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
Nov 3, 2023	1.0	Initial Draft

Contents

List of Tables

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
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**

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

Course Buddy is a scheduling software that generates personalized study plans for students from course outlines that would be incorporated into the student's own schedule. The application also comes with social components that enable friend-collaborative study sessions.

2.2 Objectives

This VnV plan would direct us through the testing process to ensure that our system would accomplish all the intended functions correctly and efficiently. Adequate tests would be planned according to functional and non-functional requirements stated in our SRS to ensure our project exhibits adequate qualities and functions. We would prioritize and polish core functions including PDF unloading and analysis, AI scheduling algorithms, compatibility with other calendar applications, and those associated non-functional requirements. If time allows, we would also aim to develop tests for AI attention detection and the social component.

2.3 Relevant Documentation

2.3.1 Development plan

As part of the development process, the writing of this VnV plan implementation of all the test cases shall follow the development process specified in the development plan.

2.3.2 SRS

The functional and non-functional requirements and their corresponding fit criteria would lead the test plan and be used as a guideline to make sure tests could cover all the documented requirements.

2.3.3 Hazard Analysis

Additional requirements related to software failures should also be covered in testing.

2.3.4 VnV Report

The result from executing this VnV plan would be recorded in the VnV report. The VnV plan could be revised from the results.

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

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 2: 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.

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.

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?

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.

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.

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. TFR16-17-D1

   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

2. TFR18-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. The user has uploaded the corresponding `PDF` file of the course syllabus on `AWS S3`.

Output: The function will show the user the calculated priority result on the extraction page and the course detail page as priority labels beside the course tasks.

Test Case Derivation: After all the priorities of course tasks have been calculated, the user should be able to preview and verify the calculated priorities on the extraction page related to the course information generated from the `PDF`. If the user agrees with the results, the user should be able to see the calculated priority result on his/her course detail page after clicking on the Add button.

How the 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 the priority label beside each task, and compare the labels according to the parameters of each task, the information extracted from the course syllabus file: *taskType*, *weight*, *deadline*. If the priority labels are reasonable according to the parameters, click the Add button to save the generated priorities. The tester then manually goes to the course detail page to check if the labels are displayed correctly.

3. TFR19-D3
   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in, set up a Pomodoro Timmer on the website, and turned on the camera on his/her device.

   Input: Live video stream captured by the camera, a series of consecutive image frames with information about faces, eyes, and movements.

   Output: After the end of the timer, the program will show the user the percentage score and a line graph chart showing the trend of the user's attention level during his/her focusing time.

   Test Case Derivation: The user should be able to detect human face and eye movements with different resolution cameras, in any lighting conditions, with different degrees of facial blocking, and at different angles, and receive reasonable analysis results after the Pomodoro Timmer ends.

How the test will be performed: Manually comparing single video frames of different resolutions to test the program's ability to localize the face and eyes of a person in different angles and lighting environments. Test the program's ability to recognize abnormality data by rapidly shaking and turning the head. Test the program's ability to recognize and classify facial expressions with different facial expressions. Partially cover the face to test whether the analysis results through eye tracking are reasonable. Test the system's analyzing ability and reaction time using real-time video data. Finding different testers to simulate different concentration situations in different environments to test the accuracy of the analysis results.

4. TFR20-D4

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in, set up a Pomodoro Timmer on the website, and turned on the camera on his/her device. The program detected the user's real-time attention score was low for a period of time.

   Input: Live video stream captured by the camera, a series of consecutive image frames with information about faces, eyes, and movements.

   Output: While the Pomodoro Timmer is on, it will be paused and inform the user via a pop-up window that his/her current attention span is low and suggest a break time.

   Test Case Derivation: The user should be able to see a notification to suggest a short break when their attention level is low and not suitable to continue the study task.

   How the test will be performed: Verify that the system gives correct feedback by looking around, and making different gestures and facial expressions to simulate different levels of attention. Verify that the system correctly and accurately triggers the reminder and pauses the timer after a specific period of inattention by performing different inattentiveness for different lengths of time.

5. TFR21-D5

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and generated or created a task list, used Pomodoro Timmers to record work time, or used a checklist view or calendar

view on the website to record their task progress each time after completing the scheduled study time period.

Input: The start and end times of the working hours for certain tasks recorded by the using the Pomodoro Timmers, the estimated progress user input at the end of the Pomodoro Timmers, as well as the user's list of tasks and the percentage of progress of the marked tasks.

Output: Displays to the user a list of tasks with completed or uncompleted status labels, and each uncompleted task has a progress bar that visually shows the progress of the current task.

