System Verification and Validation Plan for Digital Twin Forest

Team 8, Forest Mirror
Yichen Jiang
Bowen Zhang
Jiacheng Wu
Junhong Chen
Tingyu Shi

November 2, 2022

1 Revision History

Date	Version	Notes
October 29th	1.0	Initial Document

Contents

List of Tables

List of Figures

2 Symbols, Abbreviations and Acronyms

The follow are some naming conventions and definitions from SRS:

- LiDAR: Light Detection and Ranging(Scanning Technology)
- Plot: A square shaped area in the forest. There are 13 plots in total.
- Target Forest: The natural forest modelled in this product.
- Digital Twin Forest: The virtual representation of the target forest.
- Update local data: The users are allowed to update any data on their own device. The update made by a user will not be updated to the other users.
- Update official data: The developing team would release updates regularly. The update made by the developing team can be updated to all users if they accept the latest version.
- Overall forest data: These are data for the whole forest. For example, these data may include overall CO_2 concentration or plant density of the whole forest.
- Forest plot data: The target forest is separated into 13 plots. Plot forest data has the same data types as the overall forest data, however, all the data are for a specific forest plot.
- Tree parameters: Tree parameters include data such as tree ages, perimeters, heights, species, etc.
- Main page: This is the first user interface when users just lunch the system.

3 General Information

3.1 Summary

The software being tested is called *Digital Twin Forest*. This isa virtual representation of the target forest. The product will provide the following general functions:

- Provide 3D models to simulate trees, soil, terrain, atmosphere, etc, in the target forest.
- Provide User Interface to present forest data(Overall forest data and Forest plot data) and tree parameters.
- Allow users to update local forest data and the virtual forest should present the changes.
- Allow users to update official data provided by the developing team.

3.2 Objectives

This document will perform as a guide of examining the quality of our software. The first objective to be accomplished is to make sure correctness of the software. Testing will allow our team to examine all the functions of our product. The software should be executed properly, perform as designed, represent the expected data, and show the model we constructed. The second objective is to examine the usability. The possible methods might include manual testings by the testing team, questionnaires and user interviews.

3.3 Relevant Documentation

The System Verification Plan and Validation Plan mainly focuses on testing functional and non-functional requirements in the SRS documents. After the first submission of the SRS document, our team revised some SRS contents. You can find the latest SRS document here.

4 Plan

This section refers to the general testing plan of the project *Digital Twin Forest*. To start with, it assigns different roles to the team members. Then, it contains the SRS verification plan, which is divided into review by supervisor, reviewed by

classmates and reviewed by teammates. Then, it introduces the design verification which will be completed later. This section also introduced how the team will test codes. Automated testing tools are also given. Last, this section contains the plan for software validation.

4.1 Verification and Validation Team

Team Member	Roles
Yichen Jiang	Test leader
Bowen Zhang	Hardware verification and SRS verification
Junhong Chen	Code verification
Jiacheng Chen	Code verification
Tingyu Shi	SRS verification and Code verification
Dr.Gonsamo	Manual testing as user
Dr.Gonsamo's lab's members	Manual testing as users
Classmates	Manual testing as users

Table 1: Team Members and Roles

4.2 SRS Verification Plan

SRS testing refers to the review the functional and non functional requirements mentioned in the SRS document. In general, an SRS checklist will be created to verify if each requirement has been met. The SRS may be revised by adding or removing requirements based on new users' needs and constraints.

4.2.1 Review by Supervisor

The project is supervised by Dr. Gonsamo and his lab members. Our team meets weekly with graduate students in Dr. Gonsamo's lab and reviews the progress of the project. At the end of the semester, our team will have a formal meeting with Dr. Gonsamo, and the project will be verified to see if it accomplishes all his expectations. Dr. Gonsamo will take the project and do a task-based inspection. For example, the application will be installed on lab computers and functions like measuring distance between trees will be tested to see if it has an acceptable accuracy.

4.2.2 Review by Classmates

Classmates from other groups will be considered users who have no training . Non-functional requirements in SRS can be tested by these reviewers, and they will provide feedback. For example, it is clear that the application is easy to learn if the reviewers successfully access the virtual forest in one minute.

4.2.3 Review by teammates

Team members are the reviewers who are most familiar with the project. The team will go through the SRS document and carry out both system tests and unit tests. The SRS checklist will be filled to see if all the functional and non-functional requirements have been accomplished.

4.3 Design Verification Plan

This part will be revised after MG and MIS are completed.

4.4 Verification and Validation Plan

Verification and validation stand for making sure the application does what it is expected to do and meets users' needs. As mentioned in the SRS verification plan, the project will be reviewed by classmates from other groups. The team has designed a usability survey questionnaire for the reviewers to complete. The usability survey is ad hoc and the questions are focused on the non-functional requirements. As untrained users, the reviewers may provide critical feedback on questions such as "Do you think the product instructions are easy to understand." Meanwhile, the team will prepare a checklist for the requirements. By collecting the questionnaires and doing tests by themselves, the team members will make sure if each requirement in the checklist is accomplished or not.

