TED UNIVERSITY

CMPE 492

Senior Design Project II

MESAJLA

Test Plan Report

16/04/2025

Group Members:

- **Ahmet Tunahan Küçükgökce**
- **Altar Gürsoy**
- **Burak Eren Birinci**
- **Serdar Kemal Topkaya**

Table of Contents

1 Test Planning

1.1 Scope of Testing

This test plan covers both functional and non-functional requirements of the messaging app Mesajla. The plan consists of tests like unit, system and integration testing for functional requirements also performance, user acceptance and beta testing for non-functional requirements. It will include testing over major messaging features, AI interactions, and UI behaviour. The plan further includes system performance verification, usability testing, and part security testing, precisely data movement within the messaging and AI systems.

In Scope:

- Core messaging functionality (send, receive, delete)
- AI message suggestions and smart replies
- Message read receipts and typing indicators
- Login, registration, and session management
- Notifications (push and in-app)
- Media file attachments
- Basic performance and responsiveness testing

1.2 Features to be Tested

- **Messaging System**
    - Real-time message delivery
    - Message editing and deletion
    - Media attachments (images, documents)
    - Typing indicators and read receipts
- **AI Engine**
    - Smart replies and message suggestions
    - AI spam detection and filtering
- **User Management**

    - Email and social sign-up/login
    - Password recovery and validation
- **Notification System**
    - Push notifications for new messages
    - In-app alerts for typing and status updates
- **General UI/UX**
    - Accessibility and usability

1.3 Testing Objectives

- Ensure reliable and timely delivery of messages in 95% of use cases.
- Validate AI-generated message suggestions for contextual accuracy and tone.
- Confirm robustness of login/authentication workflows.
- Measure system performance under load (10 concurrent users).
- Identify and resolve UI/UX inconsistencies.
- Validate data integrity during message sync and retrieval.

2 Testing Strategy

The testing operations for this chatting application consist of elements to find out whether it meets the expectations successfully. To achieve that, there are different testing strategies used for different scenarios. These strategies check functional and/or non-functional requirements such as performance, scalability, reliability, security and so on. It is also helpful to detect errors and bugs in the development phase of the application.

These software testing techniques analyze the application from the roots of it. Each of them is viable to do certain test operations for the application. For different sections of the application, there are different methods used to test them. Here are the types of testing currently planned to use:

1. **Unit Testing**

   With the use of unit tests, the components of system and functions are tested individually. These tests have been applied to the application since the beginning of the development process.

   Goals and/or tasks of unit testing in the application:

   - Testing AI model by the inputs and verifying the accuracy.
   - Verifying the overall user authentication methodology.
   - Ensuring the propagation of real-time messages in chat management System.

2. **Integration Testing**

   The main focus of integration tests is to configure how the subsystems of architecture work together collaboratively. It ensures the functionalities of different classes blend in well in the same environment.

   Goals and/or tasks of integration testing in the application:

   - Testing the harmonization between front-end, back-end and database.
   - Data collection integration between the user and the AI model's database.
   - Synchronizing the peer-to-peer communication across the subsystems.

3. **Security Testing**

   It is used to detect if there are any vulnerabilities in the security of the users and the application. It gives recommendations to developers to solve these identified risks.

   Goals and/or tasks of security testing in the application:

   - Testing the leakage of any data both in application and the personal information of users
   - GDPR verification controls in the application.
   - Ensuring the security of the database from outside 3rd parties.

4. **Performance Testing**

   This test is to figure out how the system performs under the applied workload. Methods such as stress testing is used in the performance tests that sends data traffic to the system and analyses the bottlenecks, stability and so on.

   Goals and/or tasks of performance testing in the application:

   - High user load simulations in the system.
   - Aiming to reduce chatting response delay with the help of these tests.
   - Providing a stable system.

5. **Usability Testing**

   Usability tests focus on the users' experience of the system. It aims to enhance user satisfaction.

   Goals and/or tasks of usability testing in the application:

   - Collecting feedback from users to improve the UI to using the functionality in the front-end is smoother.
   - Testing the AI model to check whether it meets user expectations.
   - Improving the user profile management system.

6. **Boundary (Black-Box) Testing**

   Boundary testing is used to detect the error and exceptions caused by the valid and invalid inputs. It shows what your application approves and doesn't.

