Testing Report

Section 1: Introduction

1.1 Test Project Name: EZMedi

1.2. Summary of the Rest of the Test Plan:

The EZMedi Testing Report provides a comprehensive evaluation of the EZMedi mobile application, an iOS-based platform designed for medication management.

The report covers some key features like, medicine identification via barcode scanning, personal library management, and personalized medical reminders. The testing strategy was focused on the application's functionality, performance, security, and usability, utilizing tools such as XCode Simulators and physical iPhones.

Testing showed that EZMedi has an easy-to-use style and efficiently handles the most designed functionalities. Due to the application being still in its early stages of development, however, some issues with load and security testing remained unresolved.

Based on the findings, recommendations include performance optimization, the implementation of asynchronous loading and multithreading for improved efficiency.

The report's overall findings suggest that although EZMedi demonstrates promising capabilities in terms of medication management and user accessibility, there is room for improvement, especially in terms of performance enhancement and feature expansion to fully realize the platform's potential to support patients with their medication management needs.

Section 2: Feature Description

EZMedi is an iOS mobile application that allows patients to access information regarding their medication and generate reminders.

The system is primarily intended to connect users with a list of their own medications and reminders. It is not meant to be integrated with other health providers at this stage or utilized for healthcare providers' own records.

User Authentication: Login Page with email and password verification
Medicine Identification: Barcode/QR scanning using the camera and it will automatically add to the personal profile.
Medication Management: Edit/View a list of medications including information such as medication name, dosage and frequency.

Medicine Input and Search: Input the name and the company of the medicine manually to add it to the personal profile.
Medicine Reminders: Opting in for Medication Reminder and Entry of Details, setting the time for the medication and it will push notification on time.
Personal Medical Profile: Automatically add searched medicine to the user profile. Able to check and edit the reminder in the medical profile.

Section 3: Assumptions

3.1 Test Case Exclusions: List any test cases or scenarios that are excluded from testing.

- Testing on Non-IOS Platforms: this is due to the fact that EZMedi is being developed on IOS.
- Testing with non-standard hardware such as medical devices.
- Testing with very rare medications or very recently developed ones
- Testing in a language other than English
- Testing in Integration with third party applications that were not mentioned in the scope
- Testing with multi-faceted and very complex medical conditions

3.2 Test Tools, Formats, and Organizational Schemes: List the testing tools, formats, and organizational schemes used

- Testing tools: XCode Simulators and physical phones
- Formats: iOS application
- Organizational Scheme
    - Different test case for different pages (Login, Search, Profile)
    - Different users
    - Different platforms and devices

Section 4: Test Approach

4.1 Special Testing Considerations: Highlight any special considerations for testing.

1. The system must adhere to the healthcare regulations outlined in the SRS.
2. Since the system is dependent upon external APIs such as RXNorm, these integrations must also be proven to be reliable and integrated seamlessly with EZmedi
3. The system is connected with google's firebase database and we have to consider following google's rules and carefully about the test and visit rules of firebase.

4. Security is paramount in a system with information as sensitive as those of EZmedi; there must be an emphasis on testing for vulnerabilities related to unauthorized access and potential data breaches.

4.2 Test Strategy: Explain which testing techniques are used to test the different parts of the systems. Provide a rationale for the selection. Also include information on which tools, automation, and scripts are used to test each part of the system.

Unit Testing: Unit testing is important to detect any potential issues early on in the project. It is performed on the smallest testable part of an application (a unit). We have done this manually as were building each individual view/component of the system.

Integration Testing: Important to assess if all components are seamlessly integrated. Done manually through XCode simulators.

System Testing: Important to verify the complete software behaves as expected across different iOS devices. Done through XCode simulators.

Security Testing: Important to ensure the protection of User data. Done manually through user input (name, password, profile visit)

Usability Testing: To ensure that app is user-friendly and accessible to people from different backgrounds. Done through user feedback and manual testing.

We are also utilizing control systems such as GitHub to properly document any findings and manage EZMedi's different scripts.

4.3 Test Categories: Categorize the test cases, e.g., functional, performance, security, etc.

- Function testing: This ensures that all features/functions of the software operate in compliance with EZMedi's SRS.
- Performance testing: This will assess the performance of the system in terms of speed and responsiveness and its ability to handle a certain load.
- Security testing: This ensures that the system is safe from unauthorized access and any breaches or vulnerabilities.
- Usability testing: This ensures that the system is user-friendly and allows the users to seamlessly navigate through its different features.

Section 5: Test Cases

5.1 Test Group and Subgroup Definition: This is where you will define how the test cases will be structured and organized. Define test groups and subgroups for organizing test cases. Specify the objective for each group

- Function Testing Group:

  Objective: To verify that all features/functions of the software operate in compliance with EZMedi's SRS.

  Subgroups:

  Medication Identification: testing medication identification through both the barcode scanning and search by name functionalities.

  Medication Reminders: testing the reminder functionality's capability in terms of both accuracy and reliability.

- Performance testing:

  Objective: To assess the performance of the system in terms of speed and responsiveness and its ability to handle a certain load.

  Subgroups:

  Load Testing: test the system's performance under high load conditions.