4.5 Implementation Verification Plan

Implementation verification refers to the high quality code of the application. Code testing is mentioned in the system test description of this document, especially in the tests for nonfunctional requirements. For example, a static testing will be done for look and feel requirements, and the team will perform individual code walkthroughs. The team will follow the module guide and the traceability matrix to see if each function can be traced over 8 modules. Before a team member committing the code

to the GitHub repository, another team member will help him/her do the verification. If the function is not traceable and the test fails, the code needs to be revised before committing.

4.6 Automated Testing and Verification Tools

Automated testing tools, code coverage metrics, and linters have been mentioned in the development plan.
The team will use Visual Studio 2019 as the IDE of the project. The project is implemented in C sharp language, and the built-in linter of VS2019 is good enough to analyze the code and improve the quality. In VS2019, a unit test project of .NET The framework that contains MSTest unit tests will be used for the unit testing. The JetBrains dotCover will be installed as a plugin to check the code coverage. It can do coverage analysis for the unit testing plan.

4.7 Software Validation Plan

Software validation is to guarantee SRS captures the right requirements. As a Meteorologist, Dr. Gonsamo, is not only the supervisor of the project but also a stakeholder. As mentioned in the SRS verification section, Dr. Gonsamo will do a task-based inspection with the team. The functions of the application will be tested by trying to accomplish a range of task just to see if the requirements are necessary.

5 System Test Description

5.1 Tests for Functional Requirements

The system tests about functional requirements are divided into two subsets. The first subset is about testing presentations(what contents will be presented to users). The second subset is about testing if users can interact successfully with the product. According to the newly revised SRS document mentioned in section 3.3, we have 18 functional requirements in total. The following are the corresponding requirements within two subsets

- Presentation = {FR1, FR4, FR5, FR9, FR12}
- Users' interactions with the product = {FR2, FR3, FR6, FR7, FR8 FR10, FR11, FR13, FR14, FR15, FR16, FR17}

5.1.1 Presentation

This section focuses on what contents are presented to users. From the SRS, there are 5 requirements related to the presentations, which are FR1, FR4, FR5, FR9 and FR12. Each functional requirement will have at least one corresponding system test. The following are the system tests for requirements related to presentations.

1. Test-FR1
   Control: Manual

   Initial State: The product is just lunched and running. Main page is presented to users.

   Input: A click event on `Instruction` button on the main page.

   Output: Instruction page will open and instructions about how to use the product will be presented in the instruction page.

   Test Case Derivation: `Instruction` button is designed for opening instruction page that contains instructions. Therefore, a click event on `Instruction` button will open instruction page.

How test will be performed: Let testers lunch the software and click on `Instruction` button 10 times, instruction page with instructions should always open.

2. Test-FR4.1
   Control: Manual

   Initial State: The product is just lunched and running. Main page is presented to users.

   Input: A click event for `Start` button on the main page.

   Output: A progress bar will appear on the screen to indicate how many percentages of forest models are being loaded.

   Test Case Derivation: Sometimes forest models can be large and they take time to be loaded and presented to users. A progress bar can bring better user experience to users by allowing them to know what is happening within the system. Clicking `Start` will trigger the loading process of forest models. While the forest models are being loaded, the progress bar should show on the screen.

   How test will be performed: Let testers lunch the software and click on `Start` button 10 times, the progress bar shall always appear on the screen after clicking `Start`.

   Forest models in this test case mean overall digital twin forest view.

3. Test-FR4.2
   Control: Manual

   Initial State: The overall digital twin forest view with 14 plots and overall forest data is presented to users.

   Input: A click event on any forest plot.

   Output: A progress bar will appear on the screen to indicate how many percentages of forest models are being loaded.

Test Case Derivation: Sometimes forest models can be large and they take time to be loaded and presented to users. A progress bar can bring better user experience to users by allowing them to know what is happening within the system. Clicking a specific forest plot will trigger the loading process of forest models. While the forest models are being loaded, the progress bar should show on the screen.

How test will be performed: Let testers click on each forest plots for 10 times(140 times in total), the progress bar shall always appear on the screen after clicking a forest plot.

Forest models in this test case mean trees models within a specific forest plot.

4. Test-FR5
   Control: Manual

   Initial State: The product is just lunched and running. Main page is presented to users.

   Input: A click event on `Start` button on the main page.

   Output: After the progress bar reaches 100%, full view of overall digital twin forest is presented. Full view here includes 14 forest plots and overall forest data.

   Test Case Derivation: The product will present the digital twin forest in a hierarchical way. It will first display full view of the overall digital twin forest. Then users can choose to display a specific forest plot. Therefore, after clicking `Start`, full view of the overall digital twin forest should appear on the screen after all the models are loaded.

