# Chapter 3

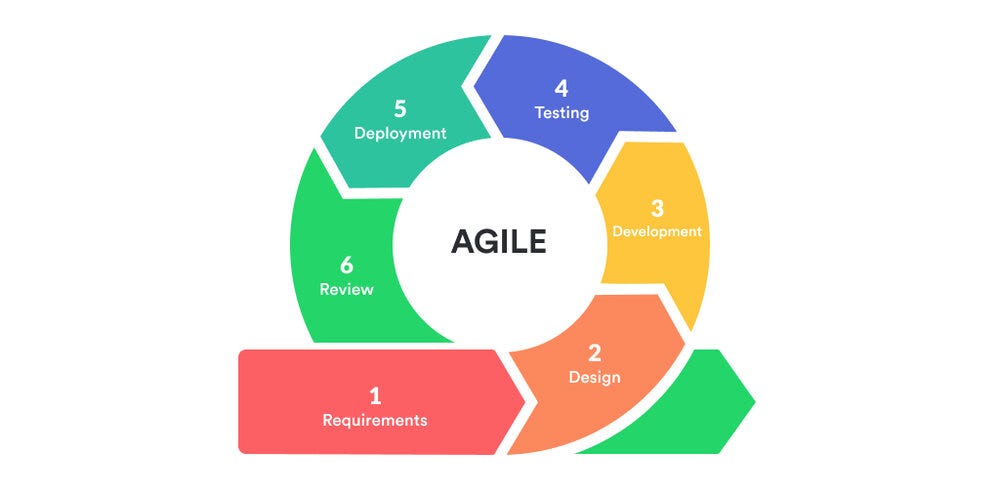
# REQUIREMENTS, ANALYSIS, AND DESIGN

## **3.1 Overview**

This chapter concentrates on defining the requirements, conducting analysis, and creating the system design for the FindNearMe mobile application. During the requirements gathering phase, interviews and observations were used to acquire information about users' functional and non-functional demands. The system analysis and design were shown using a variety of diagrams, including use cases, activity diagrams, data flow diagrams, and entities.

**3.2 Proposed Methodology**

The Agile methodology is particularly well-suited for FindNearMe due to its emphasis on flexibility, customer collaboration, and iterative progress. Agile allows for rapid adjustments based on user feedback, making it ideal for applications where requirements may evolve during development. This methodology consists of short development cycles known as sprints, typically lasting 1 to 4 weeks, during which specific features are developed, tested, and reviewed (Schwaber & Sutherland, 2017).



*Figure 3.1 Agile Methodology (Jason, J. 2022).*

Advantages:

1. **Adaptability**: Agile accommodates changes in requirements, which is essential for an application aimed at meeting diverse user needs in a dynamic marketplace.
2. **Frequent Feedback**: Regular interactions with stakeholders ensure that the product aligns with user expectations and market demands (Beck et al., 2001).
3. **Incremental Delivery**: Features can be released incrementally, allowing users to start benefiting from the application sooner rather than waiting for a full release.

While the Waterfall methodology is more traditional and linear—requiring completion of one phase before moving to the next—it may not be ideal for FindNearMe due to its rigidity. This methodology works best when project requirements are well-defined from the outset and unlikely to change (Royce, 1970). However, given the dynamic nature of local commerce applications and evolving user needs, this approach could lead to challenges if adjustments are needed after initial phases are completed.

**3.3 Methodology**

For the FindNearMe application, effective requirements gathering is critical to ensure that the final product aligns with user needs and stakeholder expectations. Based on research, two of the most effective methods for gathering requirements are Interviews and Observations.

**3.3.1 Interviews**

Interviews are a fundamental technique for gathering in-depth insights directly from stakeholders such as potential users (buyers and sellers), local business owners, and industry experts. This method allows one to ask specific questions about user needs, expectations, pain points, and desired features in a conversational manner, making it easier to gather qualitative data.

Interviews were conducted with consumers and vendors to understand their online shopping preferences. These interviews helped identify common needs and preferences, which informed the functional requirements of the system.

**3.3.2 Observations**

Observation is a valuable method for gathering requirements, particularly for applications like FindNearMe, where understanding user behavior and interactions with the system is crucial. This technique allows one to gain insights that may not be articulated during interviews or surveys.

