Ryan Sharma, Zakeriya Muhumed

Term Project: Instant Messaging App

Test Plan Document

Table of Contents

1 Introduction

This test plan document outlines the strategy, scope, approach, and resources required for the testing activities of the Discom messaging application. It aims to ensure that all functionalities of the messaging application are verified against the specified requirements, and that the application operates reliably, efficiently, and securely.

1.1 Purpose

This test plan document will provide a comprehensive guide for the testing activities of the simple messaging application. It serves as a roadmap for the testing process, outlining the strategy, scope, objectives, resources, and schedule for testing. This document ensures that all team members and stakeholders have a clear understanding of the testing goals, processes, and responsibilities.

1.2 Target Audience

The target audience for this test plan document are any stakeholders involved in the development, testing, and deployment of the simple messaging application. Primarily, it is intended for the QA (Quality Assurance) team responsible for executing the test cases and ensuring that the application meets the specified requirements and quality standards. Project managers and product owners are another key audience, as the document provides them with a clear overview of the testing strategy, schedule, and resource allocation, facilitating better project planning and risk management. Additionally, business analysts and client representatives can refer to this document to ensure that the testing activities align with business requirements and user expectations. Finally, technical support and maintainers will find this document useful for understanding potential issues and ensuring that any defects identified during testing are properly addressed and documented.

1.3 Terms and Definitions

Cypress Testing Framework: Cypress is an open-source testing framework for web applications, enabling automated tests that simulate user interactions within the browser.

Encryption: Encryption protects sensitive data by converting it into an unreadable format ensuring only authorized parties can access it.

Fault Tolerance: Fault tolerance measures a system's ability to continue operating despite faults or failures, often expressed as the maximum acceptable failure rate.

Feature Testing: A type of testing that focuses on verifying that specific functionalities (features) of the application work as intended.

Jest Testing Framework: Jest is a JavaScript testing framework known for simplicity and speed, supporting unit tests, integration tests, and snapshot tests.

Release Testing: Release testing is conducted to validate a particular version or release of the software before it is deployed to production or made available to users. Aims to ensure that the release is stable, reliable, and free from critical defects.

System Testing: A comprehensive testing phase where the entire application is tested as a whole to verify that it meets the specified requirements.

Unit Testing: A level of software testing where individual units or components of the software are tested in isolation from the rest of the application.

XSS Vulnerability: XSS is a web security flaw in which attackers inject malicious scripts into web pages viewed by other users, enabling theft of sensitive data or session hijacking.

Discom: The subject chat/real-time messaging application.

2 Test Plan Description

This section provides an overview of the scope of testing, the testing schedule, and release criteria. It outlines the testing strategy and methodologies to be employed throughout the testing process.

2.1 Scope of Testing

The scope of the test plan encompasses various levels and functionalities. At the unit testing level, we will examine individual components of the application, such as functions and methods, to ensure they function correctly in isolation. Feature testing will focus on verifying the functionality of key features like user authentication, messaging, contact management, and notifications, ensuring they meet specified requirements and operate seamlessly within the application. System testing will involve testing the application as a whole, including its interactions with external systems, to validate its performance, reliability, and security in real-world scenarios. Finally, release testing will involve conducting a final round of tests to ensure the application is ready for deployment, checking for any last-minute issues or regressions before release.

However, it's essential to note that certain aspects of the project will not be within the scope of this test plan. For example, while we will ensure that the application functions correctly on various devices and browsers, extensive compatibility testing may not be feasible within the constraints of this plan. Additionally, while we will strive to cover a wide range of test scenarios, exhaustive testing of every possible edge case may not be practical within the allocated time and resources. Therefore, some corner cases and rare scenarios (i.e. heavy user traffic)may not be explicitly tested as part of this plan.

2.2 Testing Schedule

Planning and preparation (3 days):

- Review requirements and finalize test plan.
- Set up testing environments and tools.
- Assign roles and responsibilities.

Unit testing (2 - 4 days):

- Develop unit test cases for individual components.
- Execute unit tests and debug as necessary.
- Document test results and track defects.

Feature testing (2 - 4 days):

- Test key features of the application, including user authentication, messaging, contact management, and notifications.
- Conduct functional, usability, and performance testing.
- Report and prioritize any issues identified.

System testing (3-5 days):

- Test the application as a whole, including its interactions with external systems.

Release testing (2-3 days):

- Re-run previous test cases to ensure new code changes do not affect existing functionalities.
- Verify fixes for reported issues.
- Document and track regression test results.