   How test will be performed: Let testers lunch the product and click `Start` button for 10 times, full view of the digital twin forest should always appear after the progress bar reaches 100%. Also, testers need to make sure 14 forest plots and overall forest data can be displayed.

5. Test-FR9.1
   Control: Manual

   Initial State: The product is just lunched and running. Main page is presented to users.

   Input: A click event on `Start` button on the main page.

   Output: After the progress bar reaches 100%, overall forest data should appear on two sides of the screen.

   Test Case Derivation: Placing data on two sides of the screen can leave the center of the screen to show the forest models. By doing this, users will not be interrupted to check forest models.

   How test will be performed: Let testers lunch the product and click `Start` button for 10 times, overall forest data should always appear on two sides of the screen after the progress bar reaches 100%.

6. Test-FR9.2
   Control: Manual

   Initial State: The overall digital twin forest view with 14 plots and overall forest data is presented to users.

   Input: A click event on any forest plot.

   Output: After the progress bar reaches 100%, forest plot data should appear on two sides of the screen.

   Test Case Derivation: Placing data on two sides of the screen can leave the center of the screen to show the forest models. By doing this, users will not be interrupted to check forest models.

How test will be performed: Let testers click each forest plot for 10 times(140 times in total), forest plot data should always appear on two sides of the screen after the progress bar reaches 100%.

7. Test-FR12.1
   Control: Manual

   Initial State: Overall forest data are presented.

   Input: N/A

   Output: N/A

   Test Case Derivation: Since this test does not have any output, this part is not applicable.

   How test will be performed: Let testers observe and record all the overall forest data. Then testers need to check if the data observed from the UI are exactly the same as data stored in the software. If data observed are the same as data stored in the UI, test passes. Otherwise test fails.

8. Test-FR12.2
   Control: Manual

   Initial State: Forest plot data are presented.

   Input: N/A

   Output: N/A

   Test Case Derivation: Since this test does not have any output, this part is not applicable.

   How test will be performed: Let testers observe and record all the forest plot data. Then testers need to check if the data observed from the UI are exactly

the same as data stored in the software. If data observed are the same as data stored in the UI, test passes. Otherwise test fails.

9. Test-FR12.3
   Control: Manual

   Initial State: Tree parameters are presented.

   Input: N/A

   Output: N/A

   Test Case Derivation: Since this test does not have any output, this part is not applicable.

   How test will be performed: Let testers observe and record all the tree parameters. Then testers need to check if the tree parameters observed from the UI are exactly the same as tree parameters stored in the software. If data observed are the same as data stored in the UI, test passes. Otherwise test fails.

5.1.2 Users' interactions with the product

This section focuses on users' interactions with the system. From the SRS, there are 13 requirements related to users' interactions with the system, which are FR2, FR3, FR6, FR7, FR8, FR10, FR11, FR13, FR14, FR15, FR16, FR17 and FR18. Each functional requirement will have at least one corresponding system test. The following are the system tests for requirements related to users' interactions with the system.

1. Test-FR23
   Control: Manual

   Initial State: The product is just lunched and running. Main page is presented to users.

   Input: A click event on `Start` button.

Output: The screen should change from the first user interface to overall digital twin forest view.

Test Case Derivation: If clicking `Start` button can change the main page to overall digital twin forest view. Then we can prove that `Start` button provides a way to allow users to start the virtual tour of the digital twin forest.

How test will be performed: Let testers lunch the product and click `Start` button for 10 times, full view of the digital twin forest should always appear.

2. Test-FR6.1
   Control: Manual

   Initial State: Windows to present overall forest data are presented.

   Input: A click event on `Minimize` button. A sample `Minimize` button can be found in section 7.3.

   Output: Windows to present overall forest data disappear from the screen.

   Test Case Derivation: If clicking `Minimize` button can make windows to present overall forest data disappear from the screen. The we can prove that users can minimize the user interface.

   How test will be performed: Let testers click `Minimize` button for 10 times, windows to present overall forest data always disappear.

3. Test-FR6.2
   Control: Manual

   Initial State: Windows to present forest plot data are presented.

   Input: A click event on `Minimize` button. A sample `Minimize` button can be found in section 7.3.

Output: Windows to present forest plot data disappear from the screen.

Test Case Derivation: If clicking `Minimize` button can make windows to present forest plot data disappear from the screen. The we can prove that users can minimize the user interface.

How test will be performed: Let testers click `Minimize` button for 10 times, windows to present forest plot data always disappear.

4. Test-FR6.3
   Control: Manual

   Initial State: Windows to present tree parameters are presented.

   Input: A click event on `Minimize` button. A sample `Minimize` button can be found in section 7.3.

   Output: Windows to present tree parameters disappear from the screen.

   Test Case Derivation: If clicking `Minimize` button can make windows to present tree parameters disappear from the screen. The we can prove that users can minimize the user interface.

