or Paytm for handling UPI transactions.
- **Development Tools**
  - **IDE:** Visual Studio Code for coding.
  - **Version Control:** Git for managing code versions.
  - **Project Management:** Trello or Jira for task management.
- **Operating System**
  - Development will primarily be done on Windows 10 Pro or macOS for cross-platform compatibility.

3.5 Preliminary Product Description

- **Product Name**: GPS-Based Vendor Mapping Application
- **Purpose**: To provide users with an easy-to-use mobile application that helps them locate nearby vendors based on their geolocation, enabling efficient shopping without the need for delivery services.

Key Features

The GPS-Based Vendor Mapping Application offers a range of key features designed to enhance user experience and convenience. **User Registration and Authentication** ensures a secure environment where users can easily create accounts and log in. The application utilizes **Geolocation Services** to pinpoint the user's current location, enabling them to discover nearby vendors on an interactive map. Users can seamlessly engage in **Vendor Search and Filtering**, allowing them to explore various categories and apply filters based on distance and vendor ratings. To assist users in reaching their desired vendors, the app provides **Route Generation**, offering turn-by-turn navigation. Payment options are flexible, supporting **UPI Payments** that come with attractive discounts for online transactions, while cash payments are also accepted without discounts. Additionally, users can interact with vendors through a **Vendor Interaction** feature, enabling them to rate and provide feedback on their shopping experiences. Finally, the application boasts a **User-Friendly Interface**, designed to be intuitive and accessible for all users, including those who may not be tech-savvy.

Target Audience

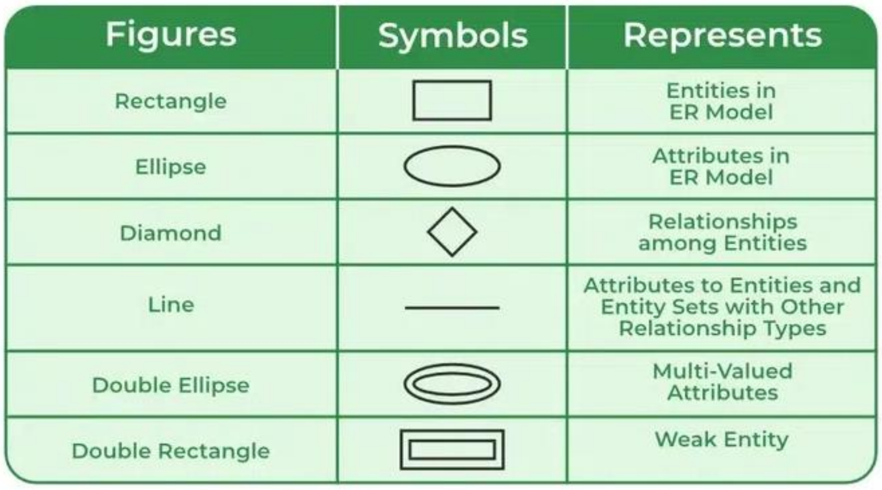
- Individuals looking for local vendors for various products and services.
- Vendors who want to increase their visibility and attract more customers.

3.6 Conceptual Models

ER Diagram

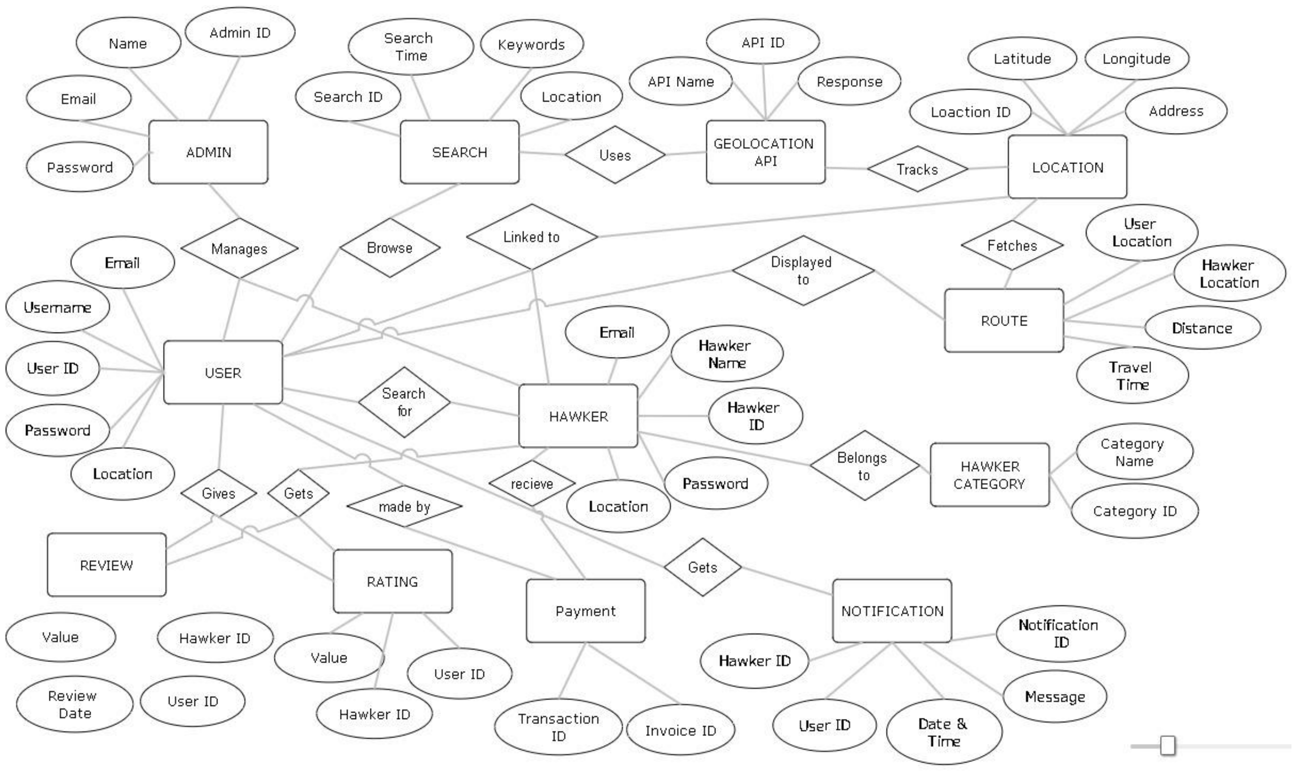
An Entity-Relationship (ER) diagram is a graphical representation of a system's data model, showcasing the entities involved, their attributes, and the relationships among them. It acts as a blueprint for database design, helping to clarify how data is structured and organized. In an ER diagram, entities are depicted as rectangles, attributes as ovals, and relationships as diamonds or lines connecting the entities. This diagram is instrumental in identifying the data needs of a system and ensures that the database design is normalized, which reduces redundancy and enhances data integrity. ER diagrams play a vital role in database design and development, aiding communication between developers and stakeholders.

Notations in ER diagram :



Figures	Symbols	Represents
Rectangle	▭	Entities in ER Model
Ellipse	⬭	Attributes in ER Model
Diamond	◇	Relationships among Entities
Line	—	Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse	◎	Multi-Valued Attributes
Double Rectangle	▣	Weak Entity

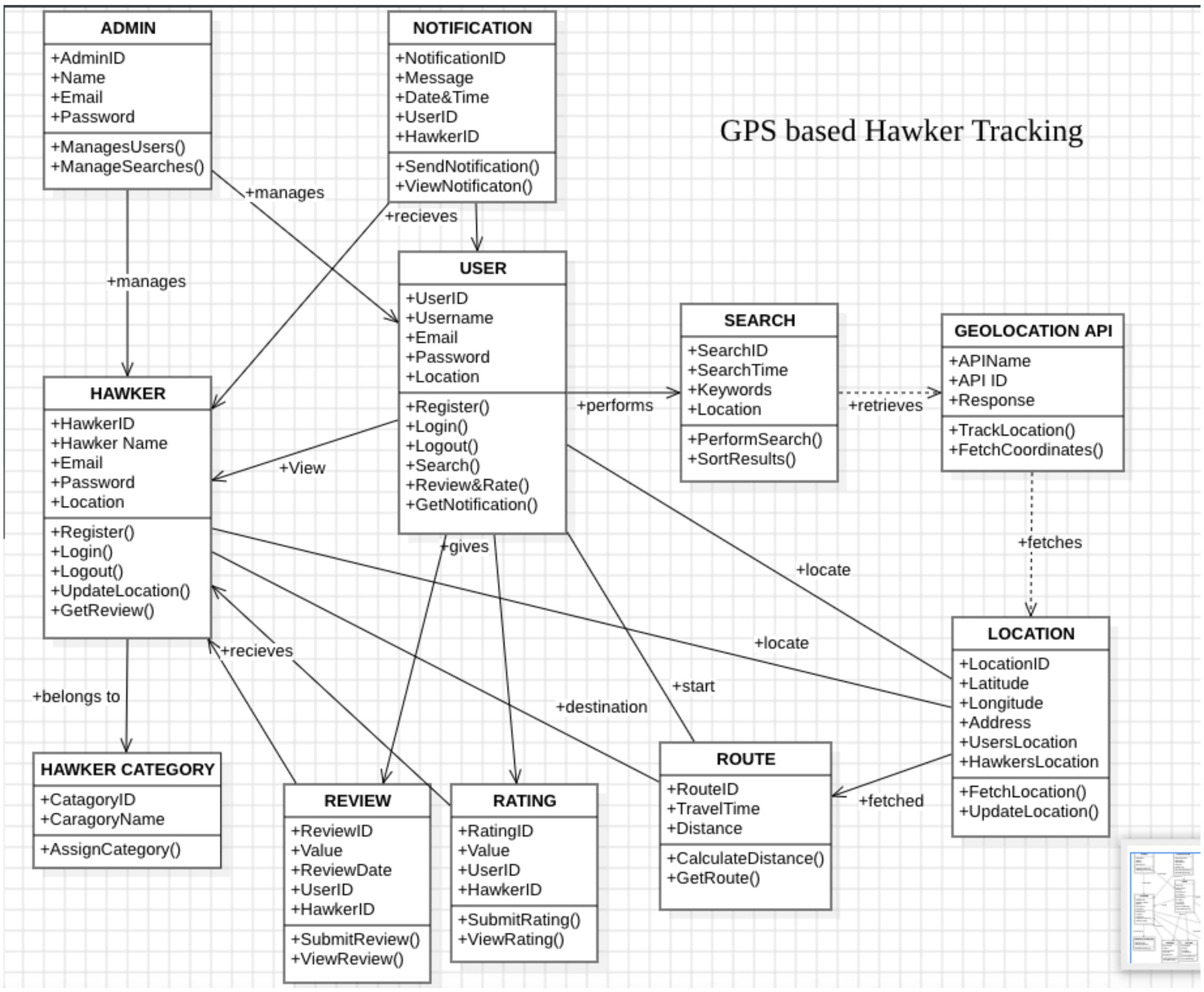
Diagram :



