TED UNIVERSITY

Faculty of Engineering

Computer Engineering Department

CMPE_492 Senior Design Project II

TEST PLAN REPORT

Name of the Project: JustiWise

Web Page URL: https://finalprojectjustiwise.github.io/CMPE_491/

Team Members:

Irmak Orhan

Ahmet Engin Büyükdığan

Aslı Algın

Ali Fuat Dündar

Supervisor: Tansel Dökeroğlu

Jury Members: Fırat Akba, Ayşe Yasemin Seydim

Table of Contents

1. Introduction

The Test Plan Report for the JustiWise project outlines a detailed and structured strategy to evaluate the reliability, security, functionality, and usability of a web-based legal simulation and AI-powered legal assistant platform. The JustiWise platform is designed to ease the legal burden on professionals and provide support for law students through simulations and document analysis. Testing plays a pivotal role in the project lifecycle, ensuring that both technical and user-facing aspects meet high-quality standards before deployment.

This report includes comprehensive information about the testing approach, including the features to be tested, testing methodologies, test data and tools, environment configuration, schedules, and the responsibilities of each team member. The testing process follows an Agile development methodology where continuous feedback and iteration help refine system components effectively.

2. Features

The key functional and non-functional features of the JustiWise platform that will be tested are as follows:

2.1. AI-powered Legal Assistant

- Users can upload legal documents.
- The system analyzes content using an LLM (Llama 3) and extracts legal terms.
- Offers personalized legal recommendations and summaries based on the uploaded content.

2.2. Legal Scenario Simulation and Virtual Courtroom

- A simulated courtroom environment where avatars represent roles such as judge, prosecutor, lawyer, and client.
- Scenarios include commercial case simulations pre-loaded with real-case datasets.
- Law students can take on different roles and respond to dynamic case events.
- AI provides automated feedback, evaluates argument strength, and helps measure student understanding.
- Designed to create an immersive learning and practice environment for legal education.

2.3. Legal Language Translator

- Translates user-friendly layman input into professional legal terms.
- Helps bridge the gap between clients and the legal system.
- Especially useful in commercial and civil cases where correct terminology is essential.

2.4. Admin Dashboard

- Enables monitoring of user behavior, tracking of completed simulations, and access to feedback logs.
- Includes moderation tools for flagging inappropriate behavior or malfunctioning AI responses.
- Provides real-time data visualization and usage reports.

2.5. User Roles and Authentication

- Role-based access control supports four user types: client, lawyer, student, and admin.
- JWT (JSON Web Token)-based authentication ensures secure login and session management.

2.6. Case Management (Lawyer Role):

- Tools for tracking case progress, managing documents, and planning tasks.
- Integration with AI Assistant outputs.
- Secure document storage and retrieval.

2.7. AI-Powered Avatar (Client Interaction):

- Ability to understand and process client narratives.
- Conversion of narrative input into structured legal language.
- Interaction flow and user experience.

2.8. Error Handling and Notifications:

- Effective display of error messages and notifications.
- Functionality of the integrated help system.

2.9. Non-Functional Requirements:

- **Security:** Encryption (end-to-end, database), hashing, protection against common vulnerabilities, secure authentication/authorization.
- **Performance:** Response times under load, simulation latency, handling of 1000 concurrent users.
- **Scalability:** System behavior as user load increases.

3. Testing Methodology

A multi-tiered approach is adopted to ensure coverage of all functional, non-functional, and user-centered requirements.

3.1 Unit Testing

- **Scope:** Core backend services such as user authentication, document parsing, AI prompt generation, and database access.
- **Tool:** JUnit
- **Purpose:** Validate correctness of logic in individual methods and services.
- **Type:** White-box testing (developers know the internal code).

3.2 System & Integration Testing

- **Scope:** Data flow from frontend to backend, AI modules, and database.
- **Tool:** Swagger UI, and manual test scripts.
- **Purpose:** Ensure complete workflow is functioning as intended.
- **Type:** Gray-box testing (partial knowledge of internal systems).

3.3 Performance Testing

- **Scope:** Simulation of up to 1000 concurrent users, simultaneous document uploads, real-time avatar response.
- **Tool:** Docker-based load agents.
- **Goal:** Identify performance bottlenecks, memory leaks, and response time issues.
- **Output:** Response time charts, memory utilization logs, thread dumps.

3.4 User Acceptance Testing (UAT)

- **Scope:** Performed by law students, lawyers, and legal consultants.
- **Method:** Usability surveys, structured feedback forms, think-aloud protocols.
- **Focus:** User satisfaction, clarity, legal usefulness, and ease of use.

3.5 Beta Testing

- **Scope:** Conducted in a real-like deployment with restricted access.
- **Goal:** Discover edge cases, behavior in diverse environments, and UI bugs.
- **Participants:** Selected users from each user persona group.