   How test will be performed: Let testers click `Minimize` button for 10 times, windows to present tree parameters always disappear.

5. Test-FR7
   Control: Manual

   Initial State: Overall digital twin forest view is presented. This view includes 14 forest plots and overall forest data.

   Input: A click event on any forest plot.

Output: Windows to present forest plot data should appear on two sides of the screen.

Test Case Derivation: If clicking any forest plot can make windows to present forest plot data appear on two sides of the screen. Then we can prove that the system is able to present forest plot data.

How test will be performed: Let testers click each forest plot for 10 times(140 times in total), windows to present forest plot data should always appear on two sides of the screen.

6. Test-FR8
   Control: Manual

   Initial State: Various tree models are presented on the screen.

   Input: A click event on any tree model.

   Output: A pop-up window should appear to show related tree parameters.

   Test Case Derivation: If clicking any tree model can make pop-up windows to present tree parameters appear on the screen. Then we can prove that the system is able to present tree parameters.

   How test will be performed: Let testers randomly select 20 trees in each forest plot. Since there are 14 forest plots, 280 trees will be selected in total. By clicking on each tree, a pop-up window should always appear on the screen to show tree parameters.

7. Test-FR10
   Control: Manual

   Initial State: Various tree models are presented on the screen.

   Input: Users scroll up or scroll down the scroll wheel on a mouse.

Output: Tree models are zoomed in or zoomed out.

Test Case Derivation: In Unity, the mouse scrolled can be programmed to adjust the distance between the camera and objects. Therefore, scrolling up and the down the scroll wheel should zoom in or zoom out the forest models.

How test will be performed: Let testers click one forest plot, and scroll up or scroll down the mouse scroll wheel. Then testers should observe the screen to make sure that tree models can be zoomed in or zoomed out.

8. Test-FR11
   Control: Manual

   Initial State: Various tree models are presented on the screen.

   Input: Users use the keyboard(clicking `W, A, S, D`).

   Output: Tree models can be presented from different view points.

   Test Case Derivation: In Unity, the keyboard can be programmed to adjust position of the camera. Therefore, using keyboard should change the point of view.

   How test will be performed: Let testers click one forest plot, and scroll up or scroll down the mouse scroll wheel. Then testers should observe the screen to make sure that tree models can be zoomed in or zoomed out.

9. Test-FR13.1
   Control: Manual

   Initial State: Windows showing overall forest data are presented to users.

   Input: A click event on `turn page` button. A sample `turn page` button can be found in section 7.3.

Output: A new page should appear in the same window.

Test Case Derivation: `Turn page` button is designed for turning pages, so clicking `turn page` button should make a new page appear on the screen. If a new page can appear on the screen by clicking `turn page` button, then we can prove that the product allows users to turn pages if the interface has multiple pages.

How test will be performed: Let testers click `turn page` button on the window showing overall forest data. A new page should always appear. Since there are two windows(on both sides of the screen) showing overall forest data, testers need to click `turn page` button for 20 times in total.

10. Test-FR13.2
Control: Manual

Initial State: Windows showing forest plot data are presented to users.

Input: A click event on `turn page` button. A sample `turn page` button can be found in section 7.3.

Output: A new page should appear in the same window.

Test Case Derivation: `Turn page` button is designed for turning pages, so clicking `turn page` button should make a new page appear on the screen. If a new page can appear on the screen by clicking `turn page` button, then we can prove that the product allows users to turn pages if the interface has multiple pages.

How test will be performed: Let testers click `turn page` button on the window showing forest plot data. A new page should always appear. Since there are two windows(on both sides of the screen) showing forest plot data for each forest plot and there are 14 forest plots in total, testers need to click `turn page` button for (20 * 2 * 14 = 560) times in total.

11. Test-FR13.3
Control: Manual

Initial State: Pop-up windows showing tree parameters are presented to users.

Input: A click event on `turn page` button. A sample `turn page` button can be found in section 7.3.

Output: A new page should appear in the same window.

Test Case Derivation: `Turn page` button is designed for turning pages, so clicking `turn page` button should make a new page appear on the screen. If a new page can appear on the screen by clicking `turn page` button, then we can prove that the product allows users to turn pages if the interface has multiple pages.

How test will be performed: Let testers click `turn page` button on the pop-up windows showing tree parameters. A new page should always appear. This test should be done on 20 trees for each forest plot.

12. Test-FR14
Control: Manual

Initial State: A forest plot view is presented to users.

Input: A click event on `Back` button located at the left bottom of the screen.

Output: Overall view of digital twin forest should appear on the screen.

Test Case Derivation: `Back` button is designed for returning to the previous user interface, so clicking `Back` button should show overall digital twin forest view.

How test will be performed: Let testers click `Back` button located at the left bottom of the screen when they are viewing forest plot models. After clicking `Back` button, overall digital twin forest should appear on the screen. This test should be done for 14 forest plots.

13. Test-FR15
Control: Manual

Initial State: Main page is presented to users.