2.3 Release Criteria

For all tests, the maximum fault tolerance is set at a level where critical functionalities must operate without failure in at least 95% of test cases. This ensures a high degree of reliability and stability in the application. Before deployment, the system must meet the criteria of passing all test cases with a success rate of 98% or higher, demonstrating robustness, performance, security, and usability across all key features and scenarios. Additionally, any critical defects identified during testing must be addressed and verified as resolved before the deployment phase.

3 Unit Testing

Performing unit tests is crucial as it helps ensure the reliability and correctness of individual components or units of code within the application. Unit tests verify that each unit behaves as expected, isolating and testing specific functionalities in a controlled environment. This helps identify bugs early in the development process and facilitates easier debugging and maintenance.

In our testing process, we conducted unit tests using the Jest testing framework, which is widely used for testing JavaScript code, particularly in React applications. The outcome of the unit tests is either a pass or fail generally, and should be mostly, if not completely positive, with the majority of tested components passing their respective test cases. The unit tests focused on testing functions, methods, and components at the granular level, ensuring that they perform as intended and meet specified requirements.

3.1 User Authentication Module

The User Authentication Module is responsible for managing user registration, login, and logout functionalities within the messaging application. It validates user credentials, generates authentication tokens, and maintains session information.

3.1.1 User Registration

The test case focuses on validating the user registration process to guarantee the successful creation of accounts. It includes inputs such as user details (username, email, password) and expects either a successful registration message or a confirmation email sent to the user. Sub-cases involve testing with valid user credentials, invalid email format, and passwords not meeting minimum requirements. The pass criteria necessitate that users receive a confirmation message or email upon successful registration, and their account information is securely stored in the database.

3.1.2 User Login

The purpose of this test case is to verify the user login process, ensuring that registered users can securely access the application. It involves inputting user credentials (username/email and password) and expects a successful login message along with an authentication token being generated. Sub-cases include testing with valid user credentials, invalid username/email, and incorrect passwords. The pass criteria require users to receive a valid authentication token upon successful login and to be redirected to the application's dashboard or home screen.

3.2 Message Sending Module

The Message Sending Module is responsible for managing the composition and transmission of messages among users within the messaging application. Its primary function is to guarantee the timely delivery of messages while maintaining secure storage in the message history. Testing focuses on several key areas: confirming users' ability to compose and send messages to one another, ensuring real-time delivery and synchronization across various devices, and validating the encryption of messages during transmission and their secure storage in the database. The release criteria outline specific expectations: messages must be sent and received promptly, message content must be encrypted during transmission and securely stored, and users must have access to their message history to retrieve previous conversations.

3.2.1 Create message

This test case focuses on validating the functionality of composing and sending messages within the application. It requires inputs such as the sender's user ID, recipient's user ID, and the message content, and expects a confirmation indicating successful message transmission. Sub-cases involve testing with valid sender and recipient IDs, empty message content, and special characters in the message content. The pass criteria mandate that messages are sent and promptly delivered to the recipient without delay, with the sender receiving a confirmation message indicating successful delivery.

4 Feature Testing

Performing feature tests is significant as it allows us to verify that the key functionalities of the messaging application work as expected from an end-user perspective. Feature testing ensures that the application meets the specified requirements and delivers the intended user experience, enhancing user satisfaction and confidence in the product.

For our feature testing, we used the Cypress testing framework, which is well-suited for end-to-end testing of web applications. Cypress provides a robust set of tools for writing and executing feature tests, allowing us to simulate user interactions and validate application behavior across different scenarios.

Our feature tests covered a range of key features, including user authentication, messaging, contact management, and notifications. Each test case simulated user actions such as logging in, sending messages, adding contacts, and receiving notifications, while verifying that the application responded correctly to these actions.

Ideally more time is required to fully test edge cases and other aspects of our messaging app, Discom, but given the time constraints, we are satisfied with the outcome.

4.1 User Authentication

The User Authentication feature serves to safeguard access to the messaging application by authenticating users throughout the login and registration procedures. It encompasses functionalities such as user registration, login, logout, and password reset. Testing of this feature is designed to ensure various aspects: user registration validation, successful user login with valid credentials, secure logout, and a smooth password reset process. Minimum release criteria stipulate that all authentication-related functionalities should operate without errors in the majority of test cases, at least 98%. Additionally, user credentials must be securely stored and transmitted, adhering to best practices for authentication and data protection. Moreover, the user authentication process is expected to be intuitive, user-friendly, and dependable.

