Project Title: System Verification and Validation Plan for Software Eng 4G06

Team 2, Parnas' Pals
William Lee
Jared Bentvelsen
Bassel Rezkalla
Yuvraj Randhawa
Dimitri Tsampiras
Matthew McCracken

November 1, 2022

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

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]

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

[symbols, abbreviations or acronyms — you can simply reference the SRS (**?**) 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]

3 General Information

3.1 Summary

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

3.2 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]

3.3 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]

?

4 Plan

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

4.1 Verification and Validation Team

[Your teammates. Maybe your supervisor. You shoud 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]

4.2 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]

4.3 Design Verification Plan

[Plans for design verification —SS]

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

[Create a checklists? —SS]

4.4 Verification and Validation Plan Verification Plan

[The verification and validation plan is an artifact that should also be verified. —SS]

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

[Create a checklists? —SS]

4.5 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]

4.6 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]

4.7 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]

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

5 System Test Description

5.1 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]

5.1.1 Workout Routine Tests

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

A Workout Routine can be Created

1. **test-WR-1**

   Control: Manual

   Initial State: Application is running.

Input: User inputs required data for a workout routine.

Output: A workout routine is stored in the database and is accessible to the user that created it. If the user determined that the workout to be public then it should be publicly visible.

Test Case Derivation: On a successful creation, the workout routine should be placed into the database.

How test will be performed: This test can be preformed with a manual insertion of creating a workout routine under a test user. These results can be determined by viewing the database and scope of visibility of the created routine.

Requirement(s): R1

2. **test-WR-2**

   Control: Manual

   Initial State: Application is running with an existing workout routine.

   Input: User edits a workout routine with new values.

   Output: A workout routine is updated with new values in the database and new values are visible to accessible users.

   Test Case Derivation: On a successful edit, the workout routine should be updated with new values database.

   How test will be performed: This test can be preformed with a manual edit of a workout routine under a test user.

   Requirement(s): R1

5.1.2 Exercise Tests

3. **test-EX-1**

   Control: Manual

   Initial State: Application is running and a workout routine is in the process of being created or edited.

Input: A user created exercise with the parameters required for the exercise.

Output: The workout routine should include the added exercise.

Test Case Derivation: Because exercises are bounded to a specific workout routine, the routine that the exercise was created under should include the new exercise.

How test will be performed: Given a workout routine, an exercise will be manually added to determine if it is linked and added to the workout routine.

Requirement(s): R2

4. **test-EX-2**

   Control: Manual

   Initial State: Application is running and a workout routine is in the process of being created or edited.

   Input: The user chooses to remove an exercise.

   Output: Some feedback indicating that an exercise has been removed. The workout routine no longer contains the removed exercise.

   Test Case Derivation: On removal of an exercise, there should be an indication towards the user notifying them of the removal. The exercise should also be deleted from the workout routine.

   How test will be performed: Given a workout routine, an exercise will be manually removed and checked to ensure that it no longer exists within the given workout routine.

   Requirement(s): R2

5. **test-EX-3**

   Control: Manual

   Initial State: Application is running and a workout routine is in the process of being created or edited.

Input: The number of exercises required to reach the limit of exercises per workout routine.

Output: A notification to the user, before the limiting exercise notifying them of the limit, and preventing them from adding another exercise.

Test Case Derivation: The user should be aware of the exercise limit per workout routine and should not be able to exceed it.

How test will be performed: The test will be preformed by manually added exercises to a workout routine until the limit. Then upon adding the limiting input, a notification is expected.

Requirement(s): R2

5.1.3 Quantifier Tests

6. **test-QT-1**

   Control: Manual

   Initial State: The application is running and an exercise is in the process of being created or edited.

   Input: A number and unit of measurement describing an exercise.

   Output: The exercise now holds the given quantifier and is displayed to the user.

   Test Case Derivation: On adding a quantifier to an exercise, there should be a unit of measurement and value associated with it.

   How test will be performed: This test will be executed by adding a quantifier to various types of exercises to ensure they correctly measure the exercise.

   Requirement(s): R3

7. **test-QT-2**

   Control: Manual

   Initial State: The application is running and an exercise is in the process of being created or edited.

   Input: Removal of a quantifier.

   Output: The exercise no longer holds a quantifier and is not displayed to the user.

   Test Case Derivation: On removing a quantifier to an exercise, it should no longer be visible to the user.

   How test will be performed: This test will be executed by removing a quantifier from an exercises to ensure it does not display any quantifying information.

   Requirement(s): R3

8. **test-QT-3**

   Control: Manual

   Initial State: The application is running and an exercise is in the process of being created or edited.

   Input: A number and unit of measurement describing an exercise.

   Output: The exercise now holds the updated quantifier.

   Test Case Derivation: On updating a quantifier to an exercise, there should be a new unit of measurement and/or value associated with it.

   How test will be performed: This test will be executed by editing a quantifier and ensuring the values are updated on completion of the edit.