Test Case Derivation: The user should be able to view their task list on the website, including the progress status of each task. The progress status of each task can be shown as marked as completed or incomplete, and incomplete tasks will have a progress bar showing the progress of completion.

How test will be performed: Create test cases, use the Pomodoro Timmer tool to keep track of work time, and use the TO-DO List view and Calendar view to mark and change the progress of tasks, and verify if the function can generate the correct view. Test cases should cover different scenarios such as task completion, task incompletion, and change in task plan. During the development phase, conduct unit and integration tests to ensure that the components of the task management tool work together correctly to support the task progress viewing functionality.

6. TFR22-D6

   Control: Functional, Manual, Dynamic

   Initial State: The user has the credentials for the corresponding calendar application he/she is trying to import, and the user has logged in to the website, is located on the Calendar page of the website, and has clicked the Import button.

   Input: The valid authentication information the user provides, such as a username and password or an authorization token, so that the system can access data from other calendar applications. The source of the event and schedule data used for import may be Calendar, Outlook, Google Calendar, and so on. This data includes information such as the date, time, and location of the event.

Output: A new combined calendar schedule of the internal and external calendar data will be displayed on the calendar page of the website without any conflict.

Test Case Derivation: Once the user has successfully authenticated and selected the calendar data to be imported, the system should be able to import this data into the user's web system so that the user can view and manage it in the calendar view of the website. The system should detect if there is a conflict between the user's internal existing events and the newly imported events. The system informs the user if there is any conflict, and provides the user with a change plan to resolve the conflict such as replacing, merging, or deleting.

How test will be performed: Test the authentication process manually against different external data sources to ensure that authentication can be performed successfully and relevant external data can be obtained. Use non-conflicting external calendar programs for import and merge to test whether the data is successfully combined and displayed correctly. Also use conflicting external calendar data for import testing to check whether the delete, merge, and replace functions are effective.

7. TFR23-D7

Control: Functional, Manual, Dynamic

Initial State: The user has the credentials for the corresponding calendar application he/she is trying to export to, and the user has logged in to the website, is located on the Calendar page of the website, and has clicked the Export button.

Input: Valid authentication information, such as a username and password or other authentication credentials. User's own task and schedule data, including *taskType*, *taskName*, *weight*, *deadline*, *startTime*, *endTime* etc. Also, the target calendar applications to which the data is exported, such as Calendar, Outlook, or Google Calendar.

Output: Tasks and events within the site can be seen on the corresponding external site with the relevant information for each of them. A success confirmation message or a failure error message at the end of the export process.

Test Case Derivation: After the user clicks EXPORT, selects the target calendar application, and completes the relevant authentication, the system should export the user's event and schedule data to the selected target calendar application, ensuring that the data is rendered correctly in other applications.

The user should be able to see the feedback to indicate whether the export operation was successful or not, if an error occurred, the user will see the appropriate error message. If the export was successful, the user should be able to get confirmation that the export operation was successful. If the user already has existing events and schedules in the target calendar application, the system should be able to recognize and inform the user and provide the user with decision options to resolve conflicts with the exported data.

How test will be performed: Manually create different types of events and calendars, then export them to different calendar applications, Calendar, Outlook, and Google Calendar, and verify the accuracy, completeness, and consistency of the data. Test whether the system can correctly recognize conflicts and provide users with decision-making options and successful conflict resolution by exporting calendar data with and without event conflicts.

8. TFR22-23-D8

   Control: Functional, Manual, Dynamic

   Initial State: The user does not have the credentials for the corresponding calendar application he/she is trying to import or export to, and the user has logged in to the website, is located on the Calendar page of the website, and has clicked the Import or Export button.

   Input: Invalid authentication information, such as a username and password or other authentication credentials. The target calendar applications to which the data is exported, such as Calendar, Outlook, or Google Calendar.

   Output: The warning message popped up to notify the user the account authorization inputs are not valid.

   Test Case Derivation: Once a user's authorization is entered incorrectly, they should be able to be notified and given the opportunity to re-enter their credential information.

   How test will be performed: Tested with incorrect authentication information. Manually try to import and export calendar data by using invalid credentials, non-existent usernames, and invalid passwords.

9. TFR24-D9

   Control: Functional, Manual, Dynamic

Initial State: The user has logged in and is on the TO-DO list view on the website.

Input: The user clicks the Download button to issue a command requesting export to PDF. The user selects the time range of the TO-DO list view and selects the period format of the export list which can be daily, or weekly. The information data in the task list includes *course*, *taskName*, *date*, *deadline*, *startTime*, *endTime*, *weight*, etc.

Output: PDF document with a to-do list view of the study plan in the selected time range and period format.