Input: A click event on `Quit` button.

Output: Software should close.

Test Case Derivation: `Quit` button is designed for quitting the software, so clicking `Quit` button quit the software.

How test will be performed: Let testers click `Quit` button located in the main page. After clicking `Quit` button, the software should close.

14. Test-FR16.1
Control: Manual

Initial State: Overall forest data are presented to users.

Input: A click event on `Update` button located at right bottom of the screen.

Output: A window should pop up which allows users to update overall forest data.

Test Case Derivation: `Update` button is designed for users to update data, so a window should pop up to allow users to update overall forest data after clicking `Update` button.

How test will be performed: Let testers click `Update` button located at the right bottom of the screen when they are viewing overall forest data for 20 times. A window should always pop up to allow users to update overall forest data.

15. Test-FR16.2
Control: Manual

Initial State: Forest plot data are presented to users.

Input: A click event on `Update` button located at right bottom of the screen.

Output: A window should pop up which allows users to update forest plot data.

Test Case Derivation: `Update` button is designed for users to update data, so a window should pop up to allow users to update forest plot data after clicking `Update` button.

How test will be performed: Let testers click `Update` button located at the right bottom of the screen when they are viewing forest plot data for 20 times. A window should always pop up to allow users to update forest plot data. This test should be done for all 14 forest plots.

16. Test-FR16.3
    Control: Manual

    Initial State: Tree parameters are presented to users.

    Input: A click event on `Update` button located at right bottom of the window presenting tree parameters.

    Output: A window should pop up which allows users to update tree parameters.

    Test Case Derivation: `Update` button is designed for users to update data, so a window should pop up to allow users to update tree parameters after clicking `Update` button.

    How test will be performed: Let testers click `Update` button located at the right bottom of the window presenting tree parameters for 20 times. A window should always pop up to allow users to update tree parameters. Testers need to select 20 random trees for all 14 forest plots to do this test.

17. Test-FR17
    Control: Manual

Initial State: New version of software is not installed.

Input: Users go to GitHub to download the new version.

Output: A new version product should be ready to use.

Test Case Derivation: Newly released version of Digital Twin Forest will be released on GitHub. If users want to use the new version of the product with latest forest data or some new features/functions, they can go to GitHub to download.

How test will be performed: Let testers go to GitHub and download a new version. The downloaded new version of software should work properly.

5.2 Tests for Nonfunctional Requirements

5.2.1 Look and Feel Requirement testing

1. Test-NFR-LF1.1

   Type: Static.
   How test will be performed: Testers open the Module Guide document and check the Traceability Matrix to see if each function trace over 8 modules.

   Expected result: Each function trace over 8 modules.

2. Test-NFR-LF1.2

   Type: Static

   How test will be performed: Referring to the feedback of the questionnaire we provided in the Appendix, over 80 percent of the users answer the forest is lifelike.

   Expected result: Over 80 percent of the users choose A or B in the first question in the questionnaire.

3. Test-NFR-LF2.1

   Type: Static

   How test will be performed: Testers will collect all the data and parameters of our forest and compare the data to the physical measurements. After that calculate the relative error of the data

   Expected result: All the data of the forest have relative errors less than 0.05.

4. Test-NFR-LF2.2

   Type: Static

   How test will be performed: Referring to the feedback of the questionnaire we provided in the Appendix, over 80 percent of the users answer the system look professional.

   Expected result: Over 80 percent of the users choose A or B in the second question in the questionnaire.

5.2.2 Usability and Humanity Requirements testing

5. Test-NFR-UH1.1

   Type: Static

   How test will be performed: Referring to the feedback of the questionnaire we provided in the Appendix, over 80 percent of the users answer the instructions are easy.

   Expected result: Over 80 percent of the users choose A or B in the third question in the questionnaire.

6. Test-NFR-UH2.1

Type: Manual

Initial State: The system is launched

Input: Testers inspect all the texts in the software

Result: All of the texts are in English.

How test will be performed: Testers open the software and inspect all the texts in the software. 100 percent of the texts should be used in English.

7. Test-NFR-UH3.1

   Type: Manual

   Initial State: The system is launched and it is in the main page.

   Input: A click event is made on the instructions option

   Output: Instructions are displayed after the event

   How test will be performed: Testers click on the instruction option, and instructions are displayed on the screen

8. Test-NFR-UH4.1

   Type: Static

   How test will be performed: Referring to the feedback of the questionnaire we provided in the Appendix, over 80 percent of the users answer they notice and understand all the icons.

   Expected result: Over 80 percent of the users choose A or B in the fourth question in the questionnaire.

9. Test-NFR-UH4.2

   Type: Static

   How test will be performed: Referring to the feedback of the questionnaire we provided in the Appendix, over 80 percent of the users answer they think the icons used in the software are appealing.

   Expected result: Over 80 percent of the users choose A or B in the fifth question in the questionnaire.

