**CS673 Software Engineering** 

**Team 1 - BU Academic Navigator (BUAN)**

**Software Test Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Natasya Liew | Team Leader | *Natasya Liew* | September 2, 2024 |
| Natthaphon Foithong | Design and Implementation Lead | *Natthaphon Foithong* | September 7, 2024 |
| Ananya Singh | Security Lead | *Ananya Singh* | Sep 3, 2024 |
| Battal Cevik | QE Lead | *Battal Cevik* | September 3, 2024 |
| Poom Chantarapornrat | Requirement Lead | *Chan P.* | September 3, 2024 |
| Yu Jun Liu | Configuration Lead | *Yujun Liu* | September 7, 2024 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| 1.0.0 | Battal Cevik | 09/15/2024 | Created Testing Summary, Added Manual Testing Report Style |
| 1.0.1 | Natasya Liew | 09/22/2024 | Proofreading and adjusting content. Added glossary and references section. |
|  |  |  |  |

[Testing Summary](#_heading=h.gjdgxs)

[Manuel Tests Reports](#_heading=h.30j0zll)

[Automated Testing Reports](#_heading=h.1fob9te)

[Testing Metrics](#_heading=h.3znysh7)

[References](#_heading=h.2et92p0)

[Glossary](#_heading=h.tyjcwt)

# Testing Summary

This section outlines the comprehensive testing strategies employed for the BUAN chatbot-driven web application. Our approach incorporated various testing methodologies to validate the system’s adherence to the design goals and requirements specified in the SDD and SPPP sections. The BUAN Academic Advising Chatbot web application underwent a comprehensive testing process to ensure robust functionality, reliability, and performance across all system components, including the frontend, backend services, AI integration, and database layers. Our testing strategy encompassed unit, integration, system, acceptance, and regression testing methodologies. The following components were subject to rigorous testing:

* **Frontend (React, Redux, and Fetch API)**
  + BUAN Chatbot Interface: Tested for smooth user interactions and appropriate response rendering. Automated tests were implemented to simulate user conversations and assess the accuracy of responses.
  + User Interaction Flow: Verified the logical progression and intuitiveness of the user journey through the web application.
  + Session Management: Verified the handling of user sessions, including session creation, validation, and expiration. Automated tests were set up to simulate user login/logout scenarios and assess session persistence.
* **Backend (Spring Boot)**
  + RESTful APIs: Validated the functionality and reliability of APIs for course management operations (CRUD operations) and request processing from the frontend. Each API endpoint underwent unit testing to ensure they returned the expected functionality and handle edge cases appropriately.
  + Data Processing: Ensured accurate transformation and handling of data between the front end, database, and Python3 AI service layers. Integration tests were conducted to validate data flows and the correctness of operations involving multiple components.
* **Backend (Python3 AI Service using OpenAI 4.0 via Langchain API and Langchain for RAG, chat history generation, and session management tooling)**
  + Custom Course Builder Algorithm: Tested to ensure the generation of accurate course recommendations for the upcoming semester. We performed algorithmic testing with various input scenarios to validate the recommendation logic.
  + Session Management: Verified the handling of chat sessions, including session creation, validation, and expiration. Automated tests were set up to simulate new chat creation scenarios and access session persistence.
  + Chat History Generation: Validated the retrieval of chat history via Langchain. We conducted functional tests to ensure user interactions with the BUAN chatbot were accurately logged and retrievable on subsequent sessions.
* **Databases (Postgresql)**
  + Chat History Storage: Verification of chat history storage was performed using unit tests to ensure that all interactions were saved correctly and that retrieval queries returned the expected results. We also conducted stress tests to assess performance under heavy loads.
* **Authentication (Okta)**
  + User Login: Authentication mechanisms were thoroughly tested to ensure security and usability. This included testing various login scenarios (successful login, failed login, password recovery) and verifying the integration with the Okta API for OAuth 2.0 flows.

**Who is involved:**

* **Development Team:** Comprising software developers responsible for building the application.
* **Quality Assurance Engineers:** Focused on ensuring the quality and functionality of the application through rigorous testing methodologies.