Test Case Derivation: The user should be able to see a preview of the exported TO-DO list study plan before the PDF file is downloaded. After the user presses the download button on the preview page, the PDF file with the task list data should be downloaded to his/her local device. The preview and the PDF file should have proper formatting, fonts, and layout, and the exported task information should be accurate and consistent.

How the test will be performed: By manually simulating user input, perform the operation of exporting the TO-DO list as a PDF, and check whether the generated PDF file contains the correct information about the tasks, expected formatting, fonts, layout, etc. Manually test using different study plans, including different numbers of tasks, different dates, and times of scheduling, etc., to ensure that the export function works properly in a variety of situations.

10. TFR24-D10

Control: Functional, Manual, Dynamic

Initial State: The user has logged in and is on the calendar view on the website.

Input: The user clicks the Download button to issue a command requesting export to PDF. The user selects the time range of the calendar list view and selects the period format of the calendar data which can be weekly or monthly. The calendar data includes *course*, *taskName*, *startDate*, *endDate deadline*, *startTime*, *endTime*, *weight*, etc.

Output: PDF document with a calendar view of the study plan in the selected time range and period format.

Test Case Derivation: The user should be able to see a preview of the exported study plan calendar before the PDF file is downloaded. After the user presses the download button on the preview page, the PDF file with the calendar data

should be downloaded to his/her local device. The preview and the PDF file should have proper formatting, fonts, and layout, and the exported event and task information should be accurate and consistent.

How the test will be performed: By manually simulating user input, perform the operation of exporting the calendar view as a PDF, and check whether the generated PDF file contains the correct information about the tasks and events, expected formatting, fonts, layout, etc. Manually test using different study plans, including different numbers of tasks, different dates, and times of scheduling, etc., to ensure that the export function works properly in a variety of situations.

4.1.4 Scheduling

1. TFR25-S1

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and is on the task creation page or the extraction page.

   Input: the task information such as *taskName*, *taskType*, *weight*, and *priority*. The user's input for the initial estimated time needed to finish the task.

   Output: the initial estimate of the total time required for the task to complete.

   Test Case Derivation: The user should be able to view an initial estimated completion time for any task generated from the course syllabus PDF documents they upload. This estimated completion time can be modified by the user at any time after the task has been created. For any tasks created manually by users, the user can assign an initial estimated completion time period and change it at any time after the task has been created.

   How the test will be performed: Generate tasks from course syllabus PDF files and create tasks directly. Testing should encompass various types of tasks with different weights, user actions, different learning time records, and progress percentage inputs. Manually try to modify the value after the tasks have been created, and try input the various values like negative numbers or text, etc.

2. TFR26-S2

   Control: Functional, Manual, Dynamic

Initial State: The user has logged in and generated or created a task list, used Pomodoro Timmers to record work time, or used a checklist or calendar view on the website to record their task progress each time after completing the scheduled study time period.

Input: Current schedule data for the task includes *deadline*, *priority*, *weight*, current progress in completing the task, the user's history of the percentage of progress completed, and the corresponding time spent on the task.

Output: The updated Learning Plan, regenerated task schedule information based on the updated task progress from the user, which may include reordering of task list and TO-DO list, new priority of the task, etc.

Test Case Derivation: Each time the user updates the task progress and records the corresponding time spent, they will receive a reminder to update the learning schedule. After that, the user should be able to view a newly generated learning plan associated with the rate of change in task progress.

How test will be performed: After simulating the user's change of task progress, check whether the system has kept all the original task information and updated the related task plan. Compare the pre- and post-update learning plans to ensure that the updated plan correctly reflects the changes in task progress.

3. TFR27-S3

Control: Functional, Manual, Dynamic

Initial State: The user has logged in and generated or created a task list, used Pomodoro Timmers to record work time, or used a checklist view or calendar view on the website to record their task progress each time after completing the scheduled study time period.

Input: *taskName*, *taskType*, *weight* and the initial estimate of the total time required for the task to complete, the percentage of completing progress of the task recorded by the user at the end of the Pomodoro Timmers, and the corresponding time spent.

Output: The time needed to finish each task is dynamically updated accordingly.

Test Case Derivation: The system should be capable of calculating the most up-to-date estimated time required to finish each task after the user records and inputs the progress percentage and the corresponding time spent.

How the test will be performed: Generate tasks from course syllabus PDF files and create tasks directly. Input multiple progress updates and corresponding study hours for each task. Estimate task completion time through both manual and automated testing methods. Testing should encompass various types of tasks with different weights, user actions, different learning time records, and progress percentage inputs. Compare the pre- and post-input of the new task progress and corresponding hours spent and check whether the changes in task estimated time needed are accurately reflected in the task data.

