

Project Title: System Verification and Validation Plan for Course Buddy

Team #5, Overwatch League

Jingyao, Qin

Qianni, Wang

Qiang, Gao

Chenwei, Song

Shuting, Shi

November 3, 2023

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

Contents

1	Symbols, Abbreviations, and Acronyms	iv
2	General Information	1
2.1	Summary	1
2.2	Objectives	1
2.3	Relevant Documentation	1
3	Plan	1
3.1	Verification and Validation Team	2
3.2	SRS Verification Plan	2
3.3	Design Verification Plan	6
3.4	Verification and Validation Plan Verification Plan	8
3.5	Implementation Verification Plan	9
3.6	Automated Testing and Verification Tools	9
3.7	Software Validation Plan	10
4	System Test Description	10
4.1	Tests for Functional Requirements	10
4.1.1	Area of Testing1	10
4.1.2	Area of Testing2	11
4.2	Tests for Nonfunctional Requirements	11
4.2.1	Area of Testing1	11
4.2.2	Area of Testing2	12
4.3	Traceability Between Test Cases and Requirements	12
5	Unit Test Description	12
5.1	Unit Testing Scope	12
5.2	Tests for Functional Requirements	12
5.2.1	Module 1	12
5.2.2	Module 2	13
5.3	Tests for Nonfunctional Requirements	13
5.3.1	Module ?	13
5.3.2	Module ?	13
5.4	Traceability Between Test Cases and Modules	14

6	Appendix	15
6.1	Symbolic Parameters	15
6.2	Usability Survey Questions?	15

List of Tables

1	Verification and Validation Team Members and Their Roles	2
---	--	---

List of Figures

1 Symbols, Abbreviations, and Acronyms

symbol	description
UI	User Interface
ML	Machine Learning
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol

This document outlines the Verification and Validation (V&V) plan for the Course Buddy project developed by Team #5, Overwatch League. The V&V plan is a critical component of our project management and quality assurance processes, ensuring that Course Buddy not only meets its specified requirements but also fulfills the needs and expectations of its users and stakeholders.

Roadmap The V&V plan is structured as follows:

1. **Symbols, Abbreviations, and Acronyms**
2. **General Information**
3. **Plan**
4. **System Test Description**
5. **Unit Test Description**

2 General Information

2.1 Summary

2.2 Objectives

2.3 Relevant Documentation

[Author](#) (2019)

3 Plan

This section outlines the comprehensive strategy for verifying and validating the Course Buddy software, ensuring its alignment with specified requirements and design standards. The plan spans from team roles in verification to the utilization of various testing and verification tools.

3.1 Verification and Validation Team

Name	Role and Specific Duties
Jinyao Qin	Lead Verifier: Oversees the entire process, coordinates with other team members, and ensures all verification steps are followed diligently.
Qianni Wang	Implementation Specialist: Reviews the codebase to ensure it aligns with the documented requirements, also verifies the code's functionality, performance, and security aspects.
Qiang Gao	Implementation Specialist: same as Qianni Wang
Chenwei Song	Manual Test Engineer: Responsible for manual test cases, ensuring that all tests run in different environments, and reporting the results in an understandable format for the team.
Shuting Shi	Test Automation Engineer: Responsible for automating test cases, ensuring that all tests run in different environments, and reporting the results in an understandable format for the team.

Table 1: Verification and Validation Team Members and Their Roles

3.2 SRS Verification Plan

For the verification of the Software Requirements Specification (SRS) document, the following approaches will be adopted:

1. **Peer Review:** The SRS will be reviewed by team members and classmates to identify any inconsistencies, ambiguities, or missing requirements.
2. **Expert Review:** Experts in software development will be consulted to ensure the requirements are complete and feasible.
3. **Client Feedback:** The document will be shared with the client or stakeholders for their feedback, ensuring alignment with their expectations and needs.

4. **Automated Analysis Tools:** Tools such as requirement management software will be used for tracing and managing requirements systematically.

Additionally, an SRS checklist will be utilized to systematically verify the content of the SRS document:

- **1. Purpose of the Project**
 - 1.1. User Business
 - 1.2. Goals of the Project
- **2. Stakeholders**
 - 2.1. Client
 - 2.2. Customer
 - 2.3. Other Stakeholders
 - 2.4. Hands-On Users of the Project
 - 2.5. User Participation
- **6. The Scope of the Work**
 - 6.1. The Current Situation
 - 6.2. The Context of the Work
 - 6.3. Specifying a Business Use Case (BUC)
- **7. Business Data Model and Data Dictionary**
 - 7.1. Business Data Model
 - 7.2. Data Dictionary
- **8. The Scope of the Product**
 - 8.1. Product Boundary
 - 8.2. Product Use Case Table
 - 8.3. Individual Product Use Cases (PUC's)
- **9. Functional Requirements**

- 9.1. Authentication
- 9.2. User Input
- 9.3. Data
- 9.4. Scheduling
- **10. Look and Feel Requirements**
 - 10.1. Appearance Requirements
 - 10.2. Style Requirements
- **11. Usability and Humanity Requirements**
 - 11.1. Ease of Use Requirements
 - 11.2. Personalization and Internationalization Requirements
 - 11.3. Learning Requirements
 - 11.4. Understandability and Politeness Requirements
 - 11.5. Accessibility Requirements
- **12. Performance Requirements**
 - 12.1. Speed and Latency Requirements
 - 12.2. Safety-Critical Requirements
 - 12.3. Precision or Accuracy Requirements
 - 12.4. Robustness or Fault-Tolerance Requirements
 - 12.5. Capacity Requirements
 - 12.6. Scalability or Extensibility Requirements
 - 12.7. Longevity Requirements
- **13. Operational and Environmental Requirements**
 - 13.1. Expected Physical Environment
 - 13.2. Requirements for Interfacing with Adjacent Systems
 - 13.3. Productization Requirements
 - 13.4. Release Requirements

- **14. Maintainability and Support Requirements**
 - 14.1. Maintenance Requirements
 - 14.2. Supportability Requirements
 - 14.3. Adaptability Requirements
- **15. Security Requirements**
 - 15.1. Access Requirements
 - 15.2. Integrity Requirements
 - 15.3. Privacy Requirements
 - 15.4. Audit Requirements
 - 15.5. Immunity Requirements
- **16. Cultural Requirements**
- **17. Compliance Requirements**
 - 17.1. Legal Requirements
 - 17.2. Standards Compliance Requirements
- **18. Open Issues**
- **19. Off-the-Shelf Solutions**
 - 19.1. Ready-Made Products
 - 19.2. Reusable Components
 - 19.3. Products That Can Be Copied
- **20. New Problems**
 - 20.1. Effects on the Current Environment
 - 20.2. Effects on the Installed Systems
 - 20.3. Potential User Problems
 - 20.4. Limitations in the Anticipated Implementation Environment
 - 20.5. Follow-Up Problems

- **21. Tasks**
 - 21.1. Project Planning
 - 21.2. Planning of the Development Phases
- **22. Migration to the New Product**
- **23. Costs**
- **24. User Documentation and Training**
 - 24.1. User Documentation Requirements
 - 24.2. Training Requirements
- **25. Waiting Room**
- **26. Ideas for Solutions**

3.3 Design Verification Plan

The design verification for our project will focus on ensuring that the design is user-friendly, intuitive, and aligns with the architectural requirements specified in the SRS. The verification plan will include the following key activities:

1. **Peer Reviews:** The design documents and models will be reviewed by team members and classmates to critique and provide feedback on the design's usability, intuitiveness, and adherence to architectural requirements.
2. **Design Walkthroughs:** Scheduled sessions where the design team presents the design to the stakeholders, including peers and supervisors, for feedback and suggestions.
3. **Prototype Testing:** Early versions of the design will be tested to gather quick feedback on the design's effectiveness and user experience.
4. **Consistency Check:** Ensuring that the design remains consistent with the requirements and objectives outlined in the SRS document.

To comprehensively verify the design, the following checklist will be used:

1. Design Documentation Review:

- Check if the design documentation is complete and clearly describes the architecture, components, and interfaces.
- Ensure that the design aligns with the project's objectives and requirements specified in the SRS.

2. User Interface (UI) and User Experience (UX) Evaluation:

- Verify that the UI design is intuitive and user-friendly.
- Ensure UI consistency across different parts of the application.
- Assess the UX for compliance with common usability standards and practices.

3. Architectural Conformity:

- Confirm that the system architecture supports all the required functionalities.
- Check for scalability, maintainability, and flexibility of the design.

4. Performance and Security Review:

- Ensure that the design incorporates adequate performance optimizations.
- Review the design for potential security vulnerabilities and data protection measures.

5. Compliance with Standards:

- Verify adherence to relevant industry and design standards.

6. Feedback Integration:

- Check that feedback from previous reviews (by classmates, peers, or stakeholders) has been adequately incorporated into the design.

3.4 Verification and Validation Plan Verification Plan

The verification and validation (V&V) plan for our project includes ensuring the integrity and effectiveness of the V&V processes themselves. Given the importance of this plan in the overall project quality assurance, the following approaches will be employed:

1. **Peer Review:** The V&V plan will be reviewed by team members and classmates to identify any omissions or areas needing improvement.
2. **Mutation Testing:** This technique will be applied to evaluate the ability of our test cases to detect faults deliberately injected into the code.
3. **Iterative Feedback Incorporation:** Feedback from all review sessions and testing phases will be systematically incorporated to refine the V&V plan.

To systematically verify the V&V plan, the following checklist will be used:

- Is the plan comprehensive, covering all aspects of software verification and validation?
- Are the responsibilities and roles in the V&V process clearly defined?
- Does the plan include a variety of testing methods (e.g., unit testing, integration testing, system testing)?
- Is there a clear process for incorporating feedback and continuous improvement in the V&V process?
- Are there criteria defined for the success of each testing phase?
- Is mutation testing included to assess the thoroughness of the test cases?
- Are there measures in place to track and resolve any identified issues during the V&V process?
- Does the plan align with the project's schedule, resources, and constraints?

3.5 Implementation Verification Plan

The Implementation Verification Plan will ensure that the software implementation adheres to the requirements and design specifications outlined in the SRS. Key components of this plan include:

- **Unit Testing:** A comprehensive suite of unit tests, as detailed in the project’s test plan, will validate individual components or modules of the software. PyTest, a flexible and powerful testing tool, will be used for writing and executing these tests.
- **Static Analysis:** Pylint and Flake8 will be employed for static code analysis to identify potential bugs, security vulnerabilities, and issues with code style and complexity.
- **Code Reviews and Walkthroughs:** Regularly scheduled code reviews and walkthroughs with team members and supervisors to inspect code quality, readability, and adherence to the Flask framework’s best practices and design patterns.
- **Continuous Integration:** Automated build and testing processes will be implemented using tools like GitHub Actions, to ensure continuous code quality, integration, and deployment.
- **Performance Testing:** The use of tools like Locust for load testing will help evaluate the application’s performance under various conditions, particularly focusing on how the Flask application handles concurrent requests and data processing.

3.6 Automated Testing and Verification Tools

For automated testing and verification in our Flask/Python project, the following tools will be employed:

- **Unit Testing Framework:** PyTest will be used for developing and running unit tests.
- **Profiling and Performance Tools:** Tools like cProfile for Python will assist in identifying performance bottlenecks and optimizing code efficiency.

- **Static Code Analyzers:** Pylint and Flake8 will be used to analyze Python code quality, adherence to coding standards, and identification of potential errors.
- **Continuous Integration:** GitHub Actions will automate the build, testing, and deployment process, ensuring continuous integration and delivery of the Python codebase.
- **Linters:** Flake8 will be used to enforce coding standards.

3.7 Software Validation Plan

The Software Validation Plan will focus on ensuring that the final product meets the requirements and expectations of the stakeholders. Key strategies include:

- **Beta Testing:** Involvement of selected users in the beta testing phase to provide real-world feedback on the software's functionality and usability.
- **Stakeholder Review Sessions:** Regular review meetings with stakeholders to confirm that the software meets the intended requirements and use cases.
- **Demo to Supervisor:** A demonstration of the software to the project supervisor following the Rev 0 demo for feedback and validation.
- **Reference to SRS Verification:** Aligning the validation activities with the SRS verification efforts to ensure consistency in meeting the documented requirements.

4 System Test Description

4.1 Tests for Functional Requirements

4.1.1 Area of Testing1

Title for Test

1. test-id1

Control: Manual versus Automatic

Initial State:

Input:

Output:

Test Case Derivation:

How test will be performed:

2. test-id2

Control: Manual versus Automatic

Initial State:

Input:

Output:

Test Case Derivation:

How test will be performed:

4.1.2 Area of Testing2

...

4.2 Tests for Nonfunctional Requirements

4.2.1 Area of Testing1

Title for Test

1. test-id1

Type: Functional, Dynamic, Manual, Static etc.

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

Type: Functional, Dynamic, Manual, Static etc.

Initial State:

Input:

Output:

How test will be performed:

4.2.2 Area of Testing2

...

4.3 Traceability Between Test Cases and Requirements

5 Unit Test Description

5.1 Unit Testing Scope

5.2 Tests for Functional Requirements

5.2.1 Module 1

1. test-id1

Type:

Initial State:

Input:

Output:

Test Case Derivation:

How test will be performed:

2. test-id2

Type:

Initial State:

Input:

Output:

Test Case Derivation:

How test will be performed:

3. ...

5.2.2 Module 2

...

5.3 Tests for Nonfunctional Requirements

5.3.1 Module ?

1. test-id1

Type:

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

Type: Functional, Dynamic, Manual, Static etc.

Initial State:

Input:

Output:

How test will be performed:

5.3.2 Module ?

...

5.4 Traceability Between Test Cases and Modules

References

Author Author. System requirements specification. <https://github.com/...>, 2019.

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

6.2 Usability Survey Questions?

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage etc. You should look to identify at least one item for each team member.
2. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?