Daniel Schuster

Term Project: Instant Messaging App

Test Plan Document

Table of Contents

1 Introduction

This test plan outlines the testing strategy for the Instant Messaging App, focusing on backend functionalities including authentication, contact management, and real-time messaging.

1.1 Purpose

The purpose of this document is to ensure systematic and comprehensive testing of the app, identifying potential issues before deployment.

1.2 Target Audience

This document is intended for developers, testers, and project stakeholders involved in the Instant Messaging App project.

1.3 Terms and Definitions

- **JWT:** JSON Web Token, used for authentication.
- **MERN Stack:** MongoDB, Express.js, React.js, Node.js.
- **Real-Time Messaging:** Instant delivery of messages between clients.

2 Test Plan Description

This section covers the scope, schedule, and release criteria for testing.

2.1 Scope of Testing

The testing will focus on backend components including authentication, database interactions, chat room management, and real-time messaging. Testing will involve individual unit tests, feature tests, and system tests.

2.2 Testing Schedule

Outline your testing schedule here. You are encouraged to design your own testing schedule template.

- **Week 1-2:** Unit Testing
- **Week 3:** Feature Testing
- **Week 4:** System and Security Testing

2.3 Release Criteria

The application will be ready for deployment when all critical test cases pass with zero major bugs and at least 90% coverage.

3 Unit Testing

Unit testing will be performed using **Jest** and **Supertest** frameworks. It will cover the individual units of the backend software to ensure each unit works properly in an isolated environment. Each unit is generally considered to be a single endpoint of the API for the purposes of this document.

3.1 User Signup

3.1.1 Valid Signup

Valid signup requires entering an email that is not already in use, a password, and a matching password in the confirm password dialog box.
Inputs: unused email, password, and matching password for confirming password
Output: 201 Created, user registered successfully

3.1.2 Invalid Signup

No email and/or no password entered:
Inputs: test using blank email, then blank password, then both blank
Output: 400 Bad Request, email and password are required

Email already in use:
Inputs: email that already is connected to an account in the database, a password
Output: 409 Conflict, email already in use

Server/database error:
Inputs: error thrown when attempting to query database
Output: 500 Internal Server Error, server or database issue

3.2 User Login

3.2.1 Valid Login

Valid login requires entering a correct email and password of a user registered in the database.

Inputs: valid email and password of registered user

Outputs: 200 OK, login successful, and a JWT token

3.2.2 Invalid Login

No email and/or password entered:

Inputs: test with blank email, then blank password, then both blank

Output: 400 Bad Request, missing email or password

No registered user that matches email entered:

Inputs: email that is not in the database, a non-blank password

Output: 404 Not Found, no user found with the given email

Invalid password:

Inputs: email of a registered user, incorrect password for that user

Output: 400 Bad Request, invalid password

Server/database error:

Inputs: error thrown when attempting to query database

Output: 500 Internal Server Error, server or database issue

3.3 User Logout

3.3.1 Valid Logout

Input: logout request for a logged in user

Output: 200 OK, logout successful

3.3.2 Invalid Logout

Input: logout request for a logged in user, error thrown connecting to server

Output: 500 Internal Server Error

3.4 Get User Info

3.4.1 Valid User Info Request

Input: User ID of a valid in-token user

Output: 200 OK, JSON response with id, email, and password of current user

3.4.2 Invalid User Info Request

No in-token user:

Inputs: no token, or invalid token

Output: 404 Not Found, user not found

Server/database error:

Inputs: error thrown when attempting to query database

Output: 500 Internal Server Error, server or database issue

3.5 Update User Profile

3.5.1 Valid User Update

Inputs: non-blank first name, last name, and color, valid user ID

Output: 200 OK, profile successfully updated

3.5.2 Invalid User Update

Missing Required Fields:

Inputs: Test with all combinations of blank first name, blank last name, blank color

Output: 400 Bad Request, missing required fields

No Matching User:

Inputs: valid first name, last name, and color, but invalid user ID

Output: 404 Not Found, user not found

Server/database error:

Inputs: error thrown when attempting to query database

Output: 500 Internal Server Error, server or database issue

3.6 Search Contacts

3.6.1 Successful Contacts Search

Inputs: valid keyword to filter users

Output: 200 OK, and array of contacts that match the search query

3.6.2 Unsuccessful Contacts Search

Blank Search:

Inputs: empty keyword in request

Output: 400 Bad Request, missing search term

Server/database error:

Inputs: error thrown when attempting to query database

Output: 500 Internal Server Error, server or database issue

3.7 Get Contacts

3.7.1 Valid Contacts Request

Input: valid token of current user

Output: 200 OK, and an array of the current user's contacts

3.7.2 Invalid Contacts Request

Inputs: error thrown when attempting to query database

Output: 500 Internal Server Error, server or database issue

3.8 Delete Messages

3.8.1 Valid Delete Message Request

Input: valid dmID of a message for the current user

Output: 200 OK, message deleted successfully

3.8.2 Invalid Delete Message Request

Bad dmID:

Input: Test using empty dmID and dmID with no matching message

Output: 400 Bad Request, missing or invalid dmID

Server/database error:

Inputs: error thrown when attempting to query database

Output: 500 Internal Server Error, server or database issue

3.9 Get Messages

3.9.1 Valid Get Message Request

Inputs: valid user ID of current user and of the contact being messaged

Output: 200 OK, array of messages between current user and the selected contact

3.9.2 Invalid Get Message Request

Missing User ID:

Inputs: Missing or invalid user ID for current user and/or the contact

Output: 400 Bad Request, missing one or both user IDs

Server/database error:

Inputs: error thrown when attempting to query database

Output: 500 Internal Server Error, server or database issue

4 Feature Testing

Feature testing is vital to ensure that units are working together as expected to provide each implemented feature of the system. It will ensure that API endpoints are working together as expected.

4.1 User Authentication

User authentication involves testing user signup, login, logout, getting user info, and updating user info. This will involve interaction tests to ensure the authentication units function together, and usefulness tests to ensure that authentication is intuitive for users.

4.1.1 Valid Authentication

This feature test will include signup, login, getting user info, updating user info, then logging out for a single account, in that order. It will ensure that all authentication units are working together to allow all expected authentication actions for a user.

4.1.2 Invalid Authentication

This will include inserting incorrect input at each stage of the process of signup, login, getting user info, updating user info, and logging out (one at a time) to ensure the correct errors are generated.

4.2 Contact Management

4.2.1 Successful Contact Management

This feature test will include verification of the ability to retrieve a user's contact list from the database, and search the contact list using keywords. It will ensure that all contact management units are working in conjunction as expected.

4.2.2 Unsuccessful Contact Management

This will include attempting to retrieve contacts with a blank search term as well as attempting to get the contact list for a invalid user that does not exist in the database to ensure the correct errors are generated.

4.3 Real-Time Messaging

Test the socket-based chat handler to ensure correct interaction between units when direct messages are being sent and received, as well as usefulness of the feature to the user.

5 System Testing

System testing is important to demonstrate the functionality of the system as a whole. System testing will include end-to-end testing of all features, along with security testing, to include testing for:

- **Injection Attacks:** Test against SQL/NoSQL injections.
- **Cross-Site Scripting (XSS):** Validate input sanitization.
- **Session Hijacking:** Ensure secure token management.
- **Brute Force Attacks:** Implement rate-limiting and password protection.