## Testing Techniques Used

**Unit Testing**Unit testing focused on evaluating individual components, including the React user interface, backend services built with Spring Boot, and Python decision algorithms. The following frameworks were employed:

* **Jest** for testing React components,
* **JUnit** for Spring Boot services,
* **PyTest** for Python services.

**Integration Testing**Integration testing assessed the interactions among the frontend, backend, AI service, and databases (PostgreSQL). This phase ensured that all system components functioned correctly in conjunction with one another, facilitating seamless data flow and interaction.

**System Testing**System testing validated the comprehensive functionality of the web application by examining the integrated systems as a whole. This included testing the user interface, backend processes, database interactions, and the AI service, ensuring all components worked together as intended.

**Acceptance Testing**Acceptance testing aimed to confirm that the application met all functional requirements. Key areas of focus included user authentication, course recommendations, and chat history management. Tests were conducted to ensure that the user experience aligned with expected outcomes and usability standards.

**Regression Testing**Following the implementation of new features, such as chat history sharing, regression testing was performed to verify that existing functionality remained intact. This phase concentrated on critical areas, including authentication processes, recommendation accuracy, and chat history retrieval.

**Testing Results**The majority of components successfully passed the testing phases, with minor issues identified, particularly concerning the integration between the courses and programs CSV files, the prompt library JSON file, the custom-built tree structure for course recommendations for the upcoming semester, and the OpenAI ChatGPT 4.0 service via the Langchain API. These issues were resolved through targeted bug fixes.

# Manual Testing Report

#### **Test Case ID: TC001, Name: User Login and Authentication**

* **New or Old:** New
* **Test items:** Okta authentication system
* **Test priority:** High
* **Dependencies:** None
* **Preconditions:** Okta setup completed, user registered in Okta
* **Input data:** User credentials
* **Test steps:**
  1. Launch the application.
  2. Navigate to the login page.
  3. Enter valid credentials.
  4. Submit.
* **Postconditions:** User should be logged in and redirected to the dashboard.
* **Expected output:** Successful login, user data fetched.
* **Actual output:** As expected.
* **Pass or Fail:** ??
* **Bug ID:** N/A

#### **Test Case ID: TC002, Name: Course Recommendation Chatbot**

* **New or Old:** New
* **Test items:** Chatbot recommendation system
* **Test priority:** High
* **Dependencies:** AI service, MongoDB
* **Preconditions:** AI service running, course data in MongoDB
* **Input data:** User input for course preferences
* **Test steps:**
  1. Start a chat with the bot.
  2. Enter course preferences (e.g., major, interests).
  3. Receive course suggestions.
* **Postconditions:** Personalized course recommendations displayed.
* **Expected output:** AI chatbot provides relevant course recommendations.
* **Actual output:** As expected.
* **Pass or Fail:** ??
* **Bug ID:** N/A

# Automated Testing Report

* + **Automated Testing Frameworks:**
    - **Frontend:** UI Automation with Selenium BDD framework.
    - **Backend:** Test NG, Rest Assured API .
    - **Python AI Service:** PyTest.
  + **Test Code Repository:**
    - Located in the /tests folder in the project repository. Separate test directories for frontend, backend, and Python service.
  + **Screenshots/Generated Reports:**
    - The generated test reports are stored in /reports/ and can be viewed in Jenkins CI after the test run.

# Testing Metrics

* + **Number of test cases:** .
  + **Test coverage:**
  + **Defect rate:**

# References

**BU MET CS Team 1.** (2024). *Project documentation* (SPPP, SPPP risk management, Progress Report, SDD, Readme.md). N. Liew, N. Foithong, A. Singh, B. Cevik, Y. Liu, P. Chantarapornrat (Authors).

**Okta.** (2023). *Authentication API documentation*. Retrieved from<https://developer.okta.com/docs/reference/api-overview/>

**Langchain.** (2023). *API documentation for OpenAI ChatGPT 4.0, chat history generation, RAG, and session management*. Retrieved from<https://docs.langchain.com/docs/>

**CS673 Course Team.** (2024). *Notes from CS673 slides*. Blackboard Course MS.

**Spring.io.** (2023). *Spring Boot documentation*. Retrieved from<https://spring.io/projects/spring-boot>

**Docker, Inc.** (2023). *Docker documentation*. Retrieved from<https://docs.docker.com/>

**Axios.** (2023). *Axios documentation*. Retrieved from<https://axios-http.com/docs/intro>

**Atlassian.** (2023). *JIRA documentation*. Retrieved from<https://support.atlassian.com/jira-software-cloud/docs/>

**GitHub.** (2023). *GitHub documentation*. Retrieved from<https://docs.github.com/en>

**PostgreSQL Global Development Group.** (2023). *PostgreSQL documentation*. Retrieved from<https://www.postgresql.org/docs/>

**Foithong, N.** (2024, September 18). *Chat AI Bot - 673ONE*. Figma. Retrieved from<https://www.figma.com/design/gjNG1bADwnFxgDqclMwQVQ/Chat-AI-Bot---673ONE?node-id=0-1&node-type=canvas>

**Braude, E., & Bernstein, M. E.** (2016). *Software engineering: Modern approaches* (2nd ed.). Waveland Press, Inc.

**Martin, R. C.** (2003). *Agile software development: Principles, patterns, and practices*.

**Bruegge, B., & Dutoit, A. H.** (2010). *Object-oriented software engineering: Using UML, patterns, and Java*.

**Pfleeger, S. L., & Atlee, J. M.** (2010). *Software engineering: Theory and practice*.

**Pressman, R. S.** (2014). *Software engineering: A practitioner’s approach* (9th ed.). McGraw-Hill.

**Van Vliet, H.** (2008). *Software engineering: Principles and practice*.

**Sommerville, I.** (2016). *Software engineering* (10th ed.).

**Sommerville, I.** (2011). *Engineering software products: An introduction to modern software engineering*.

**Farley, D.** (2022). *Modern software engineering: Doing what works to build better software faster*.

**Brooks, F. P., Jr.** (1995). *The mythical man month: Essays on software engineering* (2nd ed.). Addison-Wesley.

**Freeman, E., Freeman, E., Bates, B., & Sierra, K.** (2004). *Head first design patterns*. O'Reilly Media.

**Fowler, M., Beck, K., & Roberts, D.** (2019). *Refactoring: Improving the design of existing code* (2nd ed.). Addison-Wesley.

**McConnell, S.** (2004). *Code complete: A practical handbook of software construction* (2nd ed.). Microsoft Press.

**Martin, R. C.** (2008). *Clean code: A handbook of agile software craftsmanship*. Prentice Hall.

**Thomas, D., & Hunt, A.** (2019). *The pragmatic programmer: Your journey to mastery* (20th Anniversary ed.). Addison-Wesley.

**Winters, T., Manshreck, T., & Wright, H.** (2020). *Software engineering at Google: Lessons learned from programming over time*. O'Reilly Media.

**Humble, J., & Farley, D.** (2010). *Continuous delivery: Reliable software releases through build, test, and deployment automation*. Addison-Wesley.

**Kim, G., Behr, K., Spafford, G., & Ruen, C.** (2018). *The phoenix project: A novel about IT, DevOps, and helping your business win* (3rd ed.). IT Revolution Press.

**Forsgren, N., Humble, J., & Kim, G.** (2018). *Accelerate: The science of lean software and DevOps: Building and scaling high-performance organizations*. IT Revolution Press.

**Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N.** (2016). *The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations*. IT Revolution Press.

**Farley, D.** (2021). *Continuous delivery pipelines: How to build better software faster*. O'Reilly Media.

# Glossary of Terms

**Academic Advisor**: A faculty or staff member who guides students on academic courses, degree requirements, and career planning.

**AI (Artificial Intelligence)**: The simulation of human intelligence by machines, specifically algorithms that allow computers to perform tasks like understanding natural language and making decisions. This project integrates OpenAI’s ChatGPT 4.0 and Llama 2 for real-time course recommendations.

**API (Application Programming Interface)**: A set of protocols and tools that allow different software components to communicate and share data. In this project, APIs connect the front-end application to back-end services, enabling data exchange between the chatbot, databases, and AI models.

**Application**: A software program designed to perform a specific function for the user.

**Authentication**: The process of verifying the identity of a user to ensure secure access. Okta is used in the project for handling user authentication, ensuring only authorized users can log in.