3.6 AI Model Testing:

Scope: Evaluating the performance, accuracy, fairness, and robustness of the AI components (Llama 3 based assistant, translator, simulation avatars). This includes **benchmarking** key AI capabilities against relevant standards and criteria.

- **Accuracy:** Correctness of legal term extraction, summarization quality (vs. human summaries), translation fidelity, recommendation relevance.
- **Robustness:** Handling of ambiguous input, noisy data, unexpected user queries, potentially adversarial inputs.
- **Fairness/Bias:** Checking for biases in recommendations, translations, or simulation interactions based on protected attributes (if applicable and identifiable in test data).

- **Performance:** Latency of AI responses (summarization, translation, avatar interaction).
- **Functionality:** Evaluating specific AI tasks like narrative-to-legal-term conversion, feedback generation in simulations.

Methods: Domain-specific datasets (legal corpora, simulation logs), standard NLP metrics (Precision, Recall, F1, ROUGE, BLEU), human evaluation (expert review for quality/relevance), fairness assessment toolkits (e.g., AIF360, Fairlearn - if applicable), targeted test cases for specific AI behaviors (e.g., hallucination checks). Benchmarking against established NLP datasets (where applicable to the legal domain, e.g., potentially adapting existing QA or summarization benchmarks), internal performance/accuracy benchmarks defined by the project, and human expert performance benchmarks (for tasks like summarization quality).

Purpose: Ensure AI components meet functional and quality requirements. Utilize benchmarking to objectively measure performance against defined standards and track improvements or regressions. Validate against defined acceptance criteria for AI performance, which serve as key internal benchmarks.

Type: Black-box/Gray-box testing (depending on access to model internals/logic).

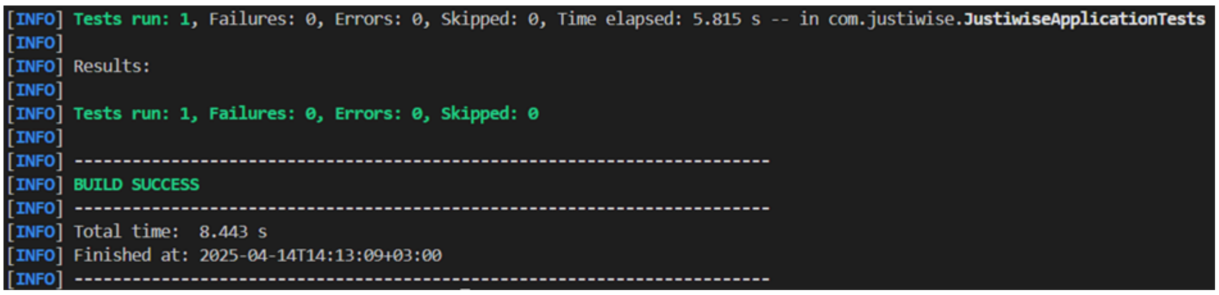
4. Test Environment & Tools

Frontend:

React web client on Chrome, Firefox, Edge, and mobile browsers.

Backend:

Java Spring Boot API running on cloud infrastructure. (Currently running locally, with planned deployment to cloud infrastructure.)



```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 5.815 s -- in com.justiwise.JustiwiseApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  8.443 s
[INFO] Finished at: 2025-04-14T14:13:09+03:00
[INFO] ------------------------------------------------------------------------
```

Database:
PostgreSQL database hosted on cloud servers.

AI Models:
Python-based NLP services (Llama 3) running on GPU-supported instances.

Category	**Tool**	**Purpose**
Unit Testing	JUnit (Java), PyTest (Python)	To test individual backend services (Java Spring Boot) and AI functions (Python).
Integration Testing	Postman,	To manually and automatically test RESTful APIs between subsystems.
System & Functional Testing	Cypress	To automate and validate end-to-end tests for the React.js frontend interface.
Performance & Load Testing	JMeter, Locust	To simulate concurrent users and test the system's performance under heavy load.
Security Testing	OWASP ZAP, Burp Suite	To identify security vulnerabilities, test encryption, 2FA, and JWT validation.
Bug/Issue Tracking	Jira, Trello	To log, track, and manage bugs or issues discovered during the testing process.
Version Control	Git, GitHub	To track changes and manage versions of test scripts, reports, and project code.
Continuous Integration (CI)	GitHub Actions, Jenkins	To automatically run tests on every code update and generate test reports.
Database Testing	pgAdmin (PostgreSQL)	To test queries, data integrity, and perform backup/restore tests on the PostgreSQL database.
Backup & Recovery Testing	AWS/GCP Backup Tools	To perform automatic backups and recovery scenario tests on cloud storage systems.
API Monitoring & Documentation	Swagger UI	To document, visualize, and test live RESTful APIs.

Log Monitoring	Logstash + Kibana	To collect, monitor, and analyze logs generated during test executions.
AI Model Evaluation & Benchmarking	Python Libraries (Scikit-learn, NLTK, spaCy), Custom Scripts, Human Review, Standard Benchmark Datasets (where applicable)	Evaluate & Benchmark AI accuracy, fairness, robustness using metrics & expert judgment.