Direct observations were carried out which revealed how consumers currently search for products in local shops, challenges faced by sellers in promoting their products and patterns in how users engage with local products vs. online marketplaces.

**3.4 Tools and Techniques**

The frontend was built with Expo, a react native framework, with Tailwind CSS. In terms of backend and database, Supabase was used, an open-source backend-as-a-service platform created as an alternative to Firebase. It offers developers a PostgreSQL database, user authentication, real-time features, and file storage.

Supabase streamlines application development by automatically creating APls and providing a user-friendly interface, making it accessible to both experienced developers and individuals with less technical expertise. So, this handles user authentication and real-time communications in addition to the backend and database.

EfficientNet, implemented in TensorFlow.js, is a state-of-the-art convolutional neural network developed by Google for image classification. This was utilized for the image search and matching functionality. For the mapping feature, Google Maps API was used.

**3.5 Ethical Consideration**

When developing the FindNearMe mobile application, several ethical considerations were taken into account to ensure user trust, safety, and compliance with legal standards. Here are the key ethical considerations relevant to this application:

1. Data Privacy and Protection: Given that FindNearMe will likely collect sensitive information such as location data and user profiles, it is essential to implement robust privacy measures.
2. User Control and Autonomy: Users should have control over their data and be able to manage permissions easily.
3. Accessibility: Given the disparities in internet access and digital literacy in Nigeria, the app should be designed to be as inclusive as possible, with considerations for users with limited internet access, older mobile devices, or lower digital literacy levels.
4. Ethical Use of AI: Since FindNearMe employs AI technologies for features like image recognition or personalized recommendations, ethical considerations surrounding AI must be addressed.
5. Fairness and Equity: Ensuring that the app does not unfairly promote certain sellers or products over others is important. Algorithms should be transparent, and sellers should have equal opportunities to showcase their products.

**3.6 Requirement Analysis**

**3.6.1 Software Requirements**

1. Front-end: Expo-React Native, Tailwind CSS
2. Back-end: Supabase
3. Database: Supabase
4. Geolocation: Google Maps
5. Operating System: Windows
6. Integrated Development Environment: VS code

**3.6.2 Hardware Requirements**

1. Mobile Phone: iOS or Android
2. A minimum of 4GB RAM
3. A stable internet connection

**3.7 Requirements Specifications**

**3.7.1 Functional Requirements Specifications**

Table 3.1: Functional Requirement Specifications

|  |  |
| --- | --- |
| **Req.**  **No.** | **Description** |
| FR1 | **User Registration and Authentication**: Users must be able to create an account using email or phone number. Users must be able to log in and log out securely. |
| FR2 | **Profile Management**: Users must be able to create and edit their profiles. Sellers must be able to manage their product listings. |
| FR3 | **Product Search and Discovery**: Users must be able to search for products using keywords or categories. The application must support image-based searches using AI-powered image recognition. Users must be able to view product details, including pricing and seller information. |
| FR4 | **Location-Based Services**: The app must provide users with location-based search results using GPS. Users must be able to view products on an interactive map. |
| FR5 | **In-App Messaging**: Users must be able to communicate with sellers through in-app messaging. Notifications must be sent for new messages or inquiries. |
| FR6 | **User Reviews and Ratings:** Users must be able to leave reviews and ratings for products and sellers. The system must display average ratings for products. |
| FR7 | **Admin Dashboard:** Admins must be able to manage user accounts and product listings. Admins must have access to analytics regarding user engagement and sales. |