4.1.1 Successful User Login

The "Successful User Login" test case aims to validate the login process of a user into the application. It requires valid inputs, namely a username and password, and expects the user to be logged in successfully and directed to the application's dashboard. Sub-cases involve authenticating the entered credentials against stored user data and ensuring redirection to the correct page post-login. Pass criteria include the successful login of the user without encountering errors and the secure maintenance of the user session until logout, allowing uninterrupted access to the application's features.

4.1.2 Successful User Login

The "Failed User Login (Invalid Credentials)" test case is designed to assess the handling of unsuccessful login attempts due to invalid credentials within the application. It requires inputs of an invalid username or password and expects the login attempt to fail, accompanied by the display of an error message indicating the invalid credentials. Sub-cases involve verifying the system's secure detection and handling of invalid login attempts and ensuring the display of appropriate error messages to the user. Pass criteria include the user receiving an error message indicating the invalid credentials, and the system successfully preventing unauthorized access while maintaining security measures.

4.2 Contact Management

The Contact Management feature in the messaging application provides users with tools to efficiently manage their contacts, facilitating seamless communication with acquaintances, colleagues, and friends. It encompasses functionalities such as adding, removing, and viewing contacts. Testing of this feature concentrates on interaction tests to ascertain users' ability to perform contact management tasks smoothly. The minimum release criteria entail users' ability to add new contacts with valid information, the successful removal of contacts without errors, and ensuring the accuracy and real-time updating of the contact list display.

4.2.1 Add Contact

The purpose of this test case is to validate the successful addition of new contacts by users within the application. Inputs include the contact name and either the contact's email or username. The expected output is the addition of the contact to the user's contact list. Sub-cases involve testing the addition of contacts with valid and unique information, along with verifying the display of appropriate error messages for invalid inputs. Pass criteria stipulate that the contact should be added to the user's contact list in the database, and it should promptly appear in the user's contact list interface without delay.

4.2.2 Remove Contact

The purpose of this test case is to confirm users' ability to remove contacts from their contact list within the application. Inputs include specifying the contact to be removed. The expected output is the removal of the contact from the user's contact list. Sub-cases involve testing the removal of contacts individually and in bulk, along with validating the display of appropriate confirmation dialogs before removal. Pass criteria require that the contact is successfully removed from the user's contact list in the database and no longer appears in the user's contact list interface after removal.

5 System Testing

The system testing process begins with System Test Planning, which involves defining test objectives, scope, and criteria, along with identifying test scenarios based on both functional and non-functional requirements. Additionally, security testing activities are planned, including the identification of potential vulnerabilities and attack vectors. Test Environment Setup follows, where test environments are configured to replicate production settings, and necessary tools, including security testing tools for vulnerability scanning and penetration testing, are deployed. Functional Testing ensues, focusing on core functionalities such as user authentication, messaging, contact management, and notifications. System behavior is verified under normal and abnormal conditions, such as network disruptions or concurrent user loads. Performance Testing is then conducted to assess system performance under varying conditions, measuring response times, throughput, and resource utilization to identify bottlenecks and optimize performance. Security Testing is a crucial step, involving vulnerability scanning to detect potential security flaws and penetration testing to simulate real-world attacks and evaluate the system's resilience. Mitigation of identified vulnerabilities is carried out through code fixes, security patches, and configuration changes. Finally, Risk-Based Security Testing prioritizes testing efforts based on the severity and likelihood of potential risks, performing risk analysis to identify critical assets, threats, and vulnerabilities, and allocating resources accordingly, with a focus on securing high-risk areas first.

The outcome of System Testing for Discom is promising across various aspects. In terms of Functional Testing, all core functionalities have been rigorously tested and found to operate as intended, with no critical defects affecting user experience or system stability detected. Performance Testing reveals satisfactory performance under diverse conditions, with acceptable response times and throughput, and identified bottlenecks have been effectively addressed to ensure optimal system performance. Security Testing has been comprehensive, with vulnerability scanning and penetration testing uncovering and resolving potential security vulnerabilities, safeguarding against common threats like injection attacks, XSS vulnerabilities, session hijacking, and brute force attacks. Through Risk-Based Security Testing, high-risk areas have been meticulously scrutinized and fortified, reducing the likelihood of security breaches and data compromises. Overall, Discom demonstrates resilience against security threats, ensuring the safety and integrity of user data and interactions, with robust security controls and measures in place to protect critical assets and mitigate potential risks effectively.