10. Test-NFR-UH5.1

    Type: Manual

    Initial State: The system is launched

    Input: Input from mouse and keyboard from users

    Output: All actions are accomplished normally

    How test will be performed: Testers open the software and try to do all the actions in the software using a mouse and keyboard. Testers should end up finishing all the actions without any trouble.

11. Test-NFR-UH5.2

    Type: Static

    How test will be performed: Referring to the feedback of the questionnaire we provided in the Appendix, over 80 percent of the users answer they can learn how to use the system within half an hour.

    Expected result: Over 80 percent of the users choose A or B in the sixth question in the questionnaire.

5.2.3 Performance Requirements

12. Test-NFR-PR1.1

    Type: Manual

    Initial State: The system is launched

    Input: Input from mouse or keyboard to make an action in the application

    Output: The system handles all the requests within 2 seconds

    How test will be performed: Testers open the system and give input to the system. For example, the tester clicks the start button. The system should respond to any request within 2 seconds.

13. Test-NFR-PR1.2

    Type: Manual

    Initial State: The system is launched

    Input: Inputs from users to make actions

    Output: The system is running at over 30 FPS for 80 percent of the time

    How test will be performed: Testers open the system and give all kinds of inputs for actions. At the same time, testers use devices to detect the FPS of the device. The system should run at over 30 FPS for 80 percent of the time

14. Test-NFR-PR1.3

    Type: Manual

Initial State: The system is launched and the system is in main page

Input: A click event is triggered on the start option

Output: The system loads all the models within 10 seconds

How test will be performed: Testers open the system and click the start option, the testers should wait for less than 10 seconds for the models to show up.

15. Test-NFR-PR3.1

    Type: Static

    How test will be performed: Testers will record all the data and parameters in the system and compare the data to the physical measurements. After that, testers calculate the relative error of the data

    Expected result: All the data of the forest have relative errors less than 0.05.

16. Test-NFR-PR4.1

    Type: Manual

    Initial State: System is not opened

    Input: Testers give inputs to open the system

    Output:The system opens normally 100 percent of the time

    How test will be performed: Testers open the system and observe the system. the system should open normally 100 percent of the time.

17. Test-NFR-PR4.2

    Type: Manual

Initial State: The system is launched

Input: Screen recording to record the system

Output: The system is working normally 24 hours a day

How test will be performed: Testers open the system and record the screen. the system should not crash 7/11.

18. Test-NFR-PR5.1

    Type: Manual

    Initial State: The system is launched without an internet connection

    Input: Testers give inputs for different actions

    Output: The system responds to all the input actions normally

    How test will be performed: Testers open the system without an internet connection. Testers do different actions in the system and see if the system responds normally, the system should finish 100 percent of the actions without any error.

19. Test-NFR-UH6.1

    Type: Static

    How test will be performed: Testers will assess the size of the system after the system is implemented

    Expected result: The size of the software is less than 10 GB.

20. Test-NFR-PR7.1

Type: Manual

Initial State: The system is implemented

Input: Add a mini function into the system

Output: The system is modified correctly without modifying over two existing modules

How test will be performed: Developers add a mini function into the system. The system is running with a mini function normally. In the process of adding the function, no more than two modules are modified.

21. Test-NFR-PR8.1

    Type: Static

    How test will be performed: Testers will inspect all the requirements in the documents and assess whether the system achieves the requirements

    Expected result: The system achieves all the requirements in the documents

5.2.4 Operational and Environmental Requirements

22. Test-NFR-OE1.1

    Type: Manual

    Initial State: The system is opened on desktops and laptops

    Input: Inputs for different actions in the system

    Output: The system operates normally on desktops and laptops

How test will be performed: Testers will run the system on desktops and laptops. The system should run all the actions without any errors.

23. Test-NFR-OE1.2

    Type: Manual

    Initial State: The system is launched

    Input: Input from mouse and keyboard from users

    Output: All actions are accomplished normally

    How test will be performed: Testers open the software and try to do all the actions in the software using a mouse and keyboard. Testers should end up finishing all the actions without any trouble.

24. Test-NFR-OE2.1

    Type: Manual

    Initial State: The system is launched on Windows 10 and MacOS 12 or later version

    Input: Input for different actions from users

    Output: All actions are accomplished normally

    How test will be performed: Testers open the software and try to do all the actions in the software on Windows 10 or later version or MacOS 12 or later version. Testers should end up finishing all the actions without any trouble.

25. Test-NFR-OE3.1

    Type: Static

How test will be performed: Testers try to download and install the application launched on the website

Expected result: The system is downloaded and installed normally and it's able to operate

How test will be performed: Testers download and install the application on different platforms and record the operations of the application to see if they operate normally.

26. Test-NFR-OE4.1

    Type: Static

    How test will be performed: Testers will check the update logs after the software is released

    Expected result: The update is done monthly

    How test will be performed: Testers will look at the update logs and see if updates are done monthly.