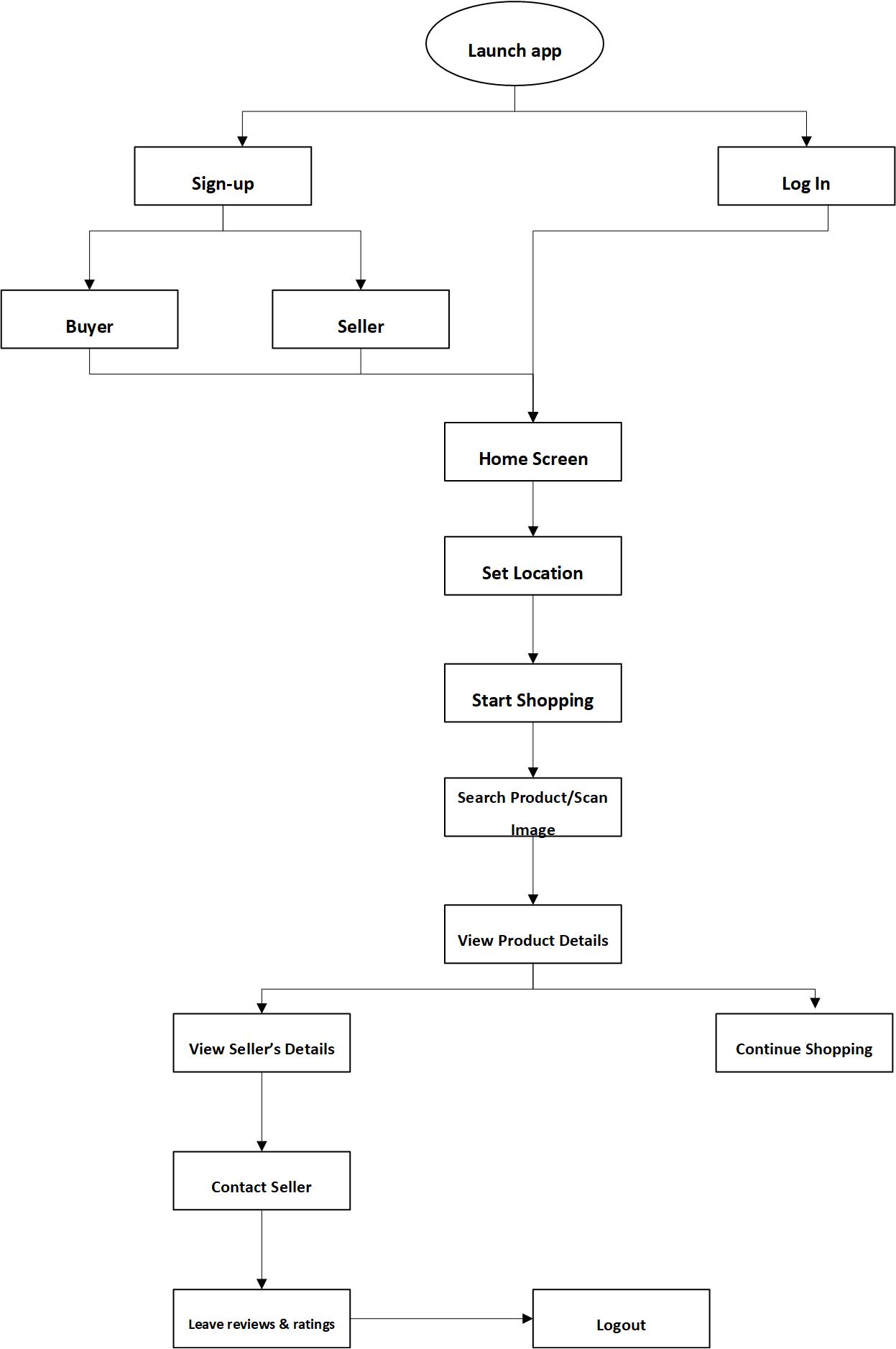
**3.7.2 Non-Functional Requirements Specifications**

Table 3.2: Non-Functional Requirement Specifications

|  |  |
| --- | --- |
| **Req.**  **No.** | **Description** |
| NFR1 | **Performance and Responsiveness**: The application should load fast under normal network conditions. Search results should be returned as soon as possible. |
| NFR2 | **Scalability**: The system should support up to 1000 concurrent users without degradation in performance. The architecture should allow for future scaling as user demand increases. |
| NFR3 | **Security**: User data should be encrypted both in transit and at rest. The application should comply with relevant data protection regulations. |
| NFR4 | **Usability**: The app should provide a seamless onboarding experience for new users. |
| NFR5 | **Compatibility**: The application should support the latest versions of iOS and Android operating systems. It should function correctly across various devices |
| NFR6 | **Maintainability**: Code should be modular and well-documented to facilitate future updates. The system should allow for easy integration of new features without major overhauls. |
| NFR7 | **Reliability**: The system should be reliable with little to no downtime during business hours. Backup systems should ensure data recovery within 24 hours in case of failure. |