Class Diagram

A class diagram is a static structure diagram in Unified Modeling Language (UML) that illustrates the classes, their attributes, methods, and the relationships among them within a system. Represented as rectangles, classes are divided into three sections: the class name, attributes (which describe the properties of the class), and methods (which define the functions or operations that can be performed). Relationships between classes include association (showing connections with multiplicity), inheritance (indicated by a solid line with a hollow triangle), aggregation (depicted with a hollow diamond), and composition (represented with a filled diamond). Class diagrams serve as blueprints for development, providing a clear structure that guides implementation, while also acting as documentation to help stakeholders understand the system's architecture. They play a crucial role in object-oriented design by facilitating analysis and ensuring that all components work together cohesively.

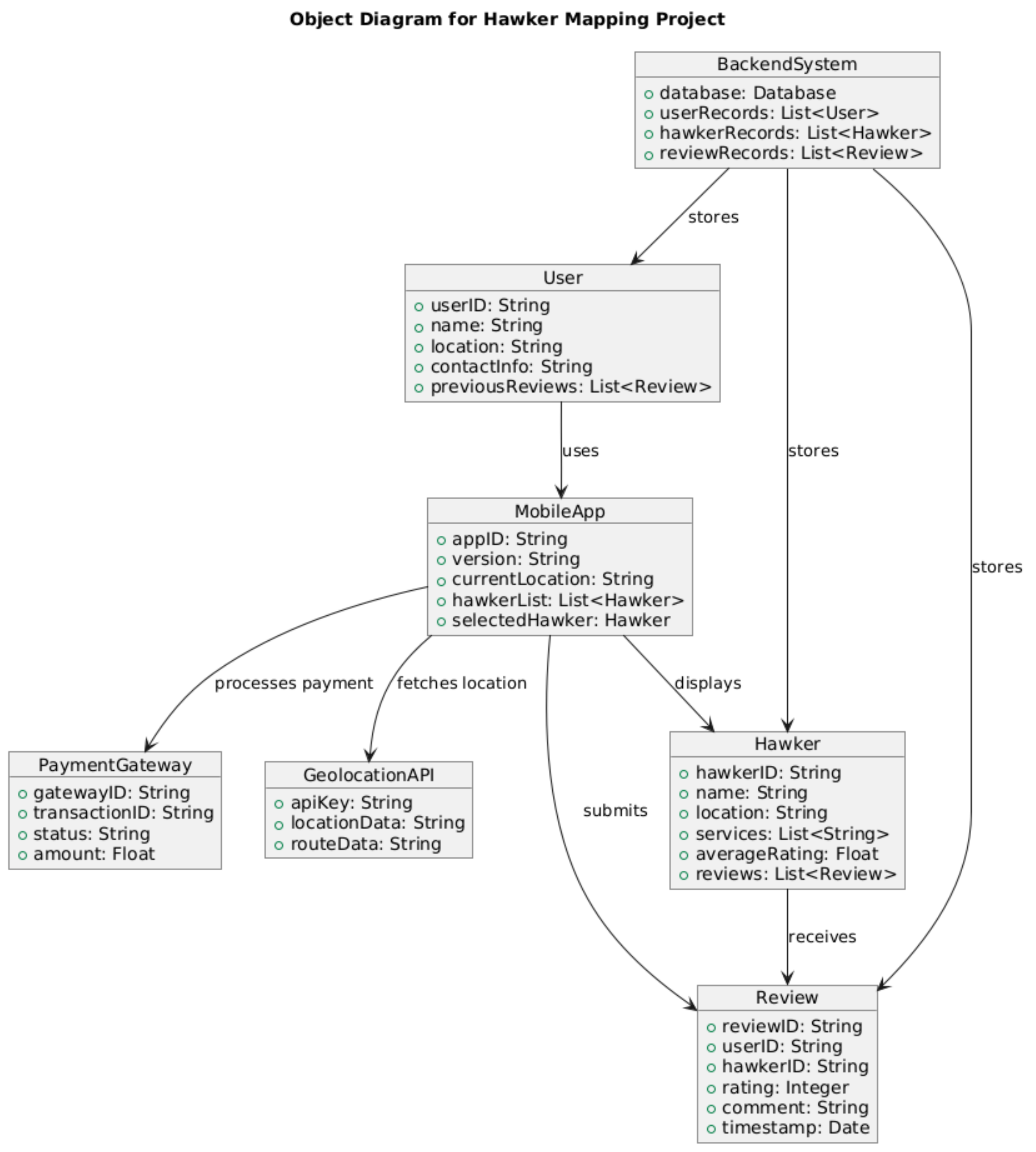
Diagram :



Object Diagram

An object diagram is a static structure diagram in Unified Modeling Language (UML) that provides a snapshot of the instances (objects) of classes at a specific point in time. It displays how these objects relate to one another, along with the current values of their attributes. Objects are represented as rectangles with underlined names, while their attributes are shown in the format attributeName: value. Links connecting the objects illustrate their relationships, such as associations, aggregation, or composition. Object diagrams serve multiple purposes, including offering a detailed representation of how objects interact and facilitating communication among developers and stakeholders. By showcasing concrete examples of classes in action, object diagrams are valuable for analyzing system behavior and testing processes.

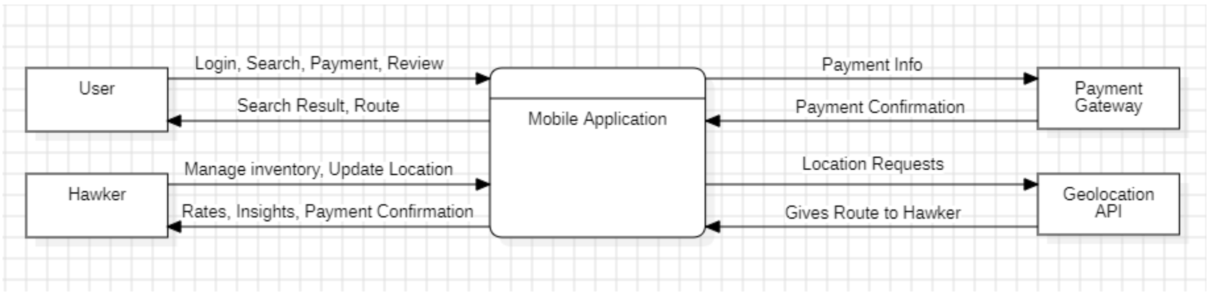
Diagram :



