**Chapter Four**

**Implementation and Testing**

**4.1 Overview**

This chapter presents the comprehensive process involved in the implementation and testing of the **FindNearMe** mobile application. The chapter describes the primary features of the application, the challenges faced during the implementation phase, and the solutions used to address these issues. Furthermore, it explains the testing procedures used to evaluate the application and assure its functionality, dependability, and usability. Lastly, the chapter provides a usage guide for end-users and concludes with a summary of all essential points.

**4.2 Main Features**

The **FindNearMe** application is designed to enhance the user experience by connecting local buyers and sellers through a variety of features. Below are the primary features that define the functionality of the app:

1. **User Registration and Login**: Users can create accounts using their email or mobile numbers. Secure login functionality ensures user data protection.
2. **Profile Management**: Users can create and edit their profiles, including personal information and preferences. Sellers can also manage their product listings directly from their profiles.
3. **Product Search Functionality**: Users can search for products using image recognition, categories, or keywords. The app provides filters to refine search results based on location, price range, and product type.
4. **Location-Based Services**: Utilizes GPS to display nearby products based on the user's current location. Users can view products on an interactive map, making it easier to find local offerings.
5. **In-App Messaging**: Facilitates direct communication between buyers and sellers through a secure messaging system. Users can ask questions about products or negotiate prices without leaving the app.
6. **Product Listings Management**: Sellers can create, edit, and delete their product listings with ease. Each listing includes detailed descriptions, images, pricing, and seller information. Users can save preferred products or sellers to a favorites list for easy access later.
7. **User Reviews and Ratings**: Users can leave reviews and ratings for products and sellers based on their experiences. The system aggregates ratings to provide average scores for products, enhancing trust among users.

**4.3 Implementation Problems**

During the development and implementation phase, several challenges arose such as:

1. **Cross-Platform Compatibility**: Ensuring consistent performance across different devices and operating systems (iOS and Android) required extensive testing and adjustments to UI elements.
2. **User Data Security**: Protecting user data against unauthorized access necessitated implementing robust security measures, including data encryption and secure authentication protocols.
3. **Image Import Issues**: During development, image imports were not functioning correctly, with the debugger showing that image paths were not being logged as expected. The issue originated from incorrect path alias configuration in tsconfig.json.
4. **Model Compatibility and Implementation**: Integrating the MobileNet model from TensorFlow.js posed several compatibility issues within the Expo environment, including: performance problems leading to application errors, significant runtime delays affecting user experience and slow product creation due to real-time image analysis.

**4.4 Overcoming Implementation Problems**

To address these challenges, the following solutions were employed:

* **Responsive Design Techniques**: The development team utilized responsive design principles to ensure that the application displayed correctly on various screen sizes and orientations.
* **Enhanced Security Measures**: The application incorporated industry-standard security practices, such as OAuth2 for authentication and secure data transmission.
* **Image Import Resolution:** To resolve the image import issues, instead of using relative paths like: **import** image **from** './assets/images/placeholder.jpg'; the correct implementation using the configured alias was: **import** placeholder **from** "@/assets/images/placeholder.jpg".
* **Cron Job Implementation**: The adopted solution for model compatibility involved implementing an asynchronous processing system using cron jobs, supported by a well-structured database schema. The cron job approach eliminates wait time during product creation, provides scalable processing based on application load, supports batch processing and operates independently from the main application flow.

**4.5 Testing**

A crucial step in guaranteeing the FindNearMe app's security, dependability, and performance was testing. The system was validated using both functional and non-functional testing approaches.

**4.5.1 Tests Plans**

Table 4.1: Test Plans

|  |  |
| --- | --- |
| **Test Plan ID** | TP-FNM-001 |
| **Test Plan Name** | FindNearMe Test Plan |
| **Version** | 1.0 |
| **Date** | 11/16/2024 |
| **Author** | Hadiza Aliyu |
| **Objective** | The primary objectives of this test plan are to:   * Verify that all features of the FindNearMe application function as intended. * Identify and resolve defects before the application is released to users. * Ensure that the application meets performance, security, and usability standards. |
| **Scope** | The scope of testing includes:   * Functional Testing * Usability Testing * Performance Testing * Security Testing * Compatibility Testing (across devices and operating systems) |
| **Test Strategy** | The testing strategy will encompass various types of testing to ensure comprehensive coverage:   * Functional Testing: Validate that all features work according to requirements. * Usability Testing: Assess user experience and interface design. * Performance Testing: Evaluate the application's responsiveness under load. * Security Testing: Identify vulnerabilities and ensure data protection. * Compatibility Testing: Test the application on different devices, screen sizes, and operating systems**.** |

**4.5.2 Test Cases**

Table 4.2: Test Case for User Registration

|  |  |
| --- | --- |
| **Test Case ID** | TC-FNM-001 |
| **Test Type** | Functional |
| **Description** | Verify that a new user can register successfully. |
| **Preconditions** | User is on the registration page. |
| **Steps** | 1. Enter valid username, email, and password. 2. Click "Sign Up." |
| **Expected Result** | User receives a confirmation message and is redirected to the profile page. |
| **Status** | Pass |

Table 4.3: Test Case for Product Search

|  |  |
| --- | --- |
| **Test Case ID** | TC-FNM-002 |
| **Test Type** | Functional |
| **Description** | Verify that users can search for products using keywords and image recognition. |
| **Preconditions** | User is logged in. |
| **Steps** | 1. Enter a keyword in the search bar/ Upload an Image. 2. Click "Search." |
| **Expected Result** | Relevant products are displayed based on the search query. |
| **Status** | Pass |