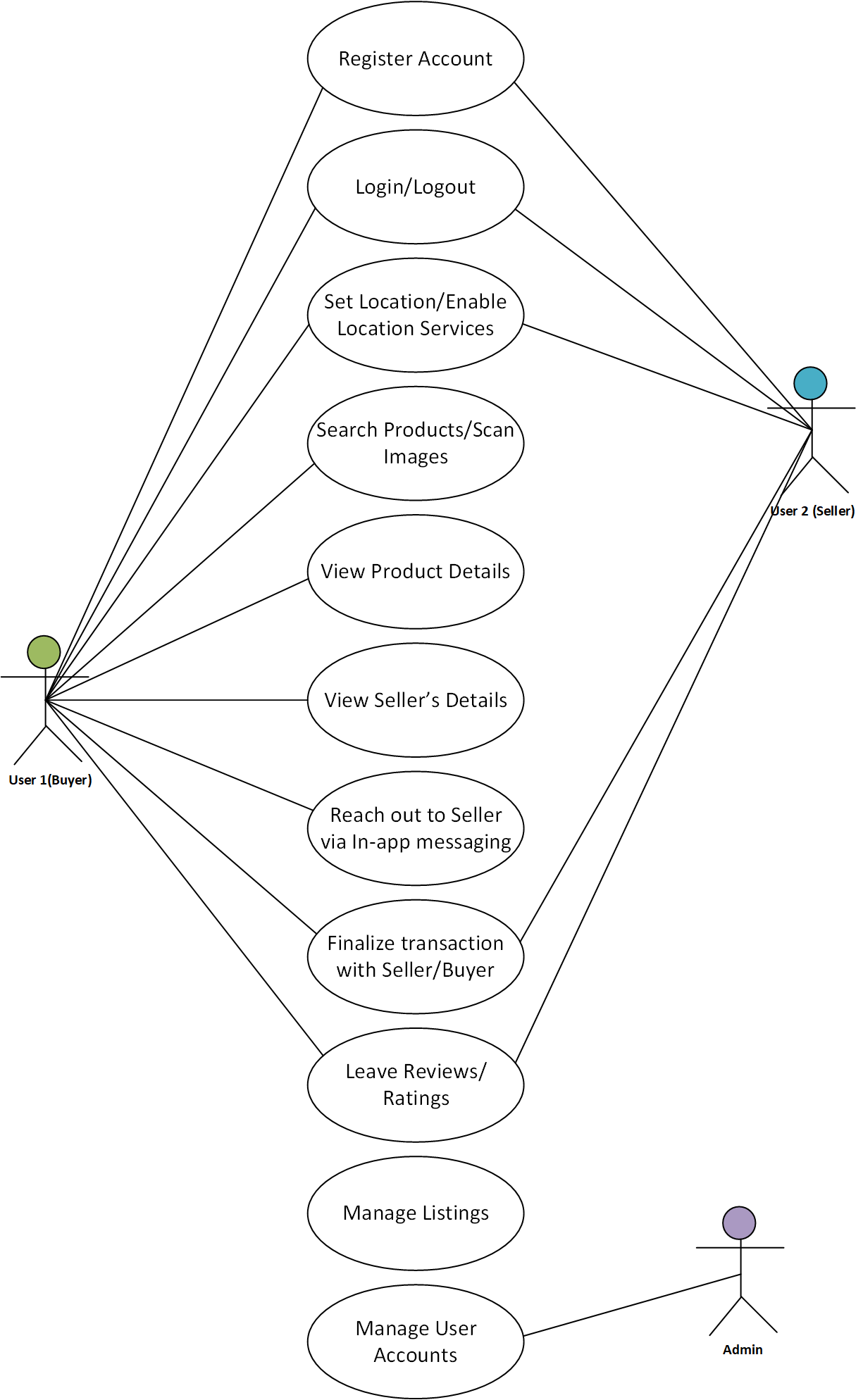
**3.8 System Design**

**3.8.1 Application Architecture**

****

*Figure 3.2 Application Architecture*

**3.8.2 Use Case Diagram**

****

*Figure 3.3 Use Case Diagram*

**3.8.3 Use Case Description**

Table 3.3: Use case Description for Sign-Up/Log-In

|  |  |  |
| --- | --- | --- |
| Use Case: | Sign-Up New User /Log-In | |
| Description | This use case describes the process by which users can create an account (sign-up) or log into the FindNearMe application. The system allows users to register with their email or phone numbers and subsequently log in to access personalized features. | |
| Actors: | User | |
| Preconditions: | 1. The user has access to the internet. 2. The user has a valid email address (for sign-up). 3. The user must not have previously registered with the same email (for sign-up). | |
| Postconditions: | 1. If the sign-up is successful, the user's details are stored in the database, and they are redirected to their profile page. 2. If the login is successful, the user is redirected to their dashboard. | |
| Main flow | User:   1. The use case begins when the user clicks the “Sign Up/Log-In” tab on the navigation bar. 2. The system displays the sign-up page, prompting the user to enter their username/email address, first name, last name, and password or log-in details. 3. The user fills in the required fields and clicks the “Sign Up/Log-In” button. | System:   1. The system verifies the given credentials. 2. If the verification is successful, the user is logged into the application. Otherwise, display an error message. 3. The system creates new user account and stores details in database. 4. Use case ends. |
| Exception  Condition: | If the login credentials are incorrect, an error message will appear. The user then has the option to retry entering correct credentials or cancel which will end the current process. | |

Table 3.4: Use case Description for Search Products

|  |  |  |
| --- | --- | --- |
| Use Case: | Search Products | |
| Description | This use case describes how users can search for products or services available in their vicinity using keywords, categories, or image upload. | |
| Actors: | User: Individuals looking to find products or services nearby. | |
| Preconditions: | 1. The user must be logged into their account. 2. The user’s location must be enabled on their device. | |
| Postconditions: | The system displays a list of products matching the search criteria. | |
| Main flow | User:   1. The use case begins when the user clicks the “Search” icon on the home screen. 2. The user inputs the search term/upload/scan image and submits the query. | System:   1. The system prompts the user to enter a keyword or select a category or upload/scan image. 2. The system retrieves product listings based on the user's location and search criteria. 3. The system displays the list of matching products, including images, prices, distance from user and seller information. |
| Exception  Condition: | **“No Results Found”** is triggered if no products match the search criteria. The system displays a message indicating no results were found and suggests alternative searches. | |

Table 3.5: Use case Description for View Product Details

|  |  |  |
| --- | --- | --- |
| Use Case: | View Product Details | |
| Description | This use case allows users to view detailed information about a specific product, including images, descriptions, pricing, and seller information. | |
| Actors: | User: Individuals interested in learning more about a specific product. | |
| Preconditions: | The user must have performed a product search or accessed a product listing. | |
| Postconditions: | The user views detailed information about the selected product. | |
| Main flow | User:   1. The use case begins when the user selects a product from the search results. 2. A user can choose to add the product to their cart or contact the seller for more information. | System:   1. The system retrieves detailed information about the selected product from the database. 2. The system displays the product details page, including: Product images, Description, Price, Seller information, Availability status. |
| Exception  Condition: | **“Product Unavailable”** is triggered if the selected product is no longer available. The system displays a message indicating that the product is unavailable and suggests similar products. | |

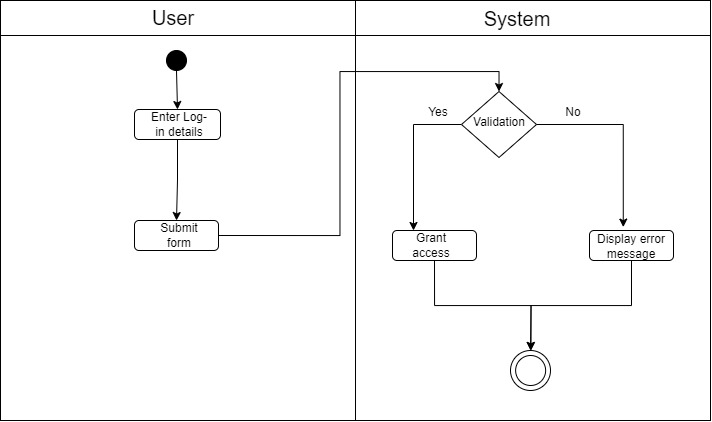
Table 3.6: Use case Description for Contact Seller via In-App Messaging