   Requirement(s): R3

5.1.4 Publicity Tests

9. **test-PB-1**

   Control: Manual

   Initial State: The application is running and a private workout routine is created.

   Input: An edit or addition to a workout routine to make it public.

   Output: The workout routine is declared public in the database and is now visible to all users.

   Test Case Derivation: This is the expected output because if the workout routine was not public before this action then it should not be visible to users. Once updating the publicity, the database should be updated and the routine should be publicly available

   How test will be performed: This test will be done by manually changing the state of a workout routine and checking for a database update and viewing the routine under a different test-user.

   Requirement(s): R4

10. **test-PB-2**

    Control: Manual

    Initial State: The application is running and a public workout routine is created.

    Input: An edit or addition to a workout routine to make is private.

    Output: The workout routine is declared private in the database and is no longer visible to any user except the creator.

    Test Case Derivation: By privatizing a routine, it should no longer be accessible to any other user than the creator. There should also be a database update to signify this.

    How test will be performed: This test will be done by manually changing the state of a workout routine and checking for a database update and viewing the routine under a different test-user.

    Requirement(s): R4

5.1.5 Workout Routine Saving Tests

11. **test-WS-1**

    Control: Manual

    Initial State: The application is running and there is a public workout routine created by another user.

    Input: The other user workout routine is saved.

    Output: The workout routine is now visible and accessible under the saved workout routines.

    Test Case Derivation: On saving a workout routine, it should be stored somewhere the user can access it for later use.

    How test will be performed: This test will be done by creating a public workout routine on one test-user. On a different test user this routine will be saved. The saved workout routines should then include the newly saved one.

    Requirement(s): R5

5.1.6 Browsing Workout Routine Tests

12. **test-BS-1**

    Control: Manual

    Initial State: The application is running and multiple workout routines are created.

    Input: navigation movements to view the public workout routines.

    Output: Multiple public workout routines should be displayed.

    Test Case Derivation: On making a routine public, it should be available to all users and also the browsing page.

    How test will be performed: By first adding multiple public test routines, then checking the public workout routines by browsing. There should exist routines to access and view.

Requirement(s): R6

13. **test-BS-2**

    Control: Manual

    Initial State: The application is running and multiple workout routines are created.

    Input: Search string inputs to view the public workout routines.

    Output: Public workout routines that match the search criteria should be displayed.

    Test Case Derivation: All workout routines that match a search, should be displayed.

    How test will be performed: By first adding multiple public test routines, then preforming a search, the routines that match the search criteria should be displayed.

    Requirement(s): R6

5.1.7 User Profile Tests

14. **test-UR-1**

    Control: Manual

    Initial State: The application is running.

    Input: The required parameters for creating a profile.

    Output: The created profile is stored in a user database and the user should be logged into their profile.

    Test Case Derivation: On creating a profile the database must store the information. To go along with this the user must maintain the login status of the created profile.

How test will be performed: On launching the application without a created profile a profile will be created. The database will then be checked to ensure the proper data matches.

Requirement(s): R7

15. **test-UP-2**

    Control: Manual

    Initial State: The application is running and there exists another user profile.

    Input: Search criteria for the searched user profile.

    Output: A user profile is displayed with their public routines, fitness goals, and other public profile data.

    Test Case Derivation: On searching for another user the data shown should only be what is made public.

    How test will be performed: By creating a test-user profile then searching for it. The profile should display all public information and hide non-public information.

    Requirement(s): R8

5.1.8 Fitness Goal Tests

16. **test-FG-1**

    Control: Manual

    Initial State: The application is running and a profile has been created or is in the process of creation.

    Input: A string parameter for a fitness goal and publicity options for the goal.

    Output: The fitness goal for the user is now updated in the database and is saved under their profile.

Test Case Derivation: The fitness goals are specific to a given user so the data for fitness goals are stored under their profile data.

How test will be performed: A test user will add a fitness goal and check if it is displayed on their profile.

Requirement(s): R9

17. **test-FG-2**

    Control: Manual

    Initial State: The application is running and a fitness goal is created.

    Input: Progress points towards a given fitness goal at a given date.

    Output: The progress displayed towards a fitness goal should be visually updated and numerically updated in the database with a date.

    Test Case Derivation: The data for a fitness goal must be updated in the database for storage. To follow this, to goal must also be visually updated accordingly to signify progress towards the goal at the given date.

    How test will be performed: A test-user will create a fitness goal and add progress points towards the goal. There should be a database update and visual update to signify this change.

    Requirement(s): R10, R11

5.1.9 Area of Testing2

...

5.2 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]

5.2.1 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:

5.2.2 Area of Testing2

...

5.3 Traceability Between Test Cases and Requirements

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

6 Unit Test Description

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS] [This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

6.1 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]

6.2 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]

6.2.1 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. ...

6.2.2 Module 2

...

6.3 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]

6.3.1 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:

6.3.2 Module ?

...

6.4 Traceability Between Test Cases and Modules

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

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

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

7.2 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

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?