4. TFR28-S4

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and uploaded a course syllabus PDF file, and is on the extraction page of the website.

   Input: The user's task list and data for tasks including *taskType*, *weight* and *deadline*.

   Output: A task priority list, based on sorting the tasks in order by task type, weight, and deadline. High-priority tasks will be ranked first.

   When the user attempts to extract all the information from the course, the program will extract the necessary data from the uploaded course syllabus PDF file and generate a list of tasks with a calculated priority number. The system should categorize all tasks into three priority levels: high, medium, and low, and assign each task a corresponding priority label. After this process, the user should be able to view the task lists with priority labels, ordered in descending order based on the priorities generated by the system.

   How test will be performed: Create test sets with task lists of different task types, weights, and deadlines, and check that the generated list of sorted tasks meets the expected priority ordering. Using special case inputs, such as an empty task list, to ensure that the system handles edge cases correctly.

5. TFR29-S5

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and generated or created a task list with task information.

Input: Task information: *taskName*, *taskType*, *weight*, *deadline priority*. The learner's preferred day and time period for the study. The user's current schedule, including scheduled events, and other existing tasks. The working time of the Pomodoro Timmer setting for each learning task.

Output: A complete schedule of the learning program, including the name, weight, deadline, priority, sub-tasks under each task, scheduled start and end times of Pomodoro Timmer for each sub-tasks. Display task and sub-task time periods and information on other external platforms.

Test Case Derivation: Select different study periods, days, and various Pomodoro interval settings to assess the system's ability to handle invalid or incomplete input data. Using valid data, confirm that the system can generate a comprehensive study plan for each task, considering preferred study times and schedules while avoiding conflicts with existing activities. Export study schedules to an external calendar program and test the system's capability to accurately and completely export study plans to other calendar applications. In cases of task scheduling conflicts, verify that the system can provide solutions, such as suggesting alternative study times or allocating sub-tasks to different days.

How test will be performed: Tests can be performed using automated testing frameworks to simulate various scenarios and input data. Write unit tests for different components and functional modules of the system. Verify that the system is able to generate the correct study plan based on different inputs. And that it can be successfully and completely exported to other calendar applications. Verify that the system is able to effectively resolve schedule conflicts and provide sound advice.

6. TFR30-S6

Control: Functional, Manual, Dynamic

Initial State: The user has logged in and generated a study schedule plan and a task list with task information and is on Pomodoro Timmer page.

Input: The generated task list with sub-tasks and the information of each sub-task, includes *taskName*, *startTime*, and *endTime*.

Output: A list of Pomodoro Timmers of subtasks.

Test Case Derivation: Using a valid list of sub-tasks and valid Pomodoro clock settings, e.g. 25 minutes of work and 5 minutes of rest, detect that each sub-

task has a corresponding Pomodoro clock with work and rest time slots. Using an empty list of sub-tasks, the output should be that no Pomodoro clocks are created and no warning messages or errors appear. Entering an invalid sub-task list or invalid Pomodoro clock settings, the user should get a warning message that no Pomodoro clock can be created.

How test will be performed: Testing can be performed either by manually operating the system or by automated test scripts. The flow of the learning cycle should be tested to ensure that the Pomodoro clock is working as expected, and the user interactions and system responses should be tested to ensure that the system can correctly handle the user's actions and changes in the task state. Tests should cover different scenarios, including normal operation, incorrect operation, and edge cases.

7. TFR31-S7

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and generated a study schedule plan and a task list with task information and is on Pomodoro Timmer page.

   Input: The user's personal account profile information, and the user's contacts list, which includes a list of contacts that the user has added and their account information. The user's and the contacts' study schedules, including course lists, task lists, and sub-task schedule lists.

   Output: The system will match the study schedules of the user and contacts and output a set of matched contact lists.

   Test Case Derivation: Simulate the user to enter his study plan and request the system to match contacts. Check if the system has successfully matched contacts that fit the study plan and outputs a list of matches. If the user sends a match request, the user's contacts should be able to receive a notification of the request, and after the contacts agree, the system should generate a schedule and a link to the online learning session and send it to the user and the contacts who accepted the request. If the contact does not accept the match request, the contact can choose to leave a message for the user, after which the user should be able to receive an alert message indicating that the match was not successful and the corresponding message from the contact.

   How test will be performed: Testing is performed by manually simulating the user's interaction with the system; the testing steps include entering a learning plan, selecting matching contacts, viewing match results, and scheduling

an online learning session. The testing process should cover a variety of scenarios, including matching under normal circumstances, no-match scenarios, updating of the contacts list, and correctness of session scheduling. The system should also handle boundary cases of input data, such as empty learning plans, incomplete user information, etc.

