# 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

## List of Tables

## List of Figures

## 1 Symbols, Abbreviations, and Acronyms

| symbol | description |
|---|---|
| UI | User Interface |
| ML | Machine Learning |
| API | Application Programming Interface |
| HTTP | Hypertext Transfer Protocol |
| PDF | Portable Document Format |
| .csv | Comma-Separated Values |
| .txt | Text file |

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 it aligns 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. **Supervisor Review:** The SRS will be reviewed by our supervisor, who can provide valuable insights from a strategic and technical perspective.

4. **Client Feedback:** The document will be shared with the client or stakeholders for their feedback, ensuring alignment with their expectations and needs.

5. **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: Please see our SRS.pdf for more details.

## 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. **Supervisor Review:** The design will be presented to the project supervisor for a thorough review, focusing on adherence to technical specifications and project objectives.

3. **Design Walkthroughs:** Scheduled sessions where the design team presents the design to the stakeholders, including peers and supervisors, for feedback and suggestions.

4. **Prototype Testing:** Early versions of the design will be tested to gather quick feedback on the design's effectiveness and user experience.

5. **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. /textitPyTest, a flexible and powerful testing tool, will be used for writing and executing these tests.
- **Static Analysis:** /textitPylint and /textitFlake8 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:** /textitPyTest will be used for developing and running unit tests.
- **Profiling and Performance Tools:** Tools like /textitcProfile for Python will assist in identifying performance bottlenecks and optimizing code efficiency.
- **Static Code Analyzers:** /textitPylint and /textitFlake8 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:** /textitFlake8 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 Authentication

1. TFR1-A1

   Control: Functional, Manual, Dynamic

   Initial State: Sign-up page that is used to create an account with username and password

Input: Valid username and password in string format (letters, numbers, etc)

Output: A prompt saying that the account has been created successfully

Test Case Derivation: If the username and password are provided in a valid format, the user should be able to create an account and should be notified that the account has been created

How test will be performed: A list of string pairs with different combinations of letters, and numbers will be given to the function that handles creating an account and we can then check our database if the provided string pairs get saved and if the website is prompting the notification telling the account has been created.

2. TFR1-A2

   Control: Functional, Manual, Dynamic

   Initial State: Sign-up page that is used to create an account with username and password

   Input: Invalid username or password (such as empty strings, too few characters, existing username etc.)

   Output: A prompt saying that the username or password is invalid with detailed information such as more characters are needed or username/password cannot be empty or the username already exists

   Test Case Derivation: If the username and password are provided in an invalid format, the user should be notified with more information

   How test will be performed: Invalid usernames and passwords such as empty strings or strings with only a few letters or duplicated usernames will be given to the function that creates an account and we will check if the function throws an exception and if the website is notifying the user to provide valid inputs instead.

3. TFR2-A3

   Control: Functional, Manual, Dynamic

Initial State: Sign-in page that is used to sign in with an existing account that includes placeholders of username and password

Input: Username and password in string format

Output: If the username and password match user data stored in our database, the user should be able to log in and be redirected to the home page, otherwise the user should be prompted saying that the username and password provided do not match

Test Case Derivation: If the username and password match, this means our user is authenticated and hence should be able to log in, authenticated users should be redirected to the home page once logged in for further actions. If the username and password do not match, the user should be prompted in some way to be notified and asked to try again

How test will be performed: A list of usernames and passwords (some of them match mock data stored in our database and some of them are just random strings that are not in our database) should be given to the function that handles user login, and we will check for those that match the mock user data in our database if the website directs to the home page and for those that do not match if the website prompts saying that username and password do not match

4. TFR3-A4

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in, and a drop-down menu is clicked and expanded

   Input: The option *log out* in the drop-down menu is selected and clicked

   Output: The user should be able to log out of his/her account and prompted saying that you have logged out successfully

   Test Case Derivation: If the option *log out* is selected and clicked, the user should be able to log out of his/her account and the user should be notified in some way so that he/she knows that the account has been logged out successfully

   How test will be performed: Manually log in using a mock account using different browsers and then log out for each one of them and see if we can log out and receive notifications about logging out

5. TFR4-A5

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and in the home page

   Input: The tab to view the scheduling information on the home page is selected and clicked