5. Test Schedule

Phase	**Start Date**	**End Date**
Unit Testing	May 5, 2025	May 10, 2025
Integration Testing	May 11, 2025	May 16, 2025
System Testing	May 17, 2025	May 22, 2025
Performance and Security Tests	May 23, 2025	May 26, 2025
AI Model Testing & Benchmarking (Focused)	May 24, 2025	Jun 7, 2025
User Acceptance Testing (UAT)	May 27, 2025	May 30, 2025
Beta Testing	June 1, 2025	June 7, 2025

6. Control Procedures

- **Version Control:** Git and GitHub will be used for managing test scripts, test data, benchmark results, and reports.
- **Bug Tracking:** Jira (or specified tool) will be used for logging, tracking, prioritizing, and resolving bugs. Each bug report will include steps to reproduce, expected vs. actual results, severity, and environment details.
- · **Test Case Management:** Test cases will be documented (e.g., in Jira, TestRail, or a shared document - *specify tool*), linked to requirements, and their status (Pass/Fail/Blocked) tracked.
- **Change Management:** Changes to requirements during the testing phase will be evaluated for their impact on testing, and the test plan/cases will be updated accordingly.
- **Backup and Rollback:** Procedures for backing up the test environment and rolling back deployments will be in place, especially during integration and system testing phases.

7. Roles and Responsibilities

Role	**Responsibility**
Test Manager (Aslı Algın)	Prepare and manage the test plan
QA Engineers(Alifuat Dündar-Irmak Orhan)	Design and execute test cases, report bugs
Backend Developers(Aslı Algın-Ahmet Engin Büyükdığan)	Support defect fixing, unit testing
Frontend Developers (Aslı Algın -Ahmet Engin Büyükdığan)	Validate UI/UX, integration tests
AI Developers(Irmak Orhan-Alifuat Dündar)	Validate AI model performance and logic

8. Risks

Risk	**Description**	**Potential Impact**	**Mitigation Strategy**
AI Model Misbehavior	AI avatar providing incorrect, incomplete, or legally inaccurate advice.	Misinformation to users, legal liability, reduced trust and	Rigorous AI testing & benchmarking with diverse datasets, clear

		customer dissatisfaction.	disclaimers, feedback loops (RLHF), monitoring.
Performance Bottlenecks	System slowdown or crash under high concurrent user load.	Users unable to access services, data loss risk, platform reputation damage.	Realistic load testing, performance profiling & benchmarking, code optimization, infrastructure scaling.
2FA Service Unavailability	SMS/Email service downtime prevents delivery of one-time passwords (OTPs).	Users unable to log in or verify their accounts, UAT interruptions.	Use reliable third-party providers, implement retry logic, monitor service status.
Data Security & Compliance Breach	Leakage or improper handling of sensitive data such as passwords, legal documents, and personal details.	Legal penalties under KVKK/GDPR, user data compromise, loss of trust.	Security testing, code reviews, adherence to secure coding practices, encryption, access controls, regular audits.
Low UAT Participation	Insufficient number of lawyers, students, and support staff participating in user acceptance testing.	Lack of sufficient user feedback, undetected bugs in production.	Early recruitment, clear communication of value, incentives (if appropriate), flexible scheduling.
Backup or Recovery Failure	Failure to create backups or properly recover data after system errors.	Permanent data loss, operational downtime, customer dissatisfaction.	Regular automated backups, periodic recovery drills, use reliable cloud backup solutions.
Device/Browser Compatibility Issues	UI/UX issues on mobile, tablet, or desktop browsers.	Poor user experience, accessibility complaints, user drop-off.	Cross-browser/cross-device testing plan, use of browser compatibility tools/services.

Test vs Production Environment Differences	Differences between test and live environments causing issues post-deployment.	System errors in production, service disruptions for real users.	Strive for environment parity (Infrastructure as Code), configuration management tools
Deployment Misconfiguration	Deployment of the wrong version or missing configuration files.	Production system downtime, missing or malfunctioning features.	Automated deployment pipelines (CI/CD), configuration validation checks, rollback procedures.
Inadequate Test Data/Benchmarks	Lack of realistic/diverse test data, or relevant benchmarks for the legal AI domain.	Poor test coverage, undetected edge cases, biased AI performance, inaccurate quality assessment.	Data generation/augmentation, data anonymization, collaboration with legal experts, research suitable benchmarks, define internal ones.

9. Conclusion

This Test Plan Report outlines a comprehensive strategy for validating the JustiWise platform. It details the testing scope across functional and non-functional requirements, including specific methodologies for AI components, simulation features, performance, and security. By defining clear procedures, roles, schedules, tools, and risk mitigation strategies, this plan aims to ensure the delivery of a high-quality, reliable, and user-centric legal tech application that meets its objectives.

10. References

[1] https://www.acm.org/code-of-ethics/

[2] https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing