TEDU Assistant

Test Plan Report

CMPE 492 Senior Project II

Advisor	Yücel Çimtay
Jury	Aslı Gençtav
	Emin Kuğu
Members	Deniz Akşimşek
	Enes Akmil
	Gizem Kurtcuoğlu
	Mehmet Batuhan Şenol

Introduction

The Test Plan outlines the scope, approach, resources, and schedule for testing the TEDU Assistant[1] system. It identifies the features to be tested, testing methodologies, test environment, test schedule, control procedures, roles and responsibilities, and potential risks associated with the testing process.

Scope

The testing activities will cover the entire TEDU Assistant system, including both server-side and client-side modules. The primary focus will be on verifying the functionality, performance, and usability of the system. The following areas will be tested:

- Voice recognition and speech-to-text conversion
- Question parsing and context handling
- Response generation and retrieval of relevant information
- Web scraping and data extraction from university websites
- Integration between server and client components
- User interface and user experience

Testing Methodology

The testing will encompass multiple testing methodologies, including:

- Unit Testing: Individual units and components will be tested in isolation to ensure they function correctly.
- Integration Testing: The interactions and interfaces between different components or subsystems of a software system are tested to ensure they work correctly when integrated.

[1] Akşimşek, D. & Akmil, E. & Kurtcuoğlu, G. & Şenol, M.B., "TEDU Assistant Final Project Specification," 2022. <https://tedu-cmpe491-group10.netlify.app/reports/spec.pdf>

- System Testing: The integrated system will be tested to verify that all components work together as expected.
- User Acceptance Testing: Real users will be involved to assess the system's usability and gather feedback.
- Beta Testing: A subset of users will participate in a beta testing phase to identify any remaining issues before full deployment.
- Regression testing: New test cases can be added when issues are discovered to ensure that existing functionality remains unaffected and the defects are not re-introduced during development.

Test Environment

The test environment will consist of the following:

- Server: Python environment with the necessary dependencies and frameworks (Flask)
- Client: Web browsers compatible with the client-side components (e.g., Google Chrome)
- Test Data: Sample questions, contexts, and expected responses for various use cases
- Test Database: A separate database for testing purposes to avoid interfering with production data
- Speech Recognition: Access to a microphone or audio input device for speech recognition testing

Test Schedule

The testing activities will be carried out in the following phases:

- Unit Testing: Conducted concurrently with development, with each module tested as it is implemented.

- Integration Testing: Conducted after unit testing, verifying the correctness of components when integrated with one another.

- System Testing: Conducted after the completion of integration testing, focusing on end-to-end functionality. This phase may have multiple iterations as issues are identified and resolved.

- User Acceptance Testing: Conducted in collaboration with real users, involving usability evaluation and gathering feedback.

- Beta Testing: Conducted with a subset of users in a controlled environment to gather final feedback and identify any remaining issues.

In addition, regression testing may be conducted for new issues identified during development or testing activities.

Risks

The following potential risks and challenges associated with the test plan should be considered:

- Incomplete or ambiguous requirements: If the requirements are not clearly defined or are subject to change, it can lead to difficulties in test case development and result in incomplete testing coverage.

- Data availability: The system relies on data published on university websites. If the data sources are inaccessible, inconsistent, or change frequently, it may impact the accuracy and reliability of the system's responses.
- Integration issues: Since the system comprises both server-side and client-side components, any issues with integration between these components can affect the overall functionality of the system.
- Performance bottlenecks: The extensive use of caching and memoization for performance optimization may introduce complexities and potential bottlenecks that could impact system performance under high load conditions.
- Speech recognition accuracy: The accuracy of the speech recognition component may vary depending on factors such as ambient noise, user pronunciation, and speech patterns, which could affect the system's ability to accurately understand user input.

Test Cases

The test cases mostly follow an “arrange-act-assert” structure (also called “given-when-then”) which is derived from Hoare triples and popular in unit testing.

TA-TEST-001 Voice Recognition Accuracy

Type: System test

Given:

- A predefined audio input representing a user query

When:

- The voice recognition component is triggered with the audio input

Then:

- The recognized text output should closely match the expected text
- The accuracy of voice recognition should meet the predefined threshold

TA-TEST-002 Question Parsing and Response Generation

Type: Integration test

Given:

- A predefined user question covering different question types and topics

When:

- The user question is submitted to the system for parsing and response generation

Then:

- The system should accurately parse the question and identify the appropriate responder

- The generated response should be relevant, coherent, and correctly answer the user's question

TA-TEST-003 Web Scraping Functionality

Type: Unit test

Given:

- A set of URLs representing different web pages on the university website

When:

- The web scraping functionality is initiated to fetch data from each URL

Then:

- The system should successfully fetch data from the provided URLs
- The fetched data should match the expected content of each web page
- Unit Test for InternshipResponder: Test ID: TA-TC-004 Description: Verify that the InternshipResponder component fetches the correct internship coordinator for a given department.

TA-TEST-004 Internship responder

Type: Unit test

Given:

- A predefined user question regarding internship information
- Mocked data for the department and corresponding internship coordinator

When:

- The InternshipResponder is invoked with the user question and department context

Then:

- The InternshipResponder should fetch the internship coordinator data for the given department

- The fetched data should match the expected data for the internship coordinator

2. Unit Test for StaffInfoResponder: Test ID: TA-TC-005 Description: Verify that the StaffInfoResponder component fetches the correct basic information about academic and administrative staff from public profiles.

TA-TEST-005 Staff info responder

Type: Unit test

Given:

- A predefined user question about staff information
- Mocked data for the staff profiles and their basic information

When:

- The StaffInfoResponder is invoked with the user question

Then:

- The StaffInfoResponder should fetch the basic information about the academic and administrative staff
- The fetched information should match the expected data for the staff profiles

3. Unit Test for FinalExamsResponder: Test ID: TA-TC-006 Description: Verify that the FinalExamsResponder component fetches the correct dates of final exams.

TA-TEST-005 Final exams responder

Type: Unit test

Given:

- A predefined user question regarding final exams
- Mocked data for the dates of final exams

When:

- The FinalExamsResponder is invoked with the user question

Then:

- The FinalExamsResponder should fetch the dates of final exams
- The fetched dates should match the expected data for the final exams

4. Unit Test for AcademicCalendarResponder: Test ID: TA-TC-007 Description: Verify that the AcademicCalendarResponder component fetches the correct dates from the academic calendar.

TA-TEST-006 Academic calendar responder

Type: Unit test

Given:

- A predefined user question related to the academic calendar
- Mocked data for the dates and events in the academic calendar

When:

- The AcademicCalendarResponder is invoked with the user question

Then:

- The AcademicCalendarResponder should fetch the relevant dates from the academic calendar
- The fetched dates should match the expected data for the academic calendar

5. Unit Test for DepartmentInfoResponder: Test ID: TA-TC-008 Description: Verify that the DepartmentInfoResponder component fetches the correct information about departments and faculties.

TA-TEST-007 Department info responder

Type: Unit test

Given:

- A predefined user question about department information
- Mocked data for the departments, faculties, and their corresponding information

When:

- The DepartmentInfoResponder is invoked with the user question

Then:

- The DepartmentInfoResponder should fetch the information about the requested departments and faculties
- The fetched information should match the expected data for the departments and faculties

6. Unit Test for OpenedCoursesResponder: Test ID: TA-TC-009 Description: Verify that the OpenedCoursesResponder component fetches the correct list of opened courses.

TA-TEST-008 Opened courses responder

Type: Unit test

Given:

- A predefined user question regarding opened courses
- Mocked data for the list of opened courses

When:

- The OpenedCoursesResponder is invoked with the user question

Then:

- The OpenedCoursesResponder should fetch the list of opened courses
- The fetched list should match the expected data for the opened courses

TA-TEST-009 User acceptance testing

Type: Acceptance test

Potential users engage in conversation with the chatbot. They are asked to evaluate the system and answer questions:

1. On a scale of 1 to 5, how would you rate the overall usability and user-friendliness of the chatbot system?
2. Did you find the chatbot's responses helpful and relevant to your queries?

3. How would you rate the accuracy of the chatbot's responses to your questions?
4. Were there any instances where the chatbot failed to understand your question or provided incorrect information? If yes, please provide details.
5. How satisfied were you with the response time of the chatbot?
6. Were you able to easily navigate and interact with the chatbot system?
7. Did the chatbot prompt for clarification when your query was ambiguous or not clear? If yes, how effective was the clarification prompt?
8. Were you able to obtain the information you were looking for through the chatbot system?
9. How would you rate the system's ability to handle multiple user queries within the same conversation?
10. Did you encounter any technical issues or errors while using the chatbot system? If yes, please describe the issue.
11. Do you have any suggestions or improvements for enhancing the chatbot system's functionality or user experience?
12. Overall, how satisfied are you with the chatbot system?

TA-TEST-010 Open beta

Type: Beta testing

The system is made publicly available for use.

Conclusion

The Test Plan provides an overview of the testing activities for the TEDU Assistant system. It outlines the scope, approach, resources, and schedule for testing, as well as the identified risks. By following this plan, the testing process can be conducted effectively to ensure the system's functionality, performance, and usability are thoroughly evaluated before deployment.