|  |  |  |
| --- | --- | --- |
| Use Case: | Contact Seller via In-App Messaging | |
| Description | This use case describes the process by which users can communicate with sellers through an in-app messaging feature. This functionality allows buyers to ask questions, negotiate prices, and confirm details about products before making a purchase. | |
| Actors: | 1. User: Individuals interested in a product who wish to contact the seller. 2. Seller: Individuals or businesses listing products for sale. | |
| Preconditions: | 1. The user must be logged into their account. 2. The user must have viewed a product listing that they are interested in. | |
| Postconditions: | 1. The message is sent to the seller, and the user receives a confirmation of the sent message. 2. The seller can respond to the user’s inquiry through the same in-app messaging system. | |
| Main flow | User:   1. The use case begins when the user views a product detail page. 2. The user clicks on the "Contact Seller" button. 3. The user composes their message (e.g., asking about product availability or negotiating price) and clicks the “Send” button. 4. The seller receives a notification of the new message in their seller dashboard. | System:   1. The system displays a "Contact Seller" button on the product detail page. 2. The system opens an in-app messaging interface where the user can type their message. 3. The system validates that the message is not empty and sends it to the seller’s inbox. 4. The system displays a confirmation message indicating that the message has been sent successfully. |
| Exception  Condition: | **“Message Sending Error”** is triggered if there is an issue sending the message (e.g., network connectivity problems). The system displays an error message prompting the user to try again later. | |

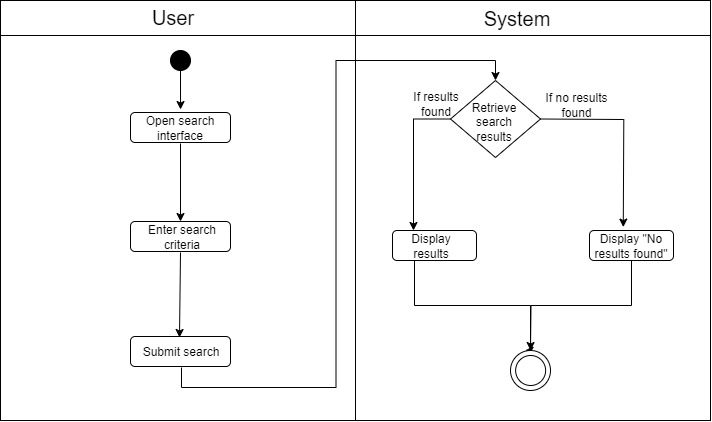
Table 3.7: Use case Description for Leave Reviews/Ratings

|  |  |  |
| --- | --- | --- |
| Use Case: | Leave Reviews/Ratings | |
| Description | This use case allows users to provide feedback on products they have purchased by leaving reviews and ratings for sellers and items. | |
| Actors: | User: Individuals who have completed a purchase and wish to leave feedback. | |
| Preconditions: | The user must have completed a purchase of a product from a seller. | |
| Postconditions: | The review and rating are stored in the database and associated with both the product and seller profiles. | |
| Main flow | User:   1. The use case begins when the user navigates to their liked list. 2. The user selects a product they wish to review. 3. The user submits their review. | System:   1. The system displays a list of purchased products eligible for review. 2. The system prompts the user to enter a rating (e.g., 1 to 5 stars) and write comments about their experience. 3. The system stores the review in the database and updates average ratings for both the product and seller profiles. |
| Exception  Condition: | **“Review Submission Error”** is triggered if there is an issue with submitting the review (e.g., network error). The system displays an error message prompting users to try again later. | |

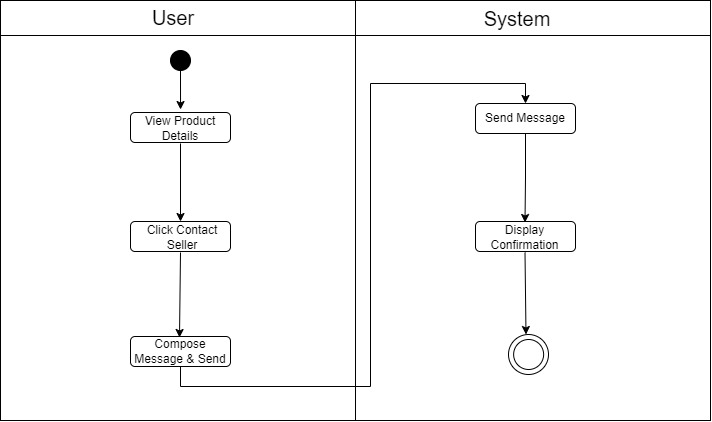
**3.8.4 Activity Diagrams**

****

*Figure 3.4 Activity Diagram for User Log-In & Authentication*

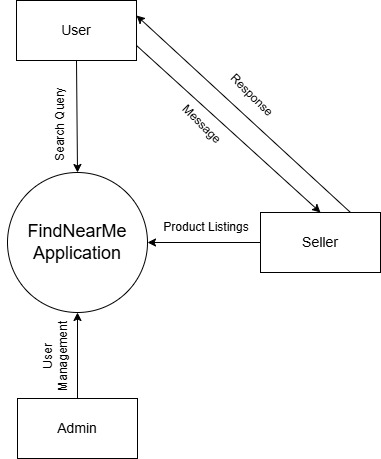
****

*Figure 3.5 Activity Diagram for Product Search*

****

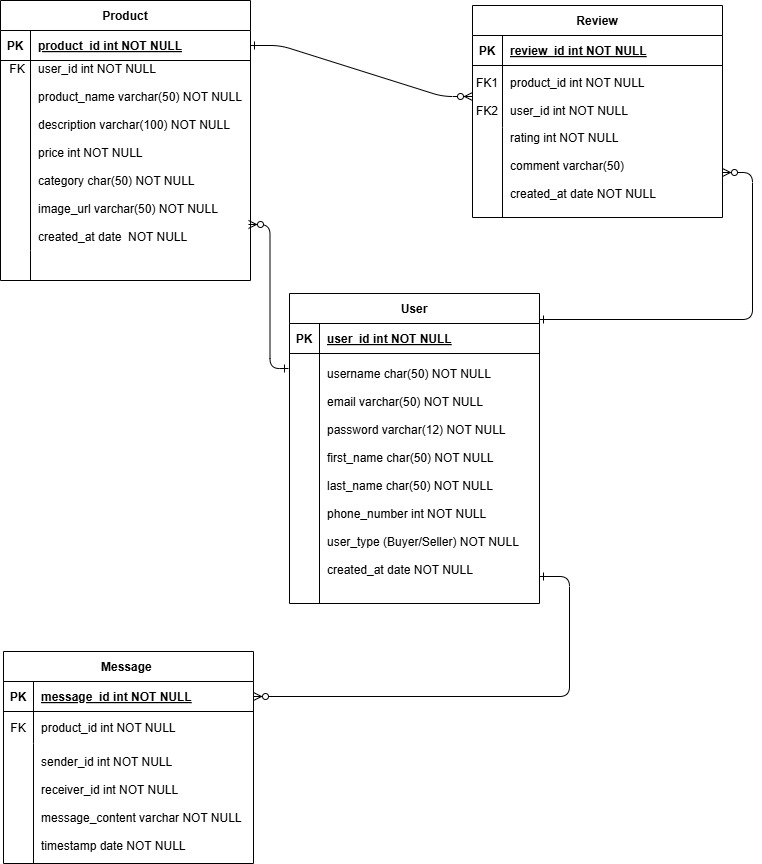
*Figure 3.6 Activity Diagram for Contact Seller*

**3.8.5 Dataflow Diagram**

****

*Figure 3.7 Data Flow Diagram (Level 0)*

**3.8.6 Entity-Relationship Diagram (ERD)**

****

*Figure 3.8 Entity Relationship Diagram*

**3.8.7 User Interface Design**

**3.9 Summary**

This chapter outlines the project's development process, strategy, tools, and procedures. It discusses the ethical considerations and provides both functional and non-functional requirements. This is based on the requirements analysis and design used throughout project development. The system design for the application integrates various components to create a cohesive platform that facilitates local commerce. By employing structured architecture and clear data flow processes alongside well-defined entities and relationships, the design aims to deliver an efficient and user-friendly experience. This comprehensive approach ensures that all functional requirements are met while allowing for scalability and future enhancements.

**References**

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Jeffries, R. (2001). Manifesto for Agile Software Development.

HubSpot Blog. (2018). A Beginner's Guide to Data Flow Diagrams.

Royce, W. W. (1970). Managing the Development of Large Software Systems. In *Proceedings of IEEE WESCON* (pp. 1-9).

Schwaber, K., & Sutherland, J. (2017). The Scrum Guide. Retrieved from <https://www.scrumguides.org/scrum-guide.html>