**Axios**: A JavaScript library for making HTTP requests from Node.js or the browser.

**AWS (Amazon Web Services)**: A cloud computing platform offering services like storage, databases, and AI tools.

**Backend**: The part of the system responsible for connecting databases, tools, APIs, and services to frontend components.

**Branches**: Versions of a codebase used for managing features or changes before merging into the main code.

**Bugs**: Errors in code causing malfunction or failure in the system.

**Caching**: A mechanism to temporarily store data for quick access, reducing load times. This project considers using caching for faster page loads by retaining frequently accessed data.

**Chat**: A platform for real-time communication via text, voice, or video.

**Chatbot**: An AI-driven system designed to engage in conversations with users, providing information and assistance based on user queries.

**ChatGPT 4.0**: The AI model used in the web application.

**Components**: Reusable parts of an application, such as UI elements or modular code.

**Database**: An organized collection of data stored electronically.

**Docker**: A platform that uses containers to package and deploy applications, ensuring that software runs the same way regardless of the environment. Docker is an optional tool in this project for maintaining consistent development and production environments.

**Fetch API**: A JavaScript API for making network requests to servers.

**Figma**: A cloud-based design tool for interface design and prototyping.

**Framework**: A collection of tools and libraries to streamline software development.

**Frontend**: The user-facing side of the web application, built using React.js. It provides the interface through which users interact with the chatbot and access course recommendations.

**CI/CD (Continuous Integration/Continuous Deployment)**: A development practice where code changes are automatically tested and deployed, ensuring that new features or bug fixes are regularly integrated into the project. GitHub Actions is used for this purpose.

**Configuration/Config**: A set of parameters to customize a program's behavior.

**GitHub**: A file management system for collaborative software development.

**HTTPs**: Hypertext Transfer Protocol Secure, a secure version of HTTP.

**Issues**: Problems or tasks tracked in project management tools.

**Java**: A programming language used for building applications.

**JIRA**: A project management tool for tracking issues, bugs, and tasks.

**Langchain**: A company offering AI tooling for Retrieval-Augmented Generation (RAG) and session management.

**LLM (Large Language Model)**: A type of AI model trained on vast amounts of text data to understand and generate human-like text responses.

**Management**: Coordinating resources and tasks to achieve objectives.

**Microservices**: A software architecture where applications are structured as independent, deployable services.

**Okta**: A cloud-based identity and access management service.

**OpenAI**: A company that provides the ChatGPT AI model.

**PostgreSQL**: An open-source relational database management system used for storing structured data efficiently and chat history and other user data in this project.

**Python3**: A version of the Python programming language.

**RAG (Retrieval-Augmented Generation)**: A technique that combines retrieval of relevant information and generative responses to improve the accuracy of chatbot replies.

**React.js**: A JavaScript library used to build the front end of the application. It allows for the creation of interactive user interfaces and handles the chat interaction between the user and the AI.

**Redux**: A state management library for JavaScript applications, often used with React.

**RESTful API**: A set of guidelines for building APIs using standard HTTP methods.

**Regression Testing**: A software testing practice that ensures new code changes don’t adversely affect existing functionality. The project implements regression testing to ensure that updates to the AI chatbot don’t disrupt other features.

**REST Assured**: A Java library used for testing RESTful APIs. The project uses REST Assured to validate that the backend APIs respond correctly and efficiently.

**Selenium**: A testing framework used to automate web browsers, validating the chatbot’s user interface. Selenium ensures the front end functions properly after each code change.

**Spring Boot**: A Java-based framework used for building and deploying back-end services, handling API requests, and managing interactions with the databases.

**TensorFlow**: A machine learning framework used to develop and train AI models. TensorFlow may be used to enhance the chatbot’s learning capabilities.

**Unit Testing**: A type of software testing where individual components of the application are tested in isolation. This project uses unit testing to ensure each module (React components, Java services, and AI models) works correctly before integrating them.

**WebSockets**: A protocol enabling real-time, two-way communication between a client and server, allowing for instant updates during chat sessions.

**Workflow**: A sequence of steps to complete a task or achieve an objective.