   Output: The scheduling information (in list view, Kanban view or calendar view, by default it is in list view) is displayed

   Test Case Derivation: The user should be able to view his/her scheduling information in a reasonable format such as list view, Kanban view or calendar view if the option to view the scheduling information is clicked on the home page

   How test will be performed: Manually log in using a list of mock accounts (the difference between these accounts would be the customized way of viewing the scheduling information such as by default using list view, using Kanban view, or using calendar view) using different browsers and then click the tab that is responsible for redirecting to the page viewing the scheduling information on the home page and check if we can be redirected to the page listing scheduling information and check if the information is displayed in the view that the account has been set

6. TFR5-A6

   Control: Functional, Manual, Dynamic

   Initial State: The user is on the page where he can retrieve his password by answering a list of security questions

   Input: Strings to each one of the security questions that match the user data stored in our database

   Output: The password gets changed successfully and updated in our database, a text or diagram should be displayed to notify the user that the password has been changed

   Test Case Derivation: The user should be able to change his password by answering the security questions correctly and should be notified in some way that the password has been changed

How test will be performed: Manually log in using a mock account, navigate to the changing password page, answer mock security questions correctly and see if we get a notification like a text or a diagram indicating that the password has been changed successfully

7. TFR5-A7

   Control: Functional, Manual, Dynamic

   Initial State: The user is on the page where he can retrieve his password by answering a list of security questions

   Input: Strings to each one of the security questions that do not match the user data stored in our database

   Output: A text or diagram should be displayed to notify the user that the answer to the security questions are not correct

   Test Case Derivation: The user should not be able to change his password by answering the security questions wrong and should be notified in some way that the answers are not correct

   How test will be performed: Manually log in using a mock account, navigate to the changing password page, answer mock security questions wrong and see if we get a notification like a text or a diagram indicating that the answers are wrong

#### 4.1.2 User Input

8. TFR6-UI1

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in, and is currently on the course page,

   Input: The option uploading PDF is selected and clicked and a PDF file is uploaded

   Output: A prompt saying that the PDF has been uploaded successfully or saying that there is an error in cases where the format does not match (other files such as txt or csv etc.)

Test Case Derivation: The user should be notified if the PDF gets uploaded successfully or when there is an error

How test will be performed: Manually log in and navigate to the course page, click on the option to upload a PDF file to see if a prompt shows up saying that it gets uploaded successfully, also upload a different file with different formats such as txt and csv and see if there is a prompt saying that the format does not match

9. TFR6-UI2

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in, and is currently on the course page,

   Input: The option uploading PDF is selected and clicked and a PDF file is uploaded, repeat multiple times to upload multiple PDFs

   Output: A prompt saying that the PDF has been uploaded successfully or saying that there is an error in cases where the format does not match (other files such as txt or csv etc.)for each upload

   Test Case Derivation: The user should be notified if the PDF gets uploaded successfully or when there is an error for each upload

   How test will be performed: Manually log in and navigate to the course page, click on the option to upload a PDF file to see if a prompt shows up saying that it gets uploaded successfully. Repeat this multiple times to upload multiple PDFs and see if getting notifications multiple times

10. TFR7-UI3

    Control: Functional, Manual, Dynamic, Automatic

    Initial State: The user has logged in for the first time, a prompt shows up on the home page to ask if the user wants to select his/her preferred study intervals now or do it later, and a timetable is displayed where multiple study intervals are indicated

    Input: The user clicks on the option saying do it now or the user clicks on the option saying do it later

Output: The user gets directed to the page where he/she can change the study intervals on a timetable if clicks on do it now or the prompt will be closed and the user stays on the home page

Test Case Derivation: The user should be able to be redirected to the page where he/she can select the preferred study intervals if he/she wants to do it now for the first time, otherwise the prompt should disappear and the user should be able to stay on the home page and do further other actions

How test will be performed: Manually log in to a new mock account and see if there is a prompt asking if we want to select the preferred study intervals now or do it later then click on do it now and see if we get directed to the page where we can select the intervals on a timetable. Repeat logging in for a different new account again but click on do it later this time to see the prompt goes away and if we stay on the home page