Table 4.4: Test Case for In-App Messaging

|  |  |
| --- | --- |
| **Test Case ID** | TC-FNM-003 |
| **Test Type** | Functional |
| **Description** | Verify that users can send messages to sellers. |
| **Preconditions** | User is viewing a product listing. |
| **Steps** | 1. Click "Contact Seller." 2. Compose a message and click "Send." |
| **Expected Result** | Message is sent successfully, and a confirmation is displayed. |
| **Status** | Pass |

Table 4.5: Test Case for Navigation

|  |  |
| --- | --- |
| **Test Case ID** | TC-FNM-004 |
| **Test Type** | Usability |
| **Description** | Verify that users can navigate through the app intuitively. |
| **Preconditions** | User is logged in. |
| **Steps** | 1. Access different sections (Home, Profile, Messages). |
| **Expected Result** | Users can navigate without confusion. |
| **Status** | Pass |

Table 4.6: Test Case for Load Handling

|  |  |
| --- | --- |
| **Test Case ID** | TC-FNM-005 |
| **Test Type** | Performance |
| **Description** | Verify that the application handles a high number of concurrent users. |
| **Preconditions** | Load testing environment set up. |
| **Steps** | 1. Simulate multiple users accessing the app simultaneously. |
| **Expected Result** | Application remains responsive without significant lag. |
| **Status** | Pass |

Table 4.7: Test Case for Data Protection

|  |  |
| --- | --- |
| **Test Case ID** | TC-FNM-006 |
| **Test Type** | Security |
| **Description** | Verify that user passwords are encrypted in the database. |
| **Preconditions** | User registration completed. |
| **Steps** | 1. Check database entries for user passwords. |
| **Expected Result** | Passwords should not be stored in plain text. |
| **Status** | Pass |

Table 4.8: Test Case for Cross-Device Compatibility

|  |  |
| --- | --- |
| **Test Case ID** | TC-FNM-007 |
| **Test Type** | Compatibility |
| **Description** | Verify that the app functions correctly on different devices (iOS and Android). |
| **Preconditions** | Devices with various OS versions available. |
| **Steps** | 1. Install and run the app on different devices (e.g., iPhone, Samsung Galaxy). |
| **Expected Result** | The app displays correctly and functions without issues on all devices. |
| **Status** | Pass |

**4.5.3 Error Reports and Corrections**

This report outlines the identified errors during the testing phase of the FindNearMe application, along with their descriptions, severity levels, and proposed corrections. The goal is to provide a clear understanding of the issues and the steps needed to resolve them.

Table 4.9: Error Reports and Corrections

|  |  |  |  |
| --- | --- | --- | --- |
| Error ID | Description | Severity | Proposed Correction |
| ERR-001 | Product search returns no results for valid keywords. | High | Investigate search algorithm; ensure it queries the database correctly. |
| ERR-002 | In-app messaging experiences delays in message delivery. | Medium | Optimize WebSocket implementation for real-time messaging; check server load handling. |
| ERR-003 | User profile updates do not reflect immediately in the UI. | Low | Implement state management to refresh user data after updates. |
| ERR-004 | App crashes when accessing product details from search results on certain devices. | Critical | Investigate device-specific compatibility issues; perform debugging on affected devices. |

**4.6 User Guide**

The following is a brief guide for users on how to navigate and use the FindNearMe application:

1. Installation: Get the app from the App Store (iOS) or Google Play Store (Android).
2. Creating an Account: Open the app and tap **"Sign Up.",** fill in your details and tap **"Create Account,** verify your email via the link sent to you. Tap **"Login,"** enter your credentials, and access your account.
3. Navigating the App:
   1. **Home Screen**: Displays featured products and a search bar.
   2. **Profile**: Access your account settings and order history via the profile icon.
   3. **Image Upload**: Displays Image Upload section
   4. **Location**: Displays Map where you can set your location.
4. Searching for Products:
5. Tap the **Search Bar**.
6. Enter keywords or categories/ Upload an image of what you are looking.
7. Tap **"Search"** to view results.
8. Use filters to refine your search.
9. Viewing Product Details: Tap a product from search results to view images, descriptions, pricing, and seller info.
10. Contacting Sellers:
11. On the product page, tap **"Contact Seller."**
12. Compose your message and tap **"Send."**
13. Managing Your Profile: Access your profile to edit personal information, change your password, and view order history.
14. Leaving Reviews and Ratings:
15. Go to your order history.
16. Select an order, rate it (1-5 stars), and leave a comment.
17. Tap **"Submit Review."**

**4.7 Summary**

This chapter outlines the implementation and testing processes for the FindNearMe mobile application. It details key features such as user registration, product search, in-app messaging, and geotagging services. The implementation involved setting up the development environment, developing core functionalities, and integrating third-party services. Challenges included ensuring cross-platform compatibility and optimizing real-time messaging. Solutions involved responsive design techniques and using WebSocket protocols for efficient communication. Testing procedures included functional, usability, performance, security, and compatibility testing to ensure quality standards were met. A user guide was also provided to help users navigate the app effectively. In conclusion, this chapter emphasizes the significance of thorough implementation and testing in delivering a reliable, user-friendly application that meets expectations across platforms.

**References**

Educative.io. (2024). The complete guide to System Design in 2024.