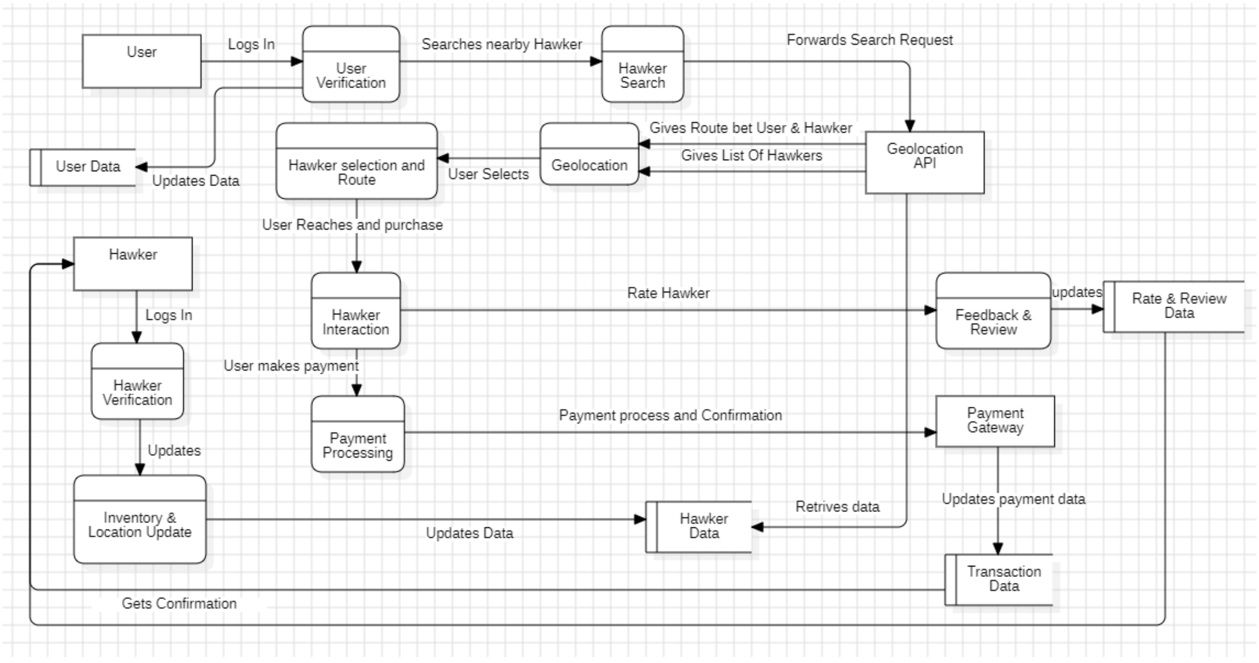
Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation that illustrates the flow of data within a system. It shows how data is processed by the system and the pathways it takes from input to output. DFDs consist of four main components: processes (which transform data), data stores (where data is stored), external entities (sources or destinations of data), and data flows (the arrows indicating the movement of data between components). DFDs help in understanding how information moves through a system, making them useful for analyzing system requirements and improving system design. They are valuable tools for stakeholders to visualize data interactions, identify inefficiencies, and ensure that the system meets business needs.

Level 0:



Level 1:



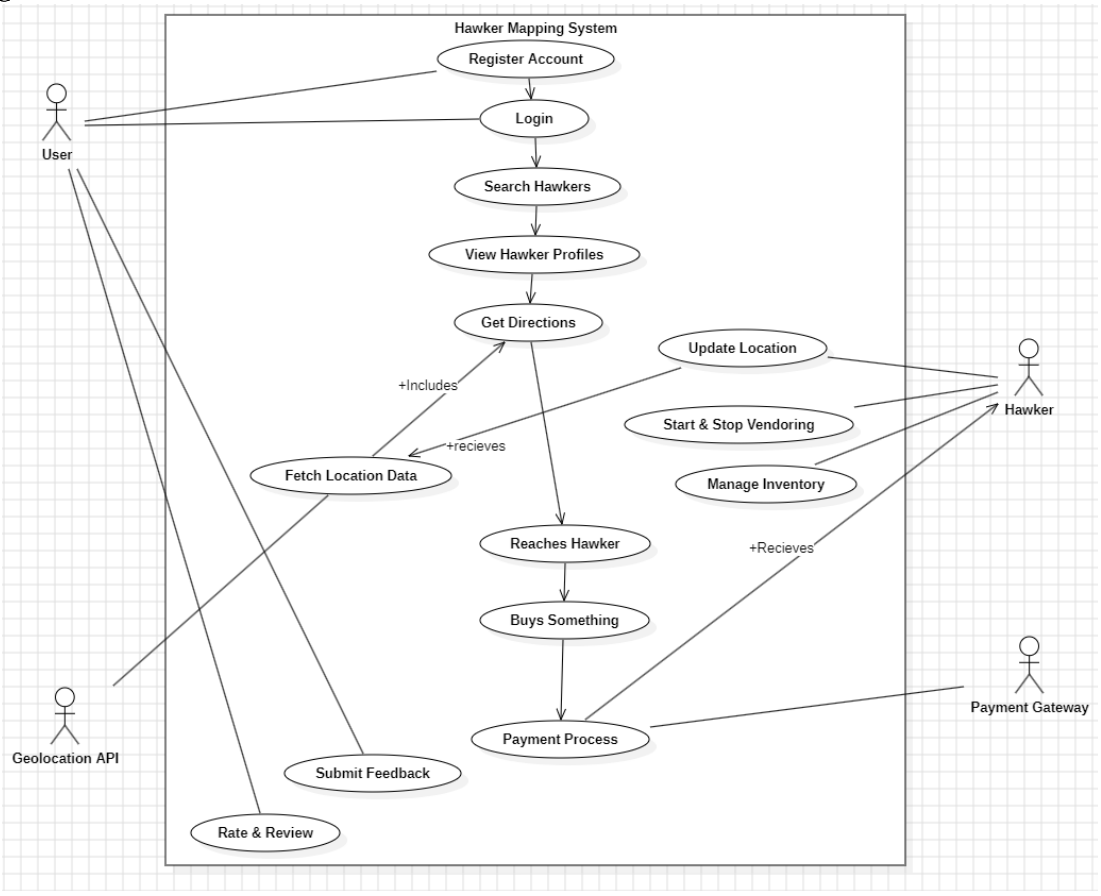
Use Case Diagram

A use case diagram is a visual representation of the interactions between users (actors) and a system, illustrating the functional requirements of the system. It captures the various ways users can interact with the system through specific use cases, which represent particular functionalities or services offered by the system. In a use case diagram, actors are depicted as stick figures, use cases as ovals, and the relationships between them are shown with lines. This diagram helps stakeholders understand the system's scope, identify user needs, and clarify the functionality that must be implemented. Use case diagrams are valuable for gathering requirements, guiding development, and ensuring that the system meets user expectations.

Notations used in Use Case :

Notation	Meaning
Use Case	Use case: An Ellipse represents a single Use Case. The name of the use case is written inside ellipse.
Actor	Actor: Represents every element which interact with the system. It may be a user a another system which interact with the system described in the use case.
—	Ordinary Relationship: Represents a connection that is a channel for the transfer of information between a use case and an actor or between use cases.
System boundary	System Boundary: A square represents the boundary of the system. Inside the system are the use cases and outside the square are the actors who interact with the system.

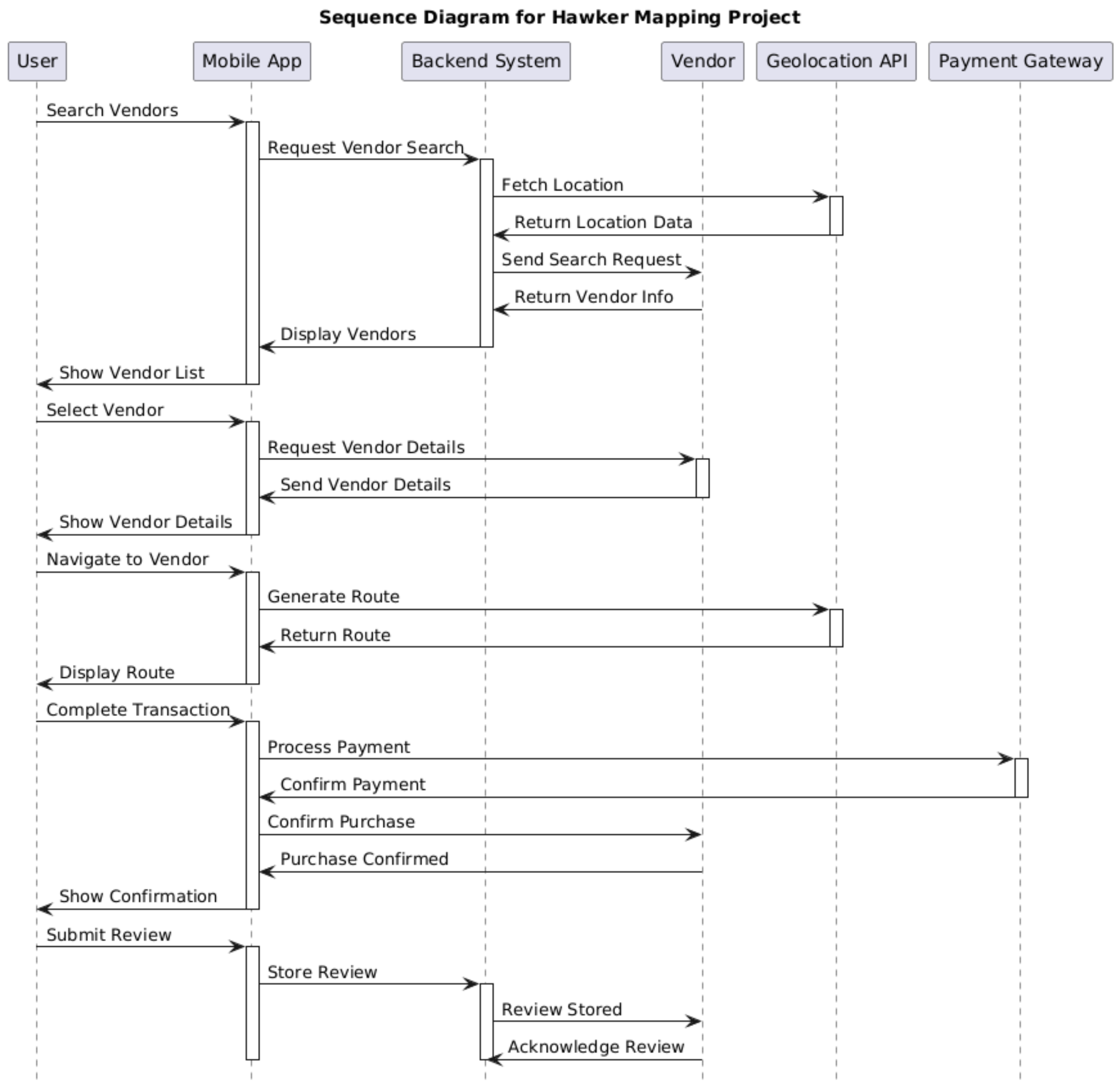
Diagram :



Sequence Diagram

A sequence diagram is an interaction diagram in Unified Modeling Language (UML) that illustrates how objects interact in a specific sequence over time. It features lifelines, which represent the presence of objects, and activation boxes that indicate when an object is active. Messages, shown as arrows connecting lifelines, represent communication between objects, including synchronous and asynchronous calls. Sequence diagrams provide a clear visualization of object interactions, helping to document use cases and identify the responsibilities of different objects in a system. They are valuable tools for modeling dynamic behavior, enhancing understanding and communication among stakeholders.