4.1.5 Scheduling

1. TFR25-S1

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and is on the task creation page or the extraction page.

   Input: the task information such as *taskName*, *taskType*, *weight*, and *priority*. The user's input for the initial estimated time needed to finish the task.

   Output: the initial estimate of the total time required for the task to complete.

   Test Case Derivation: The user should be able to view an initial estimated completion time for any task generated from the course syllabus PDF documents they upload. This estimated completion time can be modified by the user at any time after the task has been created. For any tasks created manually by users, the user can assign an initial estimated completion time period and change it at any time after the task has been created.

   How the test will be performed: Generate tasks from course syllabus PDF files and create tasks directly. Testing should encompass various types of tasks with different weights, user actions, different learning time records, and progress percentage inputs. Manually try to modify the value after the tasks have been created, and try input the various values like negative numbers or text, etc.

2. TFR26-S2

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and generated or created a task list, used Pomodoro Timmers to record work time, or used a checklist or calendar view on the website to record their task progress each time after completing the scheduled study time period.

Input: Current schedule data for the task includes *deadline*, *priority*, *weight*, current progress in completing the task, the user's history of the percentage of progress completed, and the corresponding time spent on the task.

Output: The updated Learning Plan, regenerated task schedule information based on the updated task progress from the user, which may include reordering of task list and TO-DO list, new priority of the task, etc.

Test Case Derivation: Each time the user updates the task progress and records the corresponding time spent, they will receive a reminder to update the learning schedule. After that, the user should be able to view a newly generated learning plan associated with the rate of change in task progress.

How test will be performed: After simulating the user's change of task progress, check whether the system has kept all the original task information and updated the related task plan. Compare the pre- and post-update learning plans to ensure that the updated plan correctly reflects the changes in task progress.

3. TFR27-S3

Control: Functional, Manual, Dynamic

Initial State: The user has logged in and generated or created a task list, used Pomodoro Timmers to record work time, or used a checklist view or calendar view on the website to record their task progress each time after completing the scheduled study time period.

Input: *taskName*, *taskType*, *weight* and the initial estimate of the total time required for the task to complete, the percentage of completing progress of the task recorded by the user at the end of the Pomodoro Timmers, and the corresponding time spent.

Output: The time needed to finish each task is dynamically updated accordingly.

Test Case Derivation: The system should be capable of calculating the most up-to-date estimated time required to finish each task after the user records and inputs the progress percentage and the corresponding time spent.

How the test will be performed: Generate tasks from course syllabus PDF files and create tasks directly. Input multiple progress updates and corresponding study hours for each task. Estimate task completion time through both manual and automated testing methods. Testing should encompass various types of tasks with different weights, user actions, different learning time records, and

progress percentage inputs. Compare the pre- and post-input of the new task progress and corresponding hours spent and check whether the changes in task estimated time needed are accurately reflected in the task data.

4. TFR28-S4

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and uploaded a course syllabus PDF file, and is on the extraction page of the website.

   Input: The user's task list and data for tasks including *taskType*, *weight* and *deadline*.

   Output: A task priority list, based on sorting the tasks in order by task type, weight, and deadline. High-priority tasks will be ranked first.

   When the user attempts to extract all the information from the course, the program will extract the necessary data from the uploaded course syllabus PDF file and generate a list of tasks with a calculated priority number. The system should categorize all tasks into three priority levels: high, medium, and low, and assign each task a corresponding priority label. After this process, the user should be able to view the task lists with priority labels, ordered in descending order based on the priorities generated by the system.

   How test will be performed: Create test sets with task lists of different task types, weights, and deadlines, and check that the generated list of sorted tasks meets the expected priority ordering. Using special case inputs, such as an empty task list, to ensure that the system handles edge cases correctly.

5. TFR29-S5

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and generated or created a task list with task information.

   Input: Task information: *taskName*, *taskType*, *weight*, *deadline priority*. The learner's preferred day and time period for the study. The user's current schedule, including scheduled events, and other existing tasks. The working time of the Pomodoro Timmer setting for each learning task.

   Output: A complete schedule of the learning program, including the name, weight, deadline, priority, sub-tasks under each task, scheduled start and end

times of Pomodoro Timmer for each sub-tasks. Display task and sub-task time periods and information on other external platforms.

Test Case Derivation: Select different study periods, days, and various Pomodoro interval settings to assess the system's ability to handle invalid or incomplete input data. Using valid data, confirm that the system can generate a comprehensive study plan for each task, considering preferred study times and schedules while avoiding conflicts with existing activities. Export study schedules to an external calendar program and test the system's capability to accurately and completely export study plans to other calendar applications. In cases of task scheduling conflicts, verify that the system can provide solutions, such as suggesting alternative study times or allocating sub-tasks to different days.