27. Test-NFR-OE4.2

    Type: Static

    How test will be performed: Testers will test report of every update before releasing the new version

    Expected result: Every test case passed in the test report

    How test will be performed: Testers will look at the test report and see if all of the test cases pass.

5.2.5 Maintenance and Support Requirements Testing

28. Test-NFR-MS1.1

Type: Static.

How test will be performed: Testers check the documentation of the product to see if the new features, functions, and any modifications are added to the documentation or not.

Expected result: new features and functions and modifications should be on the documentation.

29. Test-NFR-MS1.2

    Type: Static.

    How test will be performed: Testers read the documentation of the product to see if the documentation specifies the functions clearly.

    Expected result: All functions are clearly documented.

30. Test-NFR-MS1.3

    Type: Manual

    Initial State: A bug is found in the program.

    Input: Hot fix.

    Output: The bug is fixed in three days.

    How test will be performed: Record the time from the bug occurs to the time when the bug is fixed. Measure the time to see if it is within three days.

31. Test-NFR-MS2.1

    Type: Manual

Initial State: Testers open the instruction page and try to find the contact method.

Input: Testers use the contact method to contact the developers.

Output: Testers can contact the developer and be able to give feedback to them.

How test will be performed: Testers first look for the contact method on the contact page, then use the contact method to contact the developers to see if they can contact the developers successfully or not.

32. Test-NFR-MS3.1

    Type: Manual

    Initial State: The application can run on one operating system

    Input: Testers launch the application on a different operating system.

    Output: The application can run on a different operating system.

    How test will be performed: Testers try to launch the application on more than one operating system to see if it can be run successfully on a different system or not.

33. Test-NFR-MS3.2

    Type: Manual

    Initial State: The application is running on the device located indoors

    Input: Testers run the application on the device located outdoors.

    Output: The application can run on outdoor devices.

How test will be performed: Testers try to launch the application on the devices located indoors and outdoors respectively to see if the application can be used both indoors and outdoors or not.

5.2.6 Security Requirements Testing

34. Test-NFR-SR1.1

    Type: Static.

    How test will be performed: Testers try to find and download the product from GitHub and any other website .

    Expected result: Testers can not download the product in other websites other than GitHub.

35. Test-NFR-SR1.2

    Type: static.

    How test will be performed: Testers try to update the data of trees and forests via the interface and anywhere else to see if they can update the data successfully or not.

    Expected result: The testers can only update the data from the specific interface provided by developers.

36. Test-NFR-SR2.1

    Type: Manual

    Initial State: Original data about the trees and the forests stored in the application.

    Input: New data

Output: New data about the trees and the forests will be stored in the application.

How test will be performed: Testers try to input the data via the specific interface provided by the developers and anywhere else to update the data to check if the interface is the only place where users can modify the data.

37. Test-NFR-SR2.2

    Type: Manual

    Initial State: The scanning result of the computer security application is normal.

    Input: 100 Errors injected into our software.

    Output: The scan results of the computer security application are still normal.

    How test will be performed: Testers inject 100 errors on purpose in our application and see if the scan results of the computer security application will detect the errors in the computer system or not.

38. Test-NFR-SR2.3

    Type: Manual

    Initial State: The application is running and behaving properly

    Input: The use of software for a long time and the constant interactions between the testers and the software.

    Output: The application is will running in a normal way.

    How test will be performed: Testers use the software and perform some complicated tasks such as zooming in and zooming out for a long time to see if the application will crash or not.

39. Test-NFR-SR2.4

    Type: Manual

    Initial State: The original data of the trees and forests is displayed on the interface.

    Input: Some new data about the trees and the forests.

    Output: New data about the trees and the forests is displayed on the interface.

    How test will be performed: Testers record the data that will be imported into the software. After testers import the data into the software, they check if the data about eh trees and the forests displayed on the interface matches the data that they imported.

40. Test-NFR-SR2.5

    Type: Manual

    Initial State: Data and information of the trees and forests are displayed in different windows.

    Input: Some new data about the trees and the forests.

    Output: New data about the trees and the forests are still displayed in different windows and data about trees will be displayed in the window where it should be and data about forests will be displayed in the windows where it should be.

    How test will be performed: Testers record the windows that show data about trees and forests. Then, reopen the application or update the new data. Then, record the windows that show the data to see if the data about the trees and the data about the forests shows in different windows or not and if the windows match the previous windows or not.

41. Test-NFR-SR2.6

    Type: Manual

    Initial State: The application without any information about the trees and the forests

    Input: Some new data about the trees and the forests.

    Output: The windows of the application can show all the data that was imported by the testers.

    How test will be performed: Testers launch the software that has no data of the forests and trees imported. They should see the windows are empty. Then they import the data of the forests and the trees into the product, and run the application again, they look at the windows to see if there is any data shown.