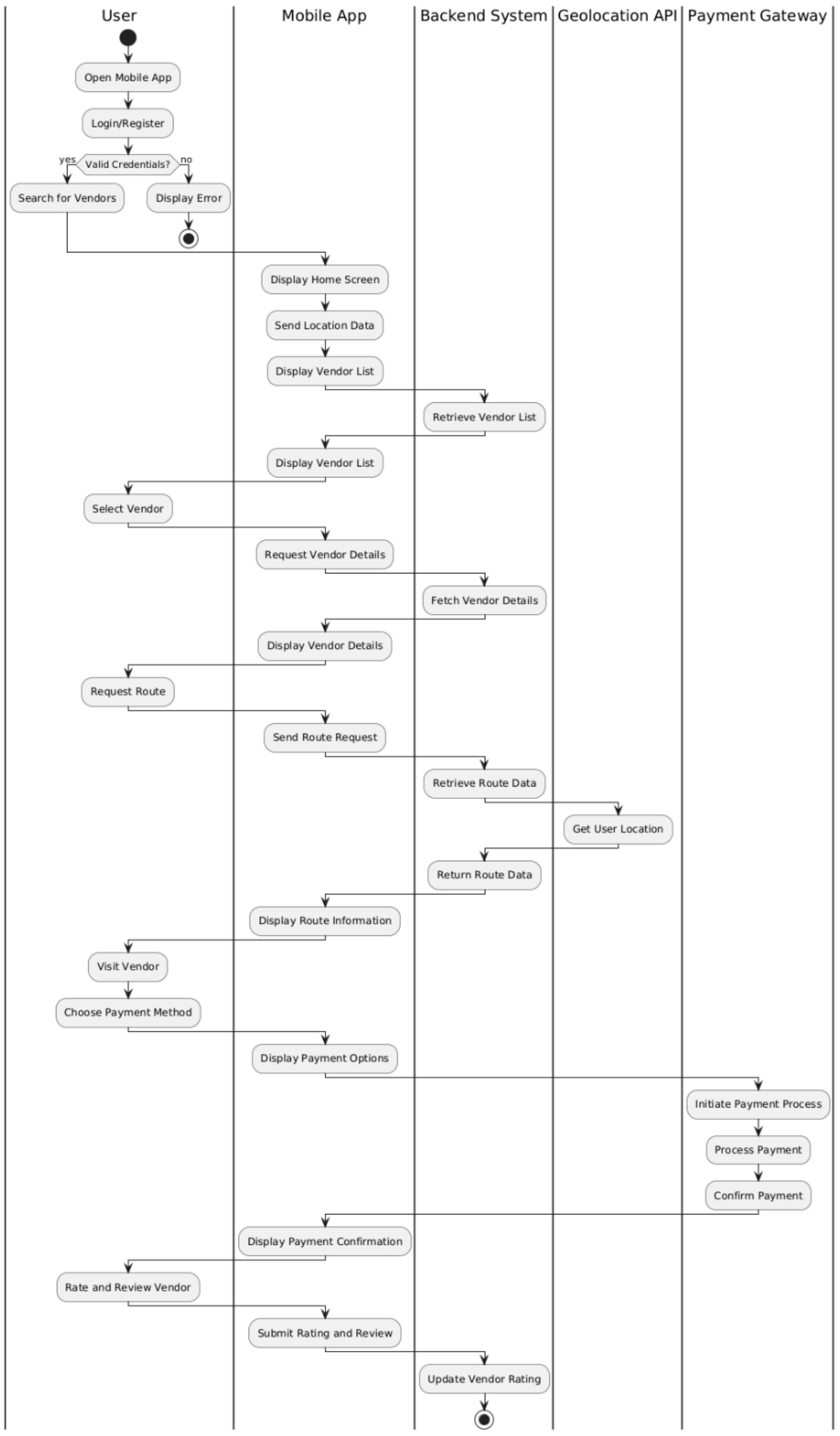
Diagram :



Activity Diagram

An activity diagram is a UML behavioral diagram that shows the flow of control or data in a system. It outlines the sequence of activities, decisions, and parallel processes in a workflow using symbols like rounded rectangles for activities and diamonds for decisions. Activity diagrams are useful for modeling business processes and complex algorithms, offering a clear visual representation that enhances understanding and communication among stakeholders.

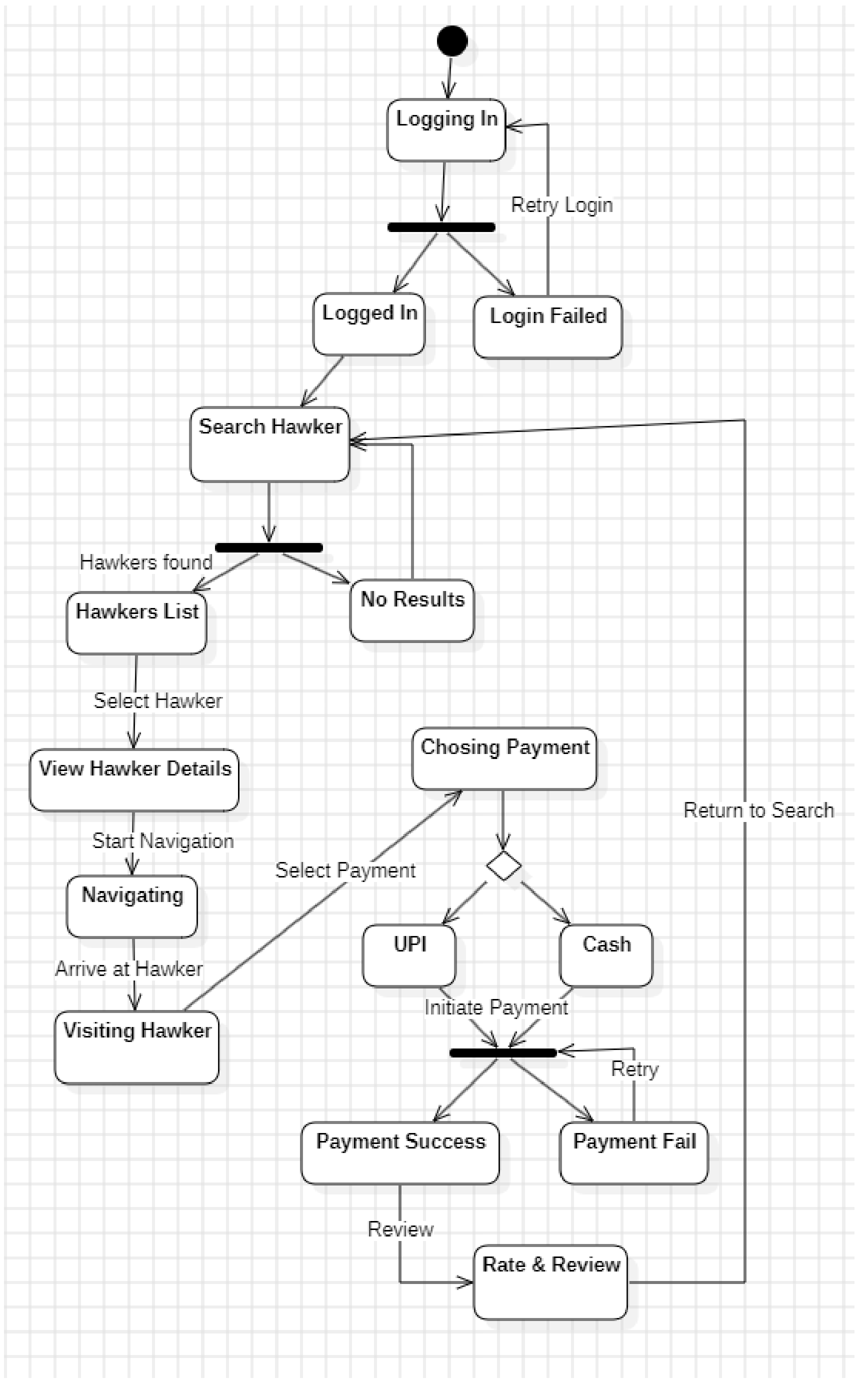
Diagram :



State Transition Diagram

A state transition diagram is a type of behavioral diagram in Unified Modeling Language (UML) that illustrates the states of an object and the transitions between those states. Each state represents a specific condition or situation of an object, while transitions are triggered by events, causing the object to change from one state to another. The diagram includes states represented as rounded rectangles and transitions depicted as arrows connecting these states. State transition diagrams are valuable for modeling the lifecycle of objects, especially in systems where behavior is dependent on state changes, such as embedded systems or user interfaces. They help clarify how objects respond to various events, facilitating better design and implementation.