How test will be performed: Tests can be performed using automated testing frameworks to simulate various scenarios and input data. Write unit tests for different components and functional modules of the system. Verify that the system is able to generate the correct study plan based on different inputs. And that it can be successfully and completely exported to other calendar applications. Verify that the system is able to effectively resolve schedule conflicts and provide sound advice.

6. TFR30-S6

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and generated a study schedule plan and a task list with task information and is on Pomodoro Timmer page.

   Input: The generated task list with sub-tasks and the information of each sub-task, includes *taskName*, *startTime*, and *endTime*.

   Output: A list of Pomodoro Timmers of subtasks.

   Test Case Derivation: Using a valid list of sub-tasks and valid Pomodoro clock settings, e.g. 25 minutes of work and 5 minutes of rest, detect that each sub-task has a corresponding Pomodoro clock with work and rest time slots. Using an empty list of sub-tasks, the output should be that no Pomodoro clocks are created and no warning messages or errors appear. Entering an invalid sub-task list or invalid Pomodoro clock settings, the user should get a warning message that no Pomodoro clock can be created.

How test will be performed: Testing can be performed either by manually operating the system or by automated test scripts. The flow of the learning cycle should be tested to ensure that the Pomodoro clock is working as expected, and the user interactions and system responses should be tested to ensure that the system can correctly handle the user's actions and changes in the task state. Tests should cover different scenarios, including normal operation, incorrect operation, and edge cases.

7. TFR31-S7

   Control: Functional, Manual, Dynamic

   Initial State: The user has logged in and generated a study schedule plan and a task list with task information and is on Pomodoro Timmer page.

   Input: The user's personal account profile information, and the user's contacts list, which includes a list of contacts that the user has added and their account information. The user's and the contacts' study schedules, including course lists, task lists, and sub-task schedule lists.

   Output: The system will match the study schedules of the user and contacts and output a set of matched contact lists.

   Test Case Derivation: Simulate the user to enter his study plan and request the system to match contacts. Check if the system has successfully matched contacts that fit the study plan and outputs a list of matches. If the user sends a match request, the user's contacts should be able to receive a notification of the request, and after the contacts agree, the system should generate a schedule and a link to the online learning session and send it to the user and the contacts who accepted the request. If the contact does not accept the match request, the contact can choose to leave a message for the user, after which the user should be able to receive an alert message indicating that the match was not successful and the corresponding message from the contact.

   How test will be performed: Testing is performed by manually simulating the user's interaction with the system; the testing steps include entering a learning plan, selecting matching contacts, viewing match results, and scheduling an online learning session. The testing process should cover a variety of scenarios, including matching under normal circumstances, no-match scenarios, updating of the contacts list, and correctness of session scheduling. The system should also handle boundary cases of input data, such as empty learning plans, incomplete user information, etc.

4.2 Tests for Nonfunctional Requirements

4.2.1 Look and feel

1. TAR-1

   Type: Dynamic, Manual

   Initial State: The web application is launched.

   Input/Condition: Users are asked to read the text on each page.

   Output/Result: MIN_UNDERSTAND% of users reporting that all the texts are legible.

   How test will be performed: The web application displayed with screens with various resolutions and sizes. MIN_TESTER_NUM users are asked to read the texts and report any unclear typography.

2. TAR-2

   Type: Dynamic, Manual

   Initial State: The web application is launched.

   Input/Condition: Users are asked to observe color on each page.

   Output/Result: MIN_UNDERSTAND% of users comfortable with the color used.

   How test will be performed: MIN_TESTER_NUM users are asked to observe the color and report any distracting or overwhelming color palette.

3. TAR-3

   Type: Dynamic, Manual

   Initial State: The web application is launched.

   Input/Condition: Users are asked to interpret icons and graphics on each page.

   Output/Result: MIN_UNDERSTAND% of users able to understand the meaning of each icon and graphic as designed.

   How test will be performed: MIN_TESTER_NUM users are asked what they think each icon and graphic means. Any mismatch from the UI designer's intention would be recorded.

4. TAR-4

   Type: Dynamic, Manual

   Initial State: The web application is launched.

   Input/Condition: Users are asked to observe the layout of each page.

   Output/Result: MIN_UNDERSTAND% of users comfortable with the payout.

   How test will be performed: The web application is launched with different devices including smartphones, tablets, and desktops. MIN_TESTER_NUM users are asked to point out any unresponsive, out-of-margin, or compressed elements.

5. TSTR-1

   Type: Dynamic, Manual

   Initial State: The web application is launched.