  Stress Testing: test the system's robustness by testing under extreme conditions beyond normal operation, testing the case that multiple users are using the system and using the system for a long time.

- Security testing:

  Objective: To ensure that the system is safe from unauthorized access and any breaches or vulnerabilities.

  Subgroups:

  Internal Data Security: test for data integrity and unauthorized access.

  External API Security: test the security of external RxNorm API integrations with the system.

- Usability testing:

  Objective: To ensure that the system is user-friendly and allows the users to seamlessly navigate through its different features.

  Subgroups:

  SwiftUI testing: test the overall layout of the application and its navigability.

  Accessibility: ensure that the application is usable to users with a myriad of capabilities.

- 5.2 Test Cases: List all test cases. Or provide a link to the test cases. Ensure that the link you provide is accessible to the instructor.

  https://docs.google.com/document/d/1rIMR9Zm9MzeHsgDn4Lj2NoXa5MetC70tFTa_1A16iVk/edit?usp=sharing

5.3 Traceability Matrix: Create a traceability matrix to map requirements to test cases

The Requirement ID is identified per the SRS

Requirement ID	Requirement Description	Test Case IDs	Status
1.1	Medication management for chronic diseases	6,7	passed
1.3.1	Integration with external APIs	1, 2, 3, 4, 8, 9, 10, 12,14, 18, 20, 21, 23, 24	1,4,7 - passed 5, 6- not tested
1.3.2 A	User Authentication	16, 19	passed
1.3.2 B	Medicine Identification via Barcode Scanning	1, 2, 8, 9, 12, 17, 18, 22	1, 8, 12, 17, 18, 22- passed 2, 9 - not tested
1.3.2 C	Medicine Input and Search	3, 4, 5, 10, 12, 14, 17, 19, 20, 21, 23, 24	3, 4, 10, 14, 17, 19, 20, 21, 23, 24 - pass 5, 12 - not tested
1.3.2 E	Medicine Reminders	6, 7, 11, 13, 15, 25	6, 11, 13, 15- passed 7, 25 - not tested
1.3.2 F	Personal Medical Profile Management	17, 19	17, 19 - passed
1.3.4 B	Hardware Limitations and Performance Requirements	25	25 - not tested

---

Section 6: Test Environment

6.1 Multiple Test Environments: List the different test environments used if applicable.

- Simulators on XCode
    - iPhone 15 Pro Max
    - iPhone 12
- Physical phones
    - iPhone 12 Pro Max
    - iPhone 12

6.2 Schematic Diagram: Provide a schematic diagram of the test environment setup if applicable.

Not yet applicable.

6.3 Test Architecture Overview: Explain the overall test architecture if applicable.

Not yet applicable.

6.4 Equipment Table: List the equipment and resources used in the testing environment if applicable.

Equipment/Resource	**Description**
Simulators on XCode: iPhone 12	Software-based iPhone simulators #1 Supported by Mac
iPhone 15 Pro Max	Software-based iPhone simulators #2 Supported by Mac
iPhone 12 Pro Max	Physical device for testing #1 Loaded with iOS 17
iPhone 12	Physical device for testing #2 Loaded with iOS 14
iPhone 13 Pro Max	Physical device for testing #3 Loaded with iOS 16.6
iPhone 12 Pro Max	Physical device for testing #4 Loaded with iOS 16.6.1
iPad Air 4	Physical device for testing #5 Loaded with iOS 17.1
iPad Pro 3	Physical device for testing #6 Loaded with iOS 14.6

Section 7: Testing Results

Provide a summary of testing results, including passed, failed, and unresolved issues

Passed:

- Functionalities of barcode scanning and search by name, successfully extracting the barcode number and could search the medication through keywords.
- Functionalities of reminders: The system will push notifications to the user if they allow the application to push the notification to the phone on time.
- The overall layout of the application contains all the functionalities and each button correctly navigates to the target page.
- The application is developed on the iOS platform, featuring user-friendly accessibility options that enhance overall usability.

Unresolved Issues:

- Testing about the hardware limitations.
- Since the application is not fully built, it temporarily could not perform load testing and security testing.

Section 8: Recommendations on Software Quality

Offer recommendations on improving the quality of the software based on testing results

1. When opening the app, sometimes the system response is relatively slow.
    a. Performance Optimization: Analyze and optimize the application code to ensure high execution efficiency and reduce resource usage.
    b. Asynchronous Loading: Design time-consuming operations as asynchronous tasks to allow users to perform other actions in the background, enhancing user experience.
2. Sometimes response is slow when scanning and recognizing barcodes.
    a. Multithreading:

        Move scanning and image processing tasks to background threads to prevent the main thread from blocking.

Enhance system efficiency during barcode scanning through concurrent processing.

   b. Scanning barcode pictures with higher contrast. Combine multiple barcode recognition algorithms to improve the scanning performance.

3. The success rate of barcode recognition during scanning is not high.
   a. Algorithm Fusion:

      Combine multiple barcode recognition algorithms to improve the success rate.

      Consider using comprehensive recognition libraries that support various barcode standards.

4. The reminders should add weekly or monthly reminders for a wider range of usage.
5. Better UI design on the barcode scanning, which we hope after scanning and add the medicine to the library the system could turn back to the search page.
   a. Change the logic of the return page view.