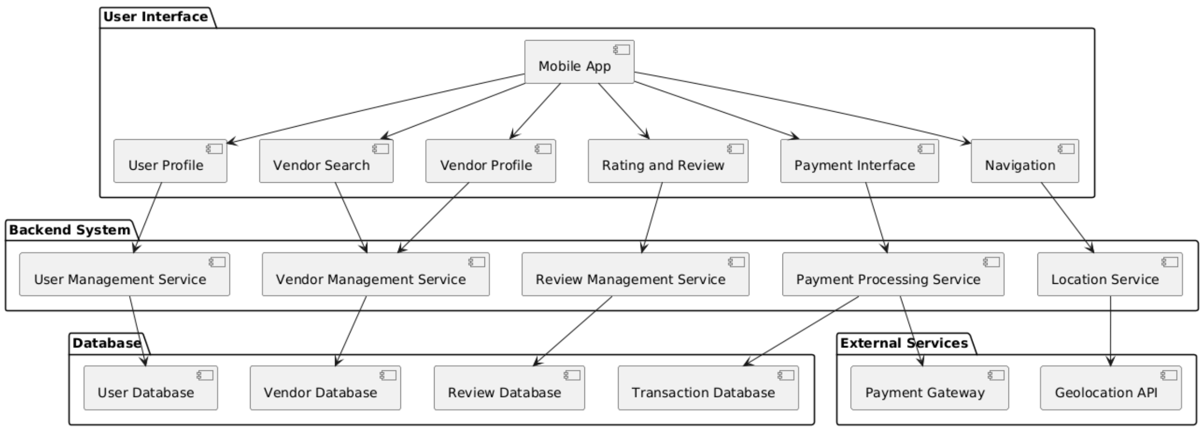
Diagram :



Component Diagram

A component diagram is a structural diagram in Unified Modeling Language (UML) that depicts the components of a system and their relationships. Components are represented as rectangles with the component name, and interfaces are shown as circles or lollipops. The diagram illustrates how different components interact with each other and with external systems through interfaces. Component diagrams are useful for visualizing the modular structure of a system, identifying dependencies, and facilitating the design of complex systems by breaking them down into manageable parts. They enhance communication among developers and stakeholders by providing a high-level overview of the system architecture and its components.

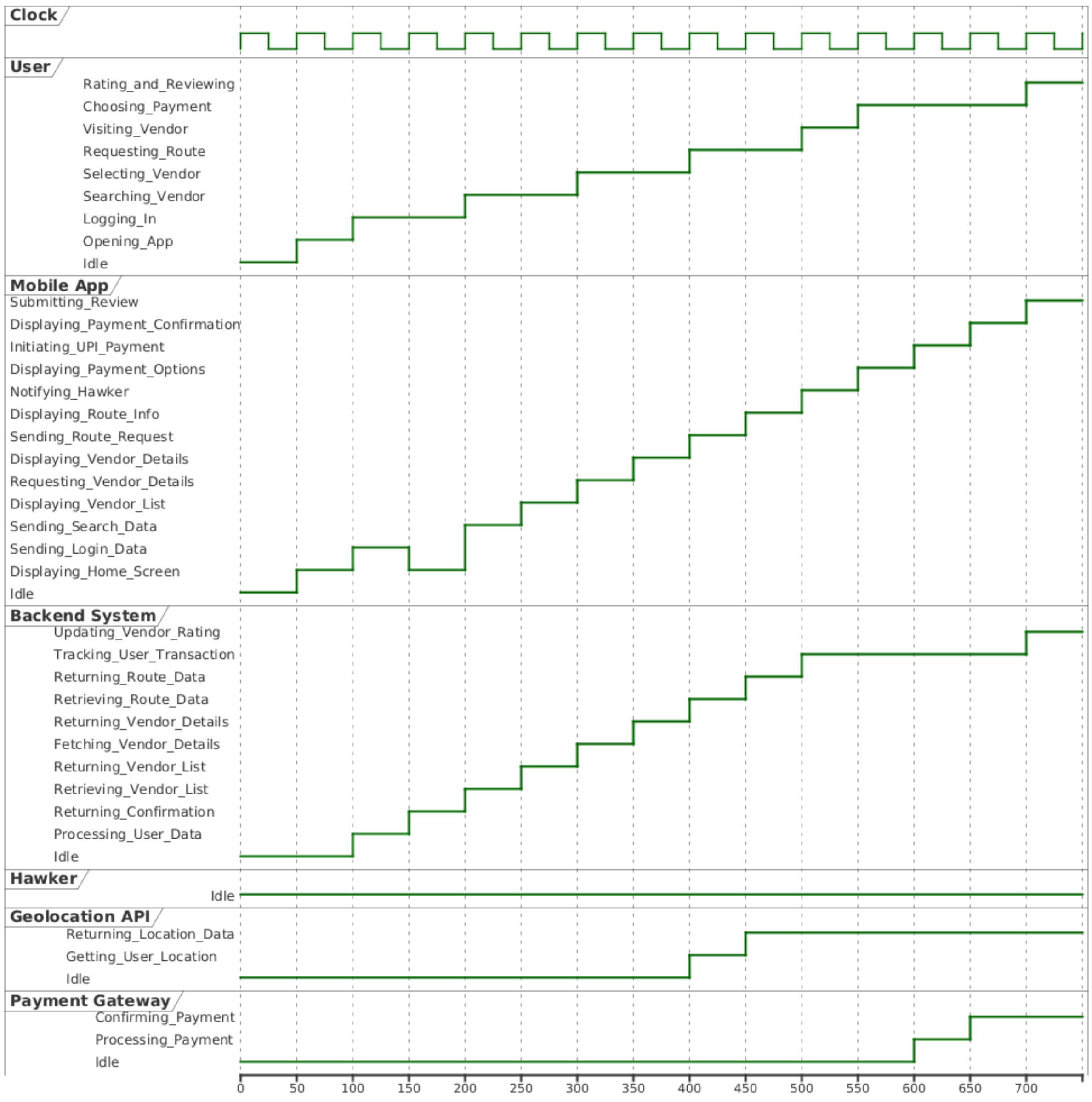
Diagram :



Timing Diagram

A timing diagram is a type of interaction diagram in Unified Modeling Language (UML) that shows the behavior of objects over time, focusing on the timing constraints of messages exchanged between them. It represents the state of one or more objects at specific times and how those states change in response to events. Timing diagrams use a time axis to display the duration and timing of messages, making them particularly useful for real-time systems and protocols. By illustrating the timing relationships and constraints between events, timing diagrams help in understanding the temporal dynamics of interactions, facilitating better design and analysis of systems that require precise timing control.

Diagram :



Chapter 4
System Design

4.1 Basic Modules

User Management Module

- **Functionality**: Manages user registration, authentication, and account settings.
- **Components**:
    - **User Registration Form**: Interface for new users to create an account with fields for name, email, password, and mobile number.
    - **Login Interface**: Allows users to log in using their credentials (email and password).
    - **Password Recovery System**: Enables users to recover forgotten passwords via email or SMS verification.
    - **Profile Management**: Allows users to update their profile information, including contact details and preferences.

Geolocation Module

- **Functionality**: Utilizes GPS to determine and display the user's current location.
- **Components**:
    - **Location Service**: Integrates with device GPS to fetch real-time location data.
    - **Map Integration**: Utilizes mapping APIs (e.g., Google Maps) to display the user's location and nearby vendors.
    - **Geofencing**: Defines geographical boundaries to trigger notifications or features when users enter specific areas.

Vendor Management Module

- **Functionality**: Manages vendor registration and profile information.
- **Components**:
    - **Vendor Registration Form**: Interface for vendors to create and manage their profiles, including name, location, and services offered.
    - **Vendor Dashboard**: Provides vendors with insights into their profiles, including customer ratings and feedback.
    - **Vendor Verification System**: Confirms vendor authenticity through document uploads and verification processes.

Search and Filtering Module

- **Functionality**: Allows users to search for vendors based on various criteria.
- **Components**:
    - **Search Bar**: Interface for users to input search queries (e.g., vendor name, category).
    - **Filter Options**: Options for users to filter results based on distance, ratings, or vendor types.
    - **Search Results Display**: Renders a list or map of vendors that match the search criteria, showing key information like ratings and distance.

Navigation Module

- **Functionality**: Provides turn-by-turn navigation to selected vendors.
- **Components**:
    - **Route Calculation Service**: Computes the best route from the user's location to the vendor using mapping APIs.
    - **Navigation Interface**: Displays step-by-step directions on the map with voice guidance options.
    - **Real-Time Traffic Updates**: Integrates traffic data to provide users with the most efficient route.