   Goals and/or tasks of performance testing in the application:

   - Testing certain characters while filling up the log-in or sign-up credentials.

- Using abbreviations while training the AI to check if it understands their meanings.

7. Functional Testing

This test analyses whether the software requirements are fulfilled in the functionalities.

Goals and/or tasks of functional testing in the application:

- Testing the system responds correctly in user authentication, controlling the credentials for each user.
- Testing the navigation of addresses on the website working as intended.
- Checking the functionalities of buttons, pagination etc. is successfully applied.

3 Test Scenarios

User Registration and Sign-in

TS01: Valid User Registration

- Purpose: Verify that any user has successfully completed the registration process with a valid email, password and security question.
- Steps: Go to the Sign-up page. Enter a valid email and password. Select a security question and enter the answer. Verify that the triangle button is active. Click it to successfully register.

TS02: Triangle Button Inactive in Incomplete Form

- Purpose: Verify that the triangle "next" button is inactive until you fill in the information required for registration.
- Steps: Navigate to the registration page, make sure at least one field is empty, check the status of the triangle button.

TS03: Redirect to Profile Settings after Registration

- Purpose: After successful registration, verify that the user is redirected to the profile setup page.
- Steps: Complete the registration process, then check if you are correctly directed to the profile page.

TS04: Valid Login Process

- Purpose: Verify that a registered user has successfully logged in with the correct credentials.
- Steps: The login page was visited, the correct user information was entered, the triangle button was clicked and the navigation was checked to see if it was successful.

TS05: Navigating from Login Page to Registration Page

- Purpose: Verify that clicking the “Sign Up” button redirects to the registration page.
- Steps: Click on the sign up button, check that you are directed to the registration page.

Profile Setup

TS06: Profile Settings

- Purpose: Verify that the user cannot continue unless they enter their first and last name
- Steps: After completing the registration process, check the automatic redirection.

TS7: Cannot Continue with Empty Profile Fields

- Purpose: Verify that the user cannot proceed unless name and surname are filled.
- Steps: Do not enter input in the name and surname fields, check the triangle button.

TS8: Verification of being redirected to the chat page after completing the profile.

- Purpose: Verify that after the name and surname are filled, to navigate to the chat page, the triangle button activates.
- Steps: Enter your name and surname, click the triangle button and check if it goes to the chat page successfully.

Real-time Messaging Functionality

TS9: Real-time Messaging

- Purpose: Verification of whether the user can send real-time messages during chat.
- Steps: Two separate accounts were logged in using two different devices. Messages were sent and instant access was checked.

TS10: Correct Message information

- Purpose: Verify that users' profile pictures, name, and online status can be viewed.
- Steps: First, a message was sent, then the sender's information was checked.

TS11: Add friends and start a conversation.

- Purpose: Verify that the user can successfully add friends and start a new chat.
- Steps: The add friend button was used and a conversation was started with the added friend.

TS12: Correct ordering of messages

- Purpose: Verify the order and timing of messages in the chat interface.
- Steps: Check the order of messages.

TS13: Sending Emoji and Files

- Purpose: Verify that emoji and files can be sent without any problems.
- Steps: Send emoji and files in chat. Then check that they are displayed correctly.

TS14: Persistence of message conversation

- Purpose: Verify that messaging history across sessions is retained.
- Steps: A message was sent to a user. The session was logged out and back in. Checked if the chat history was still there.

AI-generated Message Suggestions

TS15: AI Message Suggestions Appear

- Purpose: Verification that AI-generated message suggestions are displayed during chat.
- Steps: Chatting between users has started. Ai suggestions were checked under the chat writing field.

TS16: The selected AI message is sent correctly.

- Purpose: Clicking on a suggested message and verifying that it has progressed successfully in the chat.
- Steps: One of the suggested messages was selected. It was checked whether the message was delivered correctly.

TS17: Refreshing AI Message Recommendations

- Purpose: Verify that clicking the cross icon changes the message suggestions.
- Steps: For any suggestion, the cross symbol was clicked. It was checked whether there were new suggestions.

TS18: Multiple AI Recommendation Display

- Purpose: Verify that at least 2-3 recommendations are displayed.
- Steps: Enter the chat, check the number of suggested AI messages.

TS19: Not selecting suggested AI messages

- Purpose: Testing the system's response to new messages when AI suggestions are not used.
- Steps: AI message suggestions were not used for a few messages. Checked if new AI message suggestions were coming for other messages.

Group Chat Functionality

TS20: Creating groups and adding users

- Purpose: Verified that the user can create a group and add other users to the group.
- Steps: Clicked on the "Create group option", selected the members to be added, then checked the group list to confirm the addition of group members.

TS21: Real Time Messaging in Group Chat

- Purpose: Verify that users can type and view real-time messages in group chat.
- Steps: A message was sent in the group chat. Checked whether other members can see the messages instantly.

TS22: Viewing Groups

- Purpose: Verification that the user can view the list of groups they are a member of.
- Steps: Joined several groups. Checked if all of them appear in the group list.

TS23: Leaving the group

- Purpose: Verify that users can leave the group without encountering any errors.
- Steps: Clicked on the “Leave Group” option, then checked if the group still exists.

UI and Navigation

TS24: Correct Behavior of Triangle Buttons that Provide Navigation Between Pages

- Purpose: Verify that the Triangle buttons that navigate between pages behave as they should. (Active = black, gray = inactive)
- Steps: Tests were performed on forms on different pages. After the form was completed, the status of the button was checked.

TS25: Responsiveness

- Purpose: Validation of responsiveness on different screen sizes.
- Steps: The application was opened on different devices. The placement of the components was checked.

TS26: UI Consistency

- Purpose: Verify consistent UI across pages.
- Steps: Sign-up, Sign-in, Profile Setup, Chat Page visited then colors, fonts and icons compared.

TS27: Accuracy of online status updates

- Purpose: Checking whether real-time status updates, online or offline, are reflected correctly to the user:
- Steps: One user was logged out. The other user checked if the other was offline.

4 Test Environment

A Reliable and good test environment is very important for ensuring the functionality, performance, and reliability of the “Mesajla” Chat-app. The environment will resemble the code setup to provide consistent test results and smooth operation.

Hardware Requirements

The test environment does not require special hardware, as the application works in a web browser and uses Firebase service. However, to provide reliable and efficient testing, we have used:

CPU: i7 11800H

RAM: 16GB RAM

Storage: At least 100 GB of free space (To ensure that we can freely use different testing techniques without worrying about space)

Devices: A Desktop or laptop PC for web testing.

Software Requirements

The following software components are important for testing:

Operating System: Windows 10+, macOS, or Linux

Browser: Latest version of Chrome, Firefox, or any other browser for web testing.

Code Editor: VS Code or any other IDE

Package Manager: Node.js with npm (It comes with node.js)

Database: Firebase Firestore (Storing user data)

Authentication: Firebase Authentication (For user data)

Performance Testing

This environment is used for seeing the app’s responsiveness, work under pressure, and how it performs overall.

Purpose: Measure load times, response speed, and how the app work in heavy workload.

Tools Used:

- Lighthouse: A browser tool that checks how fast and well the app performs.

Setup: Test Firebase project with a large dataset and big traffic.

Integration Testing Environment

In this setup, all modules (chat component, authentication, Firebase services) are integrated and tested as a complete whole app.

Purpose: Ensure the different parts of the application work together.

Tools Used:

- React Testing Library: A tool to check if buttons, inputs, and other parts of the app work as expected.
- Mock-up firebase project: Instead of using real Firebase, we create a fake version for testing, so we don’t affect real data.

Setup: Local development environment with Firebase, Firestore and Authentication.

Chaos Testing Environment

This environment introduces deliberate errors or failures to assess the application's stability.

Purpose: Identify how the app handles errors like network errors, Firebase errors or corrupted data.

Tools Used:

- Simulated network problems in Dev Tools
- Random data corruption.

Setup: Local environment with real-world problems.

Security Testing Environment

This environment is focused on identifying security problems and security injections.

Purpose: Checking the system's ability to prevent unauthorized access, code injection, and data leaks.

Tools Used:

- Security Scanner: A tool that checks the app for common security problems, like unsafe links or weak spots hackers might use.
- Firebase Security Rules simulator: Helps make sure that only the right users can access certain data in the app.

Setup: Creating Firebase project with intentional security errors (for testing) and mock user accounts.

Regression Testing Environment

This environment is used to determine if the new changes and updates negatively impacts the apps functionality.

Purpose: Ensure new features or bug fixes do not impact components in a bad way.

Tools Used:

- Testing Tool: A tool that clicks around the app like a real user to make sure everything still works after changes.
- GitHub: Automatically checks the app every time someone updates the code, showing the differences so we catch problems early.

Setup: The app is tested automatically whenever new code is added. This helps make sure new updates don't break things that are already working.

Test Data Preparation

Test data includes mock user accounts and messages to simulate real time chatting. This data will not be used with the main app and separated by using:

A separate Firebase project for testing, made up user information for login, register and messages, predefined chat data to test features like message delivery, notifications and real time updates.

Tool Setup

To ensure effective testing and automation, the following tools will be configured:

Testing Framework:

- Unit Testing: React Testing Library
- End to End: User to user interaction from 2 different computers.

Test Management:

- GitHub repository for tracking test results and bugs.

5 Schedule & Resource Allocation

2.1 Test Schedule with Timelines

Phase	Start Date	End Date	Duration	Notes
Test Planning	Apr 7, 2025	Apr 14, 2025	1 week	Define scope and strategy
Test Case Design	Apr 7, 2025	Apr 14, 2025	1 week	Manual and automated tests
Environment Setup	Apr 7, 2025	Apr 14, 2025	1 week	Staging and testing DB setup
Test Execution	Apr 20, 2025	May 12, 2025	3 weeks	Functional, AI, performance tests
Bug Reporting	Apr 20, 2025	May 12, 2025	3 weeks	In parallel with test execution
Final Report	May 12, 2025	May 19, 2025	1 week	Summary and stakeholder review

2.2 Roles & Responsibilities

Team Member	Role	Responsibilities
Altar	Automation Engineer	Build and maintain test scripts for messaging and AI
Serdar	Manual Tester	Execute test cases manually, perform exploratory testing
Burak	Environment Support	Setup and manage test environments
Ahmet	AI Quality Analyst	Evaluate AI responses for relevance, accuracy, and fairness

6 Test Control and Risk Management

Effective test control and risk management are important to ensure the testing is organized, efficient, and adaptable to changes. For the Mesajla Chat app, the following techniques will be used:

Test Control Procedure

1. Bug Tracking

All bugs and issues discovered during testing will be logged in GitHub Issues.

Each issue will be put into categories by importance and assigned to the team member for solving.

2. Test Progress Monitoring

A simple test log will track which test cases have passed or failed.

3. Version Control and Change Tracking

Git will be used to manage source code and test code versions.

Branching will help separate testing cases from development and production changes.

4. Test Reports

We are goint to create test reports after each test run,

- Number of tests done
- Pass and fail numbers
- List of issues
- Problems

Risk Identification

Some of the key risk factors for this project:

Firebase Service Limitations: Testing process can be affected by Free tier limits.

Data Error: Test and Application data can be mixed if not carefully separated.

Authentication Errors: Issues with Firebase Auth.

Browser Compatibility: The app may behave differently across browsers or different screen sizes.

Risk Avoidance Plans

To manage the risks listed above, the following strategies are going to be used:

Environment Isolation: Use a separate Firebase project only for testing, thus, data will not accidentally mix.

Detailed Test Scenarios: Writing tests which are include every day scenarios, such as chat messages and user settings changes.

Cross Browser Testing: Run end to end tests in different browsers and screen sizes to catch problems early.

Back-Up Plan: If something important stops working, we'll use mock tools to keep testing how the app behaves.

7 Glossary

Engineering Standards: Technical documents for rules.

Database: Where data is stored.

Front-End: The visual interface of the web application.

Back-End: The system that runs on the server side of the application.

TS: Test Scenarios

8 References

1. Institute of Electrical and Electronics Engineers. (1998). *IEEE recommended practice for software requirements specifications* (IEEE Std 830-1998). IEEE Standards Association. https://standards.ieee.org/ieee/830/1222/
2. Institute of Electrical and Electronics Engineers. (2000). *IEEE recommended practice for architectural description for software-intensive systems* (IEEE Std 1471-2000). IEEE Standards Association. https://standards.ieee.org/ieee/1471/2187/
3. Institute of Electrical and Electronics Engineers. (2009). *IEEE standard for information technology—systems design—software design descriptions* (IEEE Std 1016-2009). IEEE Standards A