   Input/Condition: Users are asked to point out the menus and navigation bars.

   Output/Result: MIN_UNDERSTAND% of users able to locate menus and navigation tools without confusion.

   How test will be performed: MIN_TESTER_NUM users are asked to point out the menus and navigation bars without assistance.

6. TSTR-2

   Type: Dynamic, Manual

   Initial State: The web application is launched.

   Input/Condition: Users are asked to point out inconsistent UI elements.

   Output/Result: MAX_ELEMENT_BAD inconsistent UI elements found.

   How test will be performed: MIN_TESTER_NUM users are asked to explore the application and point out any inconsistent UI elements that break the harmony.

7. TSTR-3

   Type: Dynamic, Manual

   Initial State: The web application is launched.

Input/Condition: Users are asked to click each interactive UI element.

Output/Result: MAX_ELEMENT_BAD interactive UI elements being unresponsive.

How test will be performed: MIN_TESTER_NUM users are asked to explore the application and report any unresponsive interactive elements.

8. TSTR-4

   Type: Dynamic, Manual

   Initial State: The web application is launched.

   Input/Condition: Users are asked to adjust font size.

   Output/Result: MAX_UNSATISFIED% of users reporting unsuitable font size.

   How test will be performed: MIN_TESTER_NUM users are asked to report how they like the default font size and attempt to adjust the font size to their preference. Report if the adjusted font still causes difficulty in reading.

9. TSTR-5

   Type: Dynamic, Manual

   Initial State: The web application is launched.

   Input/Condition: Users are asked to perform a task while an animation is displayed.

   Output/Result: MIN_UNDERSTAND% of users who could perform the task distracted.

   How test will be performed: MIN_TESTER_NUM users are asked to perform a task that involves the animation display part of the screen. They are asked to report if they feel the animation is intrusive or distracting.

4.2.2 Usability and Humanity

Type: Dynamic, Manual Initial State: The web application is launched. Input/Condition: Users are asked to navigate to a required page. Output/Result: MIN_UNDERSTAND% of users able to navigate to a required page without confusion. How test will be performed: MIN_TESTER_NUM users are asked to navigate to a required page without assistance within a reasonable amount of time undergoing minimum trial and error.TUHR-2

Type: Dynamic, Manual Initial State: The web application is launched. Input/Condition: Users are asked to perform a core function. Output/Result: MIN_UNDERSTAND% of users able to perform a core function. How the test will be performed: MIN_TESTER_NUM users are asked to perform a core function like generating a study plan without assistance within a reasonable amount of time undergoing minimum trial and error. TUHR-3

Type: Dynamic, Manual Initial State: The web application is launched. Input/Condition: Users are asked to inspect the grammar of texts on the page. Output/Result: MAX_BAD_GRAMMAR grammar mistakes found. How test will be performed: MIN_TESTER_NUM users are asked to report grammar mistakes in all the text visible in this web application. TUHR-4

Type: Dynamic, Manual Initial State: The web application is launched. Input/Condition: Users are asked to inspect for offensive messages on the page. Output/Result: MAX_OFFENSIVE offensive messages found. How test will be performed: MIN_TESTER_NUM users are asked to report offensive messages in all the text visible in this web application. TUHR-5

Type: Dynamic, Manual Initial State: The web application is launched. Input/Condition: Users are asked to observe color combinations on the page. Output/Result: MAX_COLOR_AMBIGUOUS indistinguishable color combinations found. How test will be performed: MIN_TESTER_NUM users are asked to report indistinguishable color combinations in this web application. This user group should include people with color blindness.

4.2.3 Area of Testing2

...

4.3 Traceability Between Test Cases and Requirements

[Provide a table that shows which test cases are supporting which requirements. —SS]

5 Unit Test Description

[This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS]

[To save space and time, it may be an option to provide less detail in this section. For the unit tests you can potentially layout your testing strategy here. That is, you can explain how tests will be selected for each module. For instance, your test building approach could be test cases for each access program, including one test for normal behaviour and as many tests as needed for edge cases. Rather than create the details of the input and output here, you could point to the unit testing code. For this to work, you code needs to be well-documented, with meaningful names for all of the tests. —SS]

5.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

5.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

5.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. TPAR-1

2. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Input/Condition: A course outline is uploaded.

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

3. test-id2

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

4. ...

5.2.2 Operational and Environmental

1. TOER-1

   Type: Dynamic, Automated

   Initial State: The web application is launched.

   Input/Condition: Send requests to supported calendar APIs.

   Output/Result: `MIN_API_SUCCESS%` of successful API requests.

   How test will be performed: An automated script is used to send requests to calendar APIs and record the result.