Payment Module

- **Functionality**: Manages payment processing and options for users.
- **Components**:
    - **Payment Gateway Integration**: Connects to UPI and other payment systems for secure transactions.
    - **Transaction Confirmation System**: Sends notifications to users upon successful payment, including receipts and transaction details.
    - **Discount Management**: Applies discounts for online payments and manages cash transaction records.

Feedback and Rating Module

- **Functionality**: Collects user ratings and feedback for vendors.
- **Components**:
    - **Rating Interface**: Allows users to rate vendors on a scale (e.g., 1-5 stars).
    - **Feedback Form**: Enables users to leave detailed comments about their experiences.

    - **Rating Aggregator**: Calculates average ratings for vendors and displays them on vendor profiles.

User Interface Module

- **Functionality**: Manages the overall look and feel of the application.
- **Components**:
    - **Navigation Menus**: Provides users with easy access to different sections of the app (e.g., search, profile, payment).
    - **Responsive Layouts**: Ensures the application is accessible on various devices (smartphones, tablets).
    - **User Interaction Elements**: Includes buttons, sliders, and forms for user engagement.

Database Management Module

- **Functionality**: Handles data storage, retrieval, and management for the application.
- **Components**:
    - **Database Connection Management**: Establishes and manages connections to the database (e.g., MySQL, MongoDB).
    - **Data Schema Definition**: Defines the structure of tables and relationships in the database.
    - **CRUD Operations**: Implements Create, Read, Update, and Delete operations for user, vendor, payment, and feedback data.
    - **Data Integrity and Constraints**: Ensures data validity and consistency through constraints like primary keys, foreign keys, and unique constraints.
    - **Backup and Recovery**: Implements mechanisms for data backup and recovery to prevent data loss.

4.2 Data Design

4.2.1 Schema Design

Schema Design refers to the organization and structure of a database, which defines how data is stored, organized, and accessed. In your project, it involves creating a logical structure that outlines the different data entities, their attributes, and the relationships between them. Proper schema design ensures data integrity, efficiency in data retrieval, and easy maintenance of the database.

Usage in Your Project

1. **Organize Data**: Define how user data, hawker information, payment records, and other entities are organized.
2. **Enable Relationships**: Establish relationships between different entities (e.g., users and their payments, hawkers and their categories).
3. **Facilitate Queries**: Optimize the structure for efficient querying and retrieval of data, enhancing user experience.
4. **Ensure Data Integrity**: Implement constraints and validations to maintain the accuracy and consistency of data.

Schemas in Your Project

1. User Schema
2. Hawker Schema
3. Payment Schema
4. Feedback and Rating Schema
5. Geolocation Schema
6. Notification Schema
7. Route Schema
8. Admin Schema
9. Hawker Category Schema

These schemas will form the backbone of your application's database, supporting its functionality and ensuring that all data is appropriately managed. Let me know if you need more details on any specific schema!

4.2.2 Data Integrity and Constraints

1. User Table
Purpose: Stores user information and authentication details.

Field Name	**Data Type**	**Null**	**Key**
userId	String (UUID)	NOT NULL	Primary Key
name	String	NOT NULL	
email	String	NOT NULL	Unique
password	String (hashed)	NOT NULL	
mobile	String (optional)	NULL	
createdAt	Date	NOT NULL	
updatedAt	Date	NOT NULL	

2. Hawker Table

Purpose: Stores information about vendors (hawkers) available in the application.

Field Name	**Data Type**	**Null**	**Key**
hawkerId	String (UUID)	NOT NULL	Primary Key
name	String	NOT NULL	Unique
category	String	NOT NULL	
location	Object	NOT NULL	
address	String	NOT NULL	
rating	Float	NULL	
reviewCount	Integer	NOT NULL	Default: 0
createdAt	Date	NOT NULL	
updatedAt	Date	NOT NULL	

3. Hawker Category Table

Purpose: Defines categories for hawkers to facilitate filtering and searching.

Field Name	**Data Type**	**Null**	**Key**
categoryId	String (UUID)	NOT NULL	Primary Key
name	String	NOT NULL	Unique
description	String	NULL	
createdAt	Date	NOT NULL	
updatedAt	Date	NOT NULL	

4. Payment Table

Purpose: Manages payment transactions made by users for hawker purchases.

Field Name	**Data Type**	**Null**	**Key**
paymentId	String (UUID)	NOT NULL	Primary Key
userId	String (UUID)	NOT NULL	Foreign Key
hawkerId	String (UUID)	NOT NULL	Foreign Key
amount	Float	NOT NULL	
paymentMethod	String	NOT NULL	
status	String	NOT NULL	
timestamp	Date	NOT NULL	

5. Feedback and Rating Table

Purpose: Allows users to provide ratings and feedback for hawkers based on their experience.

Field Name	Data Type	Null	Key
feedbackId	String (UUID)	NOT NULL	Primary Key
userId	String (UUID)	NOT NULL	Foreign Key
hawkerId	String (UUID)	NOT NULL	Foreign Key
rating	Integer (1-5)	NOT NULL	CHECK
comment	String	NULL	
createdAt	Date	NOT NULL	

6. Route Table

Purpose: Stores route information for users navigating to hawker locations.

Field Name	Data Type	Null	Key
routeId	String (UUID)	NOT NULL	Primary Key
userId	String (UUID)	NOT NULL	Foreign Key
hawkerId	String (UUID)	NOT NULL	Foreign Key
startLocation	Object	NOT NULL	
endLocation	Object	NOT NULL	
directions	Array of Strings	NOT NULL	
distance	Float	NOT NULL	
duration	String	NOT NULL	
createdAt	Date	NOT NULL	
updatedAt	Date	NOT NULL	

7. Geolocation Table

Purpose: Handles geolocation data related to users and hawkers.

Field Name	**Data Type**	**Null**	**Key**
locationId	String (UUID)	NOT NULL	Primary Key
latitude	Float	NOT NULL	
longitude	Float	NOT NULL	
address	String	NOT NULL	
userId	String (optional, UUID)	NULL	Foreign Key
hawkerId	String (optional, UUID)	NULL	Foreign Key
createdAt	Date	NOT NULL	
updatedAt	Date	NOT NULL	

8. Notification Table

Purpose: Manages notifications sent to users regarding their activities or updates in the app.

Field Name	**Data Type**	**Null**	**Key**
notificationId	String (UUID)	NOT NULL	Primary Key
userId	String (UUID)	NOT NULL	Foreign Key
message	String	NOT NULL	
type	String	NOT NULL	
status	String	NOT NULL	
createdAt	Date	NOT NULL	
updatedAt	Date	NOT NULL	

9. Admin Table

Purpose: Manages administrative access and functions within the application.

Field Name	**Data Type**	**Null**	**Key**
adminId	String (UUID)	NOT NULL	Primary Key
username	String (unique)	NOT NULL	Unique
password	String (hashed)	NOT NULL	
role	String	NOT NULL	
createdAt	Date	NOT NULL	
updatedAt	Date	NOT NULL	

4.3 Procedural Design

Procedural design is a structured approach in software development that focuses on breaking down a program's tasks into smaller, well-defined procedures or functions, promoting modularity and clarity. Each procedure encapsulates specific functionality, making the code reusable and easier to maintain. Control structures, such as loops and conditional statements, guide the flow of execution, while parameters allow for flexible data handling. This methodology encourages clear documentation and enhances error handling, leading to robust programs that can be tested at the procedure level. Overall, procedural design fosters organized, efficient coding practices that simplify debugging and improve collaboration among developers

4.3.1 Logic Diagram

Logic diagrams are abstract representations used to illustrate the logical structure and relationships between components of a system or process. These diagrams focus on the conceptual flow of data or control, showing how different parts of a system interact without emphasizing the physical aspects (such as hardware or implementation details).

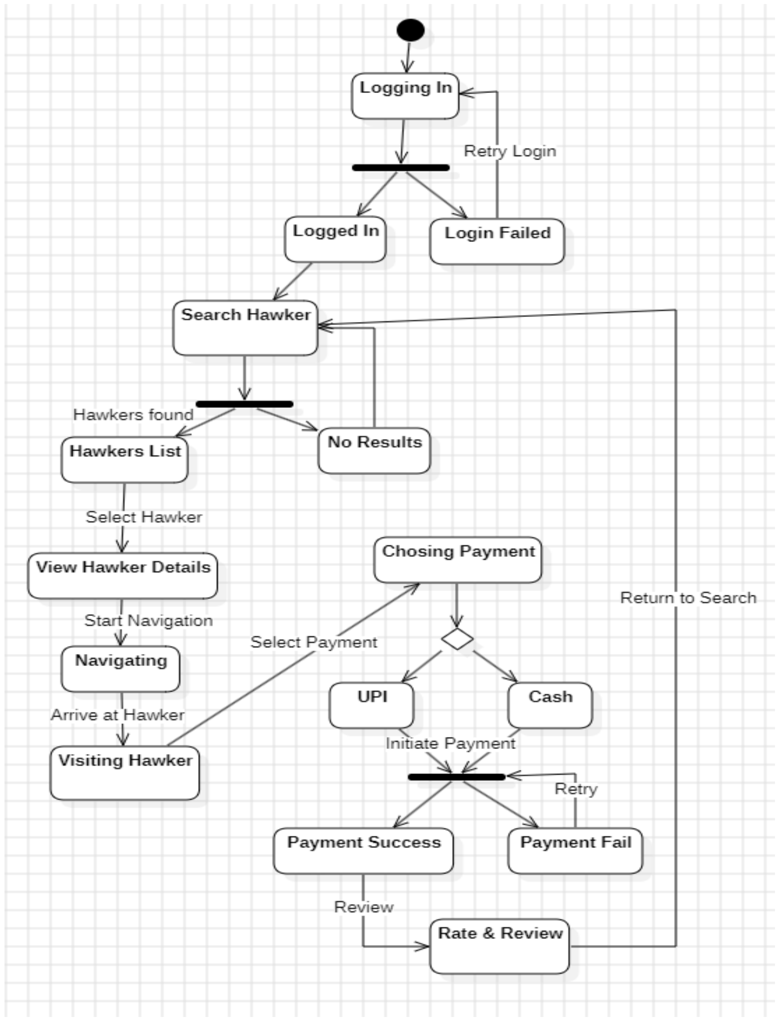


Figure 4.1 Logic Diagram

4.3.2 Data Structures

various data structures are used for efficient data storage and manipulation. Here's how they are applied:

1. **Arrays (Lists in Python):**
    - A collection of vendor information, such as ratings, distances from users, or a list of services offered.
    - Example: A list of vendor names and their corresponding distances used for sorting vendors based on proximity to the user.
2. **Dictionaries (Key-Value Pairs):**
    - Used to store structured data about vendors and users, such as vendor details, ratings, and reviews.
    - Example: { "vendor_name": "Vendor A", "rating": 4.5, "location": [latitude, longitude] } for storing vendor profiles and related details.
3. **Tables (Relational Databases):**
    - Organized data in the backend that holds vendor profiles, user accounts, reviews, and transaction records.
    - Example: A Vendors table containing fields like vendor name, location, service type, and average rating.
4. **Files:**
    - External files for backup purposes or logging important data like user interactions, payments, or historical review data.
    - Example: A CSV file storing all transactions or interactions with vendors for record-keeping.

These data structures help streamline operations like vendor searches, user navigation, and review management in your project.

4.3.3 Algorithms Design

Geolocation and Proximity Search Algorithm :

Purpose:

To find vendors near the user's current location and display them on the map, sorted by proximity.

Input Data:

- User's current location (latitude, longitude) from GPS.
- List of vendors, each with location coordinates (latitude, longitude).

Output Data:

- List of nearby vendors, sorted by distance.

Algorithm:

1. **Input:** User's current GPS location (lat_user, long_user) and a list of vendors with coordinates (lat_vendor, long_vendor).
2. **Calculate Distance:** For each vendor, calculate the distance using the **Haversine formula**:

$$\text{Average Rating} = \frac{\sum \text{Rating values}}{\text{Number of ratings}}$$

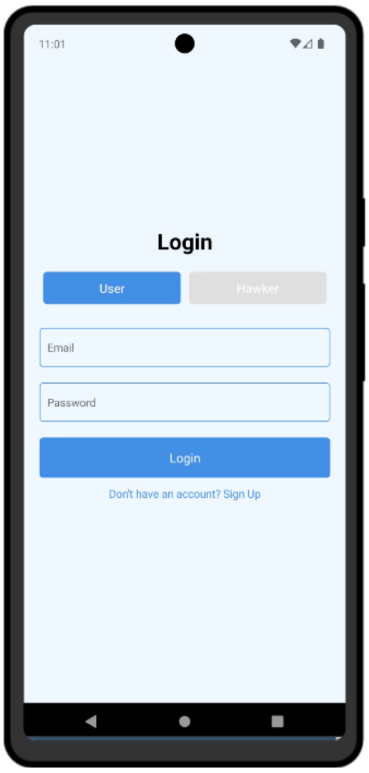
   - Where r is the radius of the Earth (6371 km).
   - $\Delta\varphi$ is the difference in latitudes, and $\Delta\lambda$ is the difference in longitudes.

3. **Store Results:** Store the calculated distance for each vendor.
4. **Sort Vendors:** Sort the vendors based on the calculated distance in ascending order.
5. **Output:** Display the sorted list of vendors along with their distance on the app.

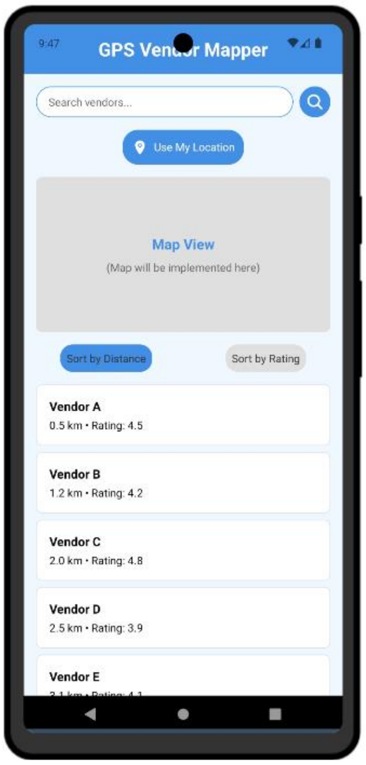
Logic Explanation:

- **Haversine formula** is used to compute the shortest distance between two points on the Earth's surface, taking into account the curvature of the Earth.
- Sorting helps in listing vendors nearest to the user.

4.4 User Interface Design



Login/SignUp Page



Vendor A
0.5 km • Rating: 4.5

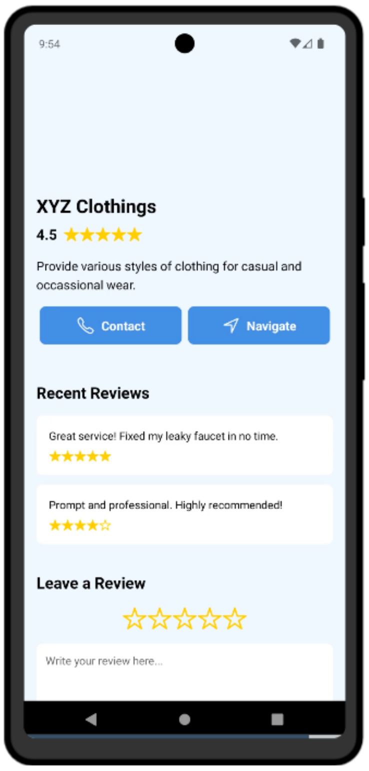
Vendor B
1.2 km • Rating: 4.2

Vendor C
2.0 km • Rating: 4.8

Vendor D
2.5 km • Rating: 3.9

Vendor E
3.1 km • Rating: 4.1

Home Page

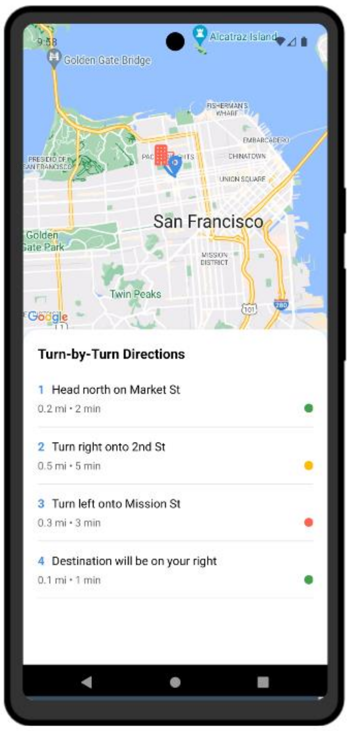


XYZ Clothings
4.5 ★★★★★
Provide various styles of clothing for casual and occassional wear.

Great service! Fixed my leaky faucet in no time. ★★★★★

Prompt and professional. Highly recommended! ★★★★☆

Hawker Profile Page



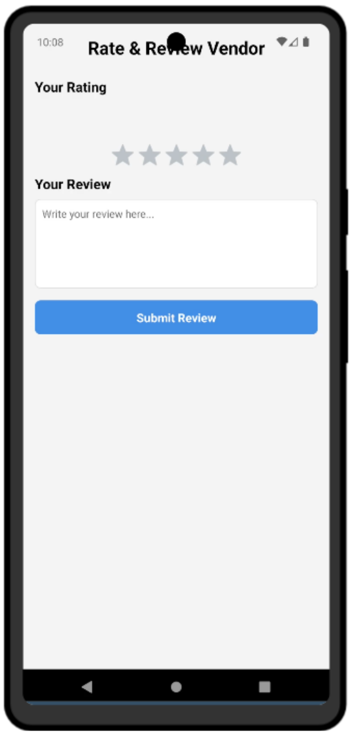
1 Head north on Market St
0.2 mi • 2 min

2 Turn right onto 2nd St
0.5 mi • 5 min

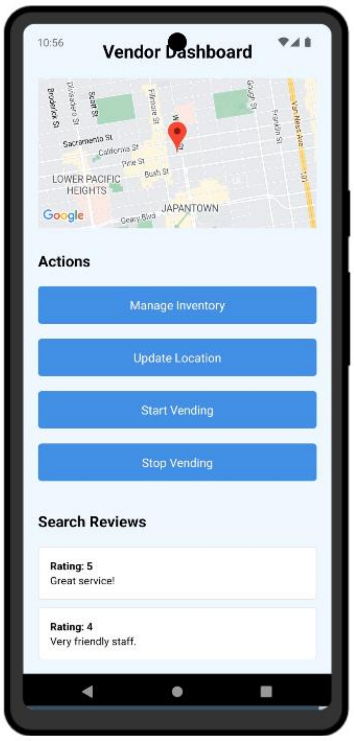
3 Turn left onto Mission St
0.3 mi • 3 min

4 Destination will be on your right
0.1 mi • 1 min

Navigation Screen Page



Rate & Review Page



Rating: 5
Great service!

Rating: 4
Very friendly staff.

Hawkers Dashboard

4.5 Security Issue

Real-Time Considerations

1. **Data Privacy**: Users' locations and personal information must be protected to prevent unauthorized access and misuse. Any real-time data transmission should use secure channels.
2. **User Authentication**: Ensuring that only authorized users (both hawkers and customers) can access specific features of the app is critical. This includes robust methods of verifying user identities.
3. **Data Integrity**: Information sent between users and the backend must remain unaltered during transmission. This requires the use of hashing and encryption methods.
4. **Session Management**: Proper management of user sessions is essential to prevent session hijacking or unauthorized access. Sessions should time out after periods of inactivity.

Security Issues

1. **Location Tracking**: Continuous location tracking raises privacy concerns. Users should be informed about data collection, and they should have the option to enable/disable location tracking.
2. **Payment Security**: Handling of payment transactions requires strict security measures to protect users' financial information. This includes implementing secure payment gateways.
3. **Vulnerability to Attacks**: The application is at risk for various cyber threats such as DDoS attacks, SQL injection, and cross-site scripting (XSS). Regular security audits and penetration testing should be conducted to identify and mitigate these risks.
4. **Insecure Data Storage**: Sensitive information should not be stored in plain text. Instead, strong encryption methods must be applied to protect user data in the database.

Avoiding Security Problems

1. **Use of HTTPS**: All data transmitted between the client and server should occur over HTTPS to secure data in transit.
2. **Implement Strong Authentication Mechanisms**: Use multi-factor authentication (MFA) to verify users. Ensure that passwords are hashed using strong algorithms.
3. **Data Encryption**: Implement end-to-end encryption for sensitive information, particularly for payment details and personal user data.
4. **Regular Updates and Patching**: Keep all software libraries and frameworks updated to protect against known vulnerabilities.

5. **User Awareness**: Educate users on security best practices, such as recognizing phishing attempts and managing app permissions effectively.

Security Policy Plans and Architecture

1. **Access Control Policies**: Define who can access what features within the application. Implement role-based access control (RBAC) for different user types (customers and vendors).
2. **Data Protection Policy**: Develop policies for how data is collected, processed, stored, and deleted, ensuring compliance with data protection regulations like GDPR or CCPA.
3. **Incident Response Plan**: Establish a plan for responding to security breaches, including notification procedures for affected users and regulatory bodies.
4. **Monitoring and Logging**: Implement monitoring solutions to detect and respond to unusual activity in real time. Maintain logs for auditing purposes.
5. **Security Architecture**: Design the application with security in mind, ensuring that security features are integrated throughout the architecture. This includes using secure coding practices, protecting APIs, and using firewalls to guard against unauthorized access.

4.6 Test Case Design

Login Page Test Cases :

Test Case ID	Description	Input	Expected Output	Actual Output
TC-LG-01	Successful login with valid credentials	Email: user@example.com, Password: Password123	User is logged in and redirected to home screen	User successfully logged in and redirected
TC-LG-02	Invalid email format	Email: userexample.com, Password: Password123	Error: 'Please enter a valid email address.'	Error message displayed correctly
TC-LG-03	Incorrect password	Email: user@example.com, Password: Wrong Password	Error: 'Invalid email or password.'	Error message displayed correctly
TC-LG-04	Attempt to login with empty fields	Email: (empty), Password: (empty)	Error: 'Please fill in both email and password.'	Error message displayed correctly
TC-LG-05	Login attempt with unregistered email	Email: nonexistent@example.com, Password: Password123	Error: 'Email not found. Please register first.'	Error message displayed correctly

Sign-Up Page Test Cases :

Test Case ID	Description	Input	Expected Output	Actual Output
TC-SU-01	Successful signup with valid details	Name: John Doe, Email: john@example.com, Password: Password123, Mobile: 9876543210	User is registered and redirected to login screen	User successfully registered and redirected
TC-SU-02	Weak password validation	Name: John Doe, Email: john@example.com, Password: 123, Mobile: 9876543210	Error: 'Password must be at least 8 characters long.'	Error message displayed correctly
TC-SU-03	Attempt to register with an already registered email	Name: John Doe, Email: existing@example.com, Password: Password123, Mobile: 9876543210	Error: 'Email is already registered. Please log in.'	Error message displayed correctly
TC-SU-04	Invalid email format during signup	Name: John Doe, Email: johnexample.com, Password: Password123, Mobile: 9876543210	Error: 'Please enter a valid email address.'	Error message displayed correctly
TC-SU-05	Missing mobile number	Name: John Doe, Email: john@example.com, Password: Password123, Mobile: (empty)	Error: 'Please enter your mobile number.'	Error message displayed correctly

Home Page Test Cases :

Test Case ID	Description	Input	Expected Output	Actual Output
TC-HP-01	Verify the Search Bar functionality	Enter vendor name or category in search bar	Display list of vendors matching the search query	As expected
TC-HP-02	Verify 'Current Location' button functionality	Click on 'Use My Location' button	User's current location is fetched and displayed on the map	As expected
TC-HP-03	Verify Map View loads nearby vendors	Open the Home Page	Map shows markers for nearby vendors	As expected
TC-HP-04	Verify the filter functionality by distance	Select 'Sort by Distance' filter	Vendors are sorted by increasing distance from the user location	As expected
TC-HP-05	Verify the vendor list displays correct details	View vendor list	Vendors names, distance, and rating are displayed correctly	As expected

Hawkers Profile Display Test Cases :

Test Case ID	Description	Input	Expected Output	Actual Output
TC-HPF-01	Verify hawker's details (name, image, services) are displayed	Open the Hawker Profile page	Hawker name, image, and services should be correctly displayed	As expected
TC-HPF-02	Verify rating and reviews are displayed correctly	View Hawker Profile page	Average rating and user reviews are displayed with correct details	As expected
TC-HPF-03	Verify 'Contact Hawker' button functionality	Click on 'Contact' button	Hawker contact information (call or message) is initiated successfully	As expected
TC-HPF-04	Verify 'Rating' interface functionality	Select stars for rating	User is able to select star rating and submit the rating successfully	As expected
TC-HPF-05	Verify 'Navigate' button functionality	Click on 'Navigate' button	Navigation to the hawker location is started on the map	As expected

Navigation Screen Test Cases :

Test Case ID	Description	Input	Expected Output	Actual Output
TC-NS-01	Verify map view is displayed	Open the Navigation Screen	The map view should be displayed showing the user's location	As expected
TC-NS-02	Verify the route from user location to vendor location	Select vendor and start navigation	The correct route from user location to the vendor location is displayed	As expected
TC-NS-03	Verify turn-by-turn directions functionality	Follow directions after starting navigation	The turn-by-turn directions are displayed correctly with visual indicators	As expected
TC-NS-04	Verify real-time traffic updates overlay	Check for traffic updates on the map	Real-time traffic updates are shown on the map as an overlay	As expected
TC-NS-05	Verify that location recalculates correctly if user changes route	Deviate from the suggested route	The route recalculates and adjusts based on the new user location	As expected

Rate & Review Page Test Cases :

Test Case ID	Description	Input	Expected Output	Actual Output
TC-RR-01	Verify the rating interface is displayed	Open the Rate and Review Page	The rating interface (stars) should be visible	As expected
TC-RR-02	Verify user can select a rating	Select a rating (e.g., 4 stars)	The selected rating is highlighted, and the value is stored	As expected
TC-RR-03	Verify user can enter feedback text	Enter feedback text in the comment box	The feedback text is successfully entered and displayed	As expected
TC-RR-04	Verify submission of a rating and feedback	Click the Submit button after entering details	A success message is shown, and the rating and feedback are saved in the database	As expected
TC-RR-05	Verify that previous ratings and comments are displayed	Open the Rate and Review Page for a vendor	The list of previous ratings and comments for that vendor is displayed	As expected

Hawkers Dashboard Test Cases :

Test Case ID	Description	Input	Expected Output	Actual Output
TC-HD-01	Verify the dashboard displays correct hawker information	Open Hawker's Dashboard	Hawker's name, location, and status (active/inactive) displayed correctly	As expected
TC-HD-02	Verify the "Manage Inventory" functionality	Click on "Manage Inventory"	Inventory management screen is displayed, allowing item updates	As expected
TC-HD-03	Verify "Start Vendoring" button functionality	Click on "Start Vendoring" button	Hawker status changes to "Active," location is updated on the map	As expected
TC-HD-04	Verify "Stop Vendoring" button functionality	Click on "Stop Vendoring" button	Hawker status changes to "Inactive," location is hidden from users	As expected
TC-HD-05	Verify reviews and ratings are displayed correctly	Open "Reviews & Ratings" section	List of user reviews and ratings is displayed correctly	As expected