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 1, 2023

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

[The intention of the VnV plan is to increase confidence in the software. However, this does not mean listing every verification and validation technique that has ever been devised. The VnV plan should also be a **feasible** plan. Execution of the plan should be possible with the time and team available. If the full plan cannot be completed during the time available, it can either be modified to "fake it", or a better solution is to add a section describing what work has been completed and what work is still planned for the future. —SS]

[The VnV plan is typically started after the requirements stage, but before the design stage. This means that the sections related to unit testing cannot initially be completed. The sections will be filled in after the design stage is complete. the final version of the VnV plan should have all sections filled in. —SS]

Contents

List of Tables

[Remove this section if it isn't needed —SS]

List of Figures

[Remove this section if it isn't needed —SS]

1 Symbols, Abbreviations, and Acronyms

symbol	description
T	Test

[symbols, abbreviations, or acronyms — you can simply reference the SRS (Author, 2019) tables, if appropriate —SS]

[Remove this section if it isn't needed —SS]

This document ... [provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

2 General Information

2.1 Summary

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

2.2 Objectives

[State what is intended to be accomplished. The objective will be around the qualities that are most important for your project. You might have something like: "build confidence in the software correctness," "demonstrate adequate usability." etc. You won't list all of the qualities, just those that are most important. —SS]

[You should also list the objectives that are out of scope. You don't have the resources to do everything, so what will you be leaving out. For instance, if you are not going to verify the quality of usability, state this. It is also worthwhile to justify why the objectives are left out. —SS]

[The objectives are important because they highlight that you are aware of limitations in your resources for verification and validation. You can't do everything, so what are you going to prioritize? As an example, if your system depends on an external library, you can explicitly state that you will assume that external library has already been verified by its implementation team. —SS]

2.3 Relevant Documentation

[Reference relevant documentation. This will definitely include your SRS and your other project documents (design documents, like MG, MIS, etc). You can include these even before they are written, since by the time the project is done, they will be written. —SS]

Author (2019)

[Don't just list the other documents. You should explain why they are relevant and how they relate to your VnV efforts. —SS]

3 Plan

[Introduce this section. You can provide a roadmap of the sections to come. —SS]

3.1 Verification and Validation Team

[Your teammates. Maybe your supervisor. You should do more than list names. You should say what each person's role is for the project's verification. A table is a good way to summarize this information. —SS]

3.2 SRS Verification Plan

[List any approaches you intend to use for SRS verification. This may include ad hoc feedback from reviewers, like your classmates, or you may plan for something more rigorous/systematic. —SS]

[Maybe create an SRS checklist? —SS]

3.3 Design Verification Plan

[Plans for design verification —SS]

[The review will include reviews by your classmates —SS]

[Create a checklists? —SS]

3.4 Verification and Validation Plan Verification Plan

[The verification and validation plan is an artifact that should also be verified. Techniques for this include review and mutation testing. —SS]

[The review will include reviews by your classmates —SS]

[Create a checklists? —SS]

3.5 Implementation Verification Plan

[You should at least point to the tests listed in this document and the unit testing plan. —SS]

[In this section you would also give any details of any plans for static verification of the implementation. Potential techniques include code walkthroughs, code inspection, static analyzers, etc. —SS]

3.6 Automated Testing and Verification Tools

[What tools are you using for automated testing. Likely a unit testing framework and maybe a profiling tool, like ValGrind. Other possible tools include a static analyzer, make, continuous integration tools, test coverage tools, etc. Explain your plans for summarizing code coverage metrics. Linters are another important class of tools. For the programming language you select, you should look at the available linters. There may also be tools that verify that coding standards have been respected, like flake9 for Python. —SS]

[If you have already done this in the development plan, you can point to that document. —SS]

[The details of this section will likely evolve as you get closer to the implementation. —SS]

3.7 Software Validation Plan

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

[You might want to use review sessions with the stakeholder to check that the requirements document captures the right requirements. Maybe task based inspection? —SS]

[For those capstone teams with an external supervisor, the Rev 0 demo should be used as an opportunity to validate the requirements. You should plan on demonstrating your project to your supervisor shortly after the scheduled Rev 0 demo. The feedback from your supervisor will be very useful for improving your project. —SS]

[For teams without an external supervisor, user testing can serve the same purpose as a Rev 0 demo for the supervisor. —SS]

[This section might reference back to the SRS verification section. —SS]

4 System Test Description

4.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

[Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. —SS]

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

4.1.2 User Input

6. TFR5-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

7. TFR5-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

8. TFR6-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

9. TFR7-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

10. TFR8-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

11. TFR9-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

12. TFR10-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

13. TFR10-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

14. TFR10-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

15. TFR11-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

16. TFR12-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

17. TFR13-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

18. TFR14-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

[]

19. TFR15-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

20. TFR16-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

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. Not all projects will necessarily have nonfunctional requirements related to accuracy —SS]

[Tests related to usability could include conducting a usability test and survey. The survey will be in the Appendix. —SS]

[Static tests, review, inspections, and walkthroughs, will not follow the format for the tests given below. —SS]

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

[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. test-id1

   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:

2. 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:

3. ...

5.2.2 Module 2

...

5.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

5.3.1 Module ?

1. test-id1

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

   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

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

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?

[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.

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?