11. TFR8-UI4

    Control: Functional, Manual, Dynamic, Automatic

    Initial State: The user has logged in, and a timetable is displayed where multiple study intervals are indicated

    Input: Drag and select the preferred time interval

    Output: The preferred time interval is highlighted and a save button is clicked

    Test Case Derivation: The selected time interval should be highlighted to indicate that this has been selected and once it is selected it should be saved after the save button is clicked

    How test will be performed: Manually log in and navigate to the setting to change the time interval, we will check if a timetable is displayed and different time intervals are indicated in some ways, then we will select multiple different time intervals and check if selected intervals are highlighted and if the selected intervals are saved to the database once the save button is clicked

12. TFR9-UI5

Control: Functional, Manual, Dynamic, Automatic

Initial State: The user has logged in for the first time, a prompt shows up on the home page to ask if the user wants to select his/her preferred study and break intervals now or do it later, and a Pomodoro is displayed where these can be input

Input: The user clicks on the option saying do it now or the user clicks on the option saying do it later

Output: The user gets directed to the page where he/she can change the study and break intervals on a Pomodoro if he clicks on do it now, or the prompt will be closed and the user stays on the home page if he clicks to do it later

Test Case Derivation: The user should be able to be redirected to the page where he/she can select the preferred study and break intervals if he/she wants to do it now for the first time, otherwise the prompt should disappear and the user should be able to stay on the home page and do further other actions

How test will be performed: Manually log in to a new mock account and see if there is a prompt asking if we want to select the preferred study and break intervals now or do it later then click on do it now and see if we get directed to the page where we can select the intervals on the Pomodoro. Repeat logging in for a different new account again but click on do it later this time to see if the prompt goes away and if we stay on the home page

13. TFR10-UI6

    Control: Functional, Manual, Dynamic, Automatic

    Initial State: The user has logged in, and is on the page showing the Pomodoro settings

    Input: The user selects how much time they want to spend studying versus how much time taking a break

    Output: After the save button is clicked the Pomodoro settings will now reflect what the user had previously input for study and break times

Test Case Derivation: The intervals for studying and taking a break should be shown on the timer once it has been input and save has been clicked

How test will be performed: Manually log in and navigate to the setting to change the pomodoro study and break intervals, we will check if the Pomodoro is displayed and different time intervals are indicated in some ways, and then we will input multiple different intervals and check if the timer reflects these intervals and are saved to the database once the save button is clicked

14. TFR11-UI7

    Control: Functional, Manual, Dynamic

    Initial State: The user has logged in and the friend list panel is clicked and expanded

    Input: The option to send a friend request to a specific user (by username) is selected and clicked

    Output: A prompt shows up and asks the user to provide the username that he/she wants to send a friend request to

    Test Case Derivation: Once the option of sending a friend request is selected, a prompt should be provided to ask the user to input the username that he/she wants to send a friend request to

    How test will be performed: Manually log in, expand the friend list, click on sending friend request option, and then check if a prompt asking the user to provide the username that he/she wants to send a friend request to shows up

15. TFR11-UI8

    Control: Functional, Manual, Dynamic

    Initial State: A prompt is displayed asking to provide the username that he/she wants to send a friend request to

    Input: Valid and existing username and a send button is left-clicked

    Output: A text or diagram shows up to indicate that the friend request has been sent successfully

Test Case Derivation: When the user provides a valid username, the friend request should be sent successfully and there has to be a way to tell user that this has been done with no errors

How test will be performed: Manually log in, expand the friend list, click on the sending friend request option, and then input a list of valid and existing usernames and click on the send button to see if we get notifications (by text or diagram) saying that the friend request has been sent

16. TFR11-UI9

    Control: Functional, Manual, Dynamic

    Initial State: A prompt is displayed asking to provide the username that he/she wants to send a friend request to

    Input: Invalid username such as empty string and a send button is left-clicked

    Output: A text or diagram shows up to indicate that the username provided is invalid

    Test Case Derivation: When the user provides an invalid username, the friend request should not be sent and there has to be a way to tell the user that the username provided is invalid

    How test will be performed: Manually log in, expand the friend list, click on the sending friend request option, and then input a list of invalid usernames such as empty strings and click on the send button to see if we get a notification (by text or diagram) saying that the username provided is invalid