2. TOER-2

   Type: Dynamic, Automated

Initial State: The web application is launched.

Input/Condition: Run regression test suites.

Output/Result: MIN_REGRESSION_PASS% of passed regression tests.

How test will be performed: An automated script is used to run regression tests.

5.2.3 Maintainability and Support

1. TSPR-1

   Type: Dynamic, Manual

   Initial State: The web application is launched.

   Input/Condition: Users are asked to contact support.

   Output/Result: MAX_SUPPORT_STEP steps needed to reach help.

   How test will be performed: MIN_TESTER_NUM users are asked to reach the helpdesk through email, phone, and chatbot and record the steps it takes from the main page to help.

2. TSPR-2

   Type: Dynamic, Automatic

   Initial State: The web application is launched.

   Input/Condition: User feedback is inputted.

   Output/Result: User input in database.

   How test will be performed: A script is used to input user feedback and check if this feedback gets stored in the database.

5.2.4 Module ?

...

5.3 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

parameter	value	unit	description
MIN_UNDERSTAND%	95	N/A	the minimum percentage of testers who can understand among all testers
MAX_UNSATISFIED%	5	N/A	the maximum percentage of testers reporting uncomfortable
MAX_TRIAL_TIME	1200	s	the maximum allowed trial time
MIN_TESTER_NUM	20	N/A	the minimum number of testers needed
MAX_BAD_GRAMMAR	0	N/A	the maximum occurrence of grammar mistakes allowed
MAX_OFFENSIVE	0	N/A	the maximum occurrence of offensive messages allowed
MAX_ELEMENT_BAD	0	N/A	the maximum occurrence of inconsistent or unresponsive elements allowed
MAX_ATTACK_SUCCESS	0	N/A	the maximum occurrence of successful attack
MAX_COLOR_AMBIGUOUS	0	N/A	the maximum occurrence of indistinguishable color combinations allowed
MAX_TIME_PROCESS	2	s	the maximum processing time for a core function
MIN_PRECISION%	95	N/A	the minimum precision of the algorithm
MIN_OPERABLE%	95	N/A	the minimum percentage of system being operable

MIN_API_SUCCESS%	95	N/A	the minimum percentage of successful API calls
MIN_REGRESSION_PASS%	100	N/A	the minimum percentage of successful API calls
MAX_RESPONSE_TIME	24	h	The maximum issue resolve response time
MIN_LANGUAGE	5	N/A	The minimum number of languages that can be translated
MAX_SUPPORT_STEP	5	N/A	The maximum steps needed for asking for support

6.2 Usability Survey Questions

[This is a section that would be appropriate for some projects. —SS]

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

[This section is not required for CAS 741 —SS]

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.

    - UI/UX usability validation tools such as *UserTesting*, *Lookback.io*. to better evaluate our product is user-friendly in a couple of perspectives: effective, learnable, and user-friendly.
    - Dynamic Testing Tools such as *Behave*, which is a tool that allows users to write the test cases in human languages to test for python-system framework.
    - AI Model Validation Frameworks such as *Snitch AI* and *scikit-learn* which can help our trained morel enhance quality and troubleshoot quickly.
    - Static Code Analysis Tools such as *SonarQube* to ensure the code quality which also can be integrated with *CI/CD* for continuous development
    - Enhance continuous delivery/deployment by exploring the *Actions* features in *GitHub* Pro to build custom workflow pipeline.

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?

Knowledge or Skills	**Approaches**	**Assigned Team Member**	**Reason**
UI/UX Usability validation	Use *ChatGPT*, *Google*, watch online tutorials, or ask supervisor for help	Shuting, Shi	Working on the initial UI design, familiar with the key features and the components of website. Therefore, can detect the usability requirements of our target user groups and easy to make modifications accordingly
Dynamic Testing Tools	Use *ChatGPT*, *Google*, watch online tutorials	Qiang, Gao	Have the related experience in the previous co-op work terms, implemented similar functionality in previous project. Strong interest in the dynamic testing section.
AI Model Validation Framework	Use *ChatGPT*, *Google*, watch online tutorials	Qianni, Wang	Experience with many ML projects where these libraries are being used in AI programs and previous co-op work terms. Working on the model training, data-sets selection and integration, familiar with the model algorithm, easy to do modifications if encounters specific model bias.
Static Code Analysis Tools	Use *ChatGPT*, *Google*, watch online tutorials	Chenwei, Song	Experience in enhancing clean code in previous co-op work terms. Strong interest in the code analysis section.
GitHub Action Feature	Use *ChatGPT*, *Google*, and watch online tutorials	Jingyao, Qin	Strong interest in GitHub features, have related experience in the previous coop term, quick to hand on this technique.