17. TFR12-UI10

    Control: Functional, Manual, Dynamic

    Initial State: A notification indicating that there is an incoming friend request (could use sound effects or highlight the notification icon)

    Input: The notification gets left-clicked and the button accepting the friend request is clicked

Output: A text or diagram showing that the friend request has been accepted

Test Case Derivation: When the user accepts an incoming friend request, he/she should be notified when the friend request gets accepted successfully

How test will be performed: Manually log in to one of the mock accounts, send a friend request to another mock account, log in to the account receiving the friend request and then check if there is a notification indicating that there is an incoming friend request, accept the friend request and then see if a text or a diagram gets displayed indicating that the friend request has been accepted

18. TFR13-UI11

    Control: Functional, Manual, Dynamic

    Initial State: A notification indicating that there is an incoming friend request (could use sound effects or highlight the notification icon)

    Input: The notification gets left-clicked and the button rejecting the friend request is clicked

    Output: A text or diagram showing that the friend request has been rejected

    Test Case Derivation: When the user rejects an incoming friend request, he/she should be notified that it has been rejected when the friend request gets rejected

    How test will be performed: Manually log in to one of the mock accounts, send a friend request to another mock account, log in to the account receiving the friend request and then check if there is a notification indicating that there is an incoming friend request, reject the friend request and then see if a text or a diagram gets displayed indicating that the friend request has been rejected

19. TFR14-UI12

    Control: Functional, Manual, Dynamic

Initial State: The user is logged in and is on the page that displays their tasks and current progress on those tasks

Input: The user can input for each task what their current progress status and then save it

Output: The task list will be updated with the current progress of the task

Test Case Derivation: The user should be able to see a list of the current tasks he has to do and how far along he is on them. He should also be able to manually change the progress for each task when he has made progress or feels he needs more time

How test will be performed: Manually log in and navigate to the task list page, select a task to edit the progress on, and then input a random percentage ranging from 0 to 100, Then, after pressing the save button this change should be committed and reflected on the task list page upon revisiting

20. TFR15-UI13

    Control: Functional, Manual, Dynamic

    Initial State: The user is logged in and is on the page that displays their tasks and current progress on those tasks with a subtask marked as complete and there is a pop-up asking if the pace is comfortable

    Input: The user clicks if the pace is comfortable or not on the given pop-up

    Output: On clicking that the pace is comfortable, nothing is changed, if the pace is not comfortable the time for each task will change accordingly

    Test Case Derivation: The system curates a study plan for the user, so if the user does not feel as if he is doing well with this plan he should be able to provide feedback and have the study plan change around this feedback

    How test will be performed: Manually log in head to the task list and complete a task. On the pop-up showing asking if the pace is comfortable, select that it is comfortable. Then, check that no change has been made to the timeline. Repeat this, but this time select that the pace is not comfortable, and then check that the study plan has changed how much time should be spent on the tasks

#### 4.1.3 Data

1. TFR16-D1
   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and is on the PDF extraction page

   Input: The user clicks to upload a PDF of his course outline, selects the PDF he wishes to upload, and clicks upload

   Output: The page shows the PDF has been uploaded successfully and is ready for extraction

   Test Case Derivation: The user should be able to upload his course schedule and have the website display which PDF has been uploaded and if it has been successful and is ready for parsing

   How test will be performed: Manually log in and then navigate to the PDF extraction page. Click on the button to select a PDF to upload and then select the PDF that has the course outline. Check that the task list displayed by the website shows the correct information that can be found on the PDF

2. TFR17-D2

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and is on the PDF extraction page and has uploaded the course outline PDF

   Input: The user presses to extract the course information from the uploaded PDF

   Output: The page shows lists the course information extracted from the PDF as a task list

   Test Case Derivation: The user should be able to upload his course schedule and have the website extract the necessary information from it in order to create his schedule. After it has extracted the information it should display it to the user for quick verification

How test will be performed: Manually log in and upload a course PDF. Continue and click to parse the course outline and create a task list. Check that the extracted information from the list including *courseName*, *taskType*, *taskName*, weight, and deadline from the uploaded course outline is correctly displayed as a task list

## 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?