42. Test-NFR-SR3.1

    Type: Static

    How test will be performed: Referring to the feedback of the questionnaire we provided in the Appendix, all of the users answer the product did not ask them to provide their personal information.

    Expected result: All the users choose B in the ninth question in the questionnaire.

43. Test-NFR-SR3.2

    Type: Static

    How test will be performed: Referring to the feedback of the questionnaire we provided in the Appendix, all of the users answer they didn't receive notifications from the application after they turned off the notification in the application.

Expected result: All the users choose B in the seventh question in the questionnaire.

5.2.7 Cultural and Political Requirements Testing

44. Test-NFR-CP1.1

    Type: Static

    How test will be performed: Referring to the feedback of the questionnaire we provided in the Appendix, all the users answer that the contents of the product are not offensive to them.

    Expected result: 100 percent of the users choose B for the eighth question in the questionnaire.

5.2.8 Legal Requirements Testing

45. Test-NFR-LR2.1

    Type: Static.

    How test will be performed: A Canadian law expert will assess the contents like texts, images, data, and modules to check if the all contents in this product abide by Canadian laws and regulations.

    Expected result: All the contents in this product abide by all Canadian laws and regulations.

5.3 Traceability Between Test Cases and Requirements

Function Requirement	Test Case ID
FR1	Test-FR1
FR2	Test-FR23
FR3	Test-FR23
FR4	Test-FR4.1, Test-FR4.2
FR5	Test-FR5
FR6	Test-FR6.1, Test-FR6.2, Test-FR6.3
FR7	Test-FR7
FR8	Test-FR8
FR9	Test-FR9.1, Test-FR9.2
FR10	Test-FR10
FR11	Test-FR11
FR12	Test-FR12.1, Test-FR12.2, Test-FR12.3
FR13	Test-FR13.1, Test-FR13.2, Test-FR13.3
FR14	Test-FR14
FR15	Test-FR15
FR16	Test-FR16.1, Test-FR16.2, Test-FR16.3
FR17	Test-FR17

Table 2: Traceability between Functional Requirements and Tests

Non-functional Requirement	Test Case ID
NFR-LF1.1	Test-NFR-LF1.1
NFR-LF1.2	Test-NFR-LF1.2
NFR-LF2.1	Test-NFR-LF2.1
NFR-LF2.2	Test-NFR-LF2.2
NFR-UH1.1	Test-NFR-UH1.1
NFR-UH2.1	Test-NFR-UH2.1
NFR-UH3.1	Test-NFR-UH3.1
NFR-UH4.1	Test-NFR-UH4.1
NFR-UH4.2	Test-NFR-UH4.2
NFR-UH5.1	Test-NFR-UH5.1
NFR-UH5.2	Test-NFR-UH5.2
NFR-PR1.1	Test-NFR-PR1.1
NFR-PR1.2	Test-NFR-PR1.2
NFR-PR1.3	Test-NFR-PR1.3
NFR-PR3.1	Test-NFR-PR3.1
NFR-PR4.1	Test-NFR-PR4.1
NFR-PR4.2	Test-NFR-PR4.2
NFR-PR5.1	Test-NFR-PR5.1
NFR-PR6.1	Test-NFR-PR6.1
NFR-PR7.1	Test-NFR-PR7.1
NFR-PR8.1	Test-NFR-PR8.1

Table 3: Traceability between Non-Functional Requirements and Tests Part 1

Non-functional Requirement	Test Case ID
NFR-OE1.1	Test-NFR-OE1.1
NFR-OE1.2	Test-NFR-OE1.2
NFR-OE2.1	Test-NFR-OE2.1
NFR-OE3.1	Test-NFR-OE3.1
NFR-OE4.1	Test-NFR-OE4.1
NFR-OE4.2	Test-NFR-OE4.2
NFR-MS1.1	Test-NFR-MS1.1
NFR-MS1.2	Test-NFR-MS1.2
NFR-MS1.3	Test-NFR-MS1.3
NFR-MS2.1	Test-NFR-MS2.1
NFR-MS3.1	Test-NFR-MS3.1
NFR-MS3.2	Test-NFR-MS3.2
NFR-SR1.1	Test-NFR-SR1.1
NFR-SR1.2	Test-NFR-SR1.2
NFR-SR2.1	Test-NFR-SR2.1
NFR-SR2.2	Test-NFR-SR2.2
NFR-SR2.3	Test-NFR-SR2.3
NFR-SR2.4	Test-NFR-SR2.4
NFR-SR2.5	Test-NFR-SR2.5
NFR-SR2.6	Test-NFR-SR2.6
NFR-SR3.1	Test-NFR-SR3.1
NFR-SR3.2	Test-NFR-SR3.2
NFR-CP3.1	Test-NFR-CP3.1
NFR-LR3.2	Test-NFR-LR3.2

Table 4: Traceability between Non-Functional Requirements and Tests Part 2

6 Unit Test Description

This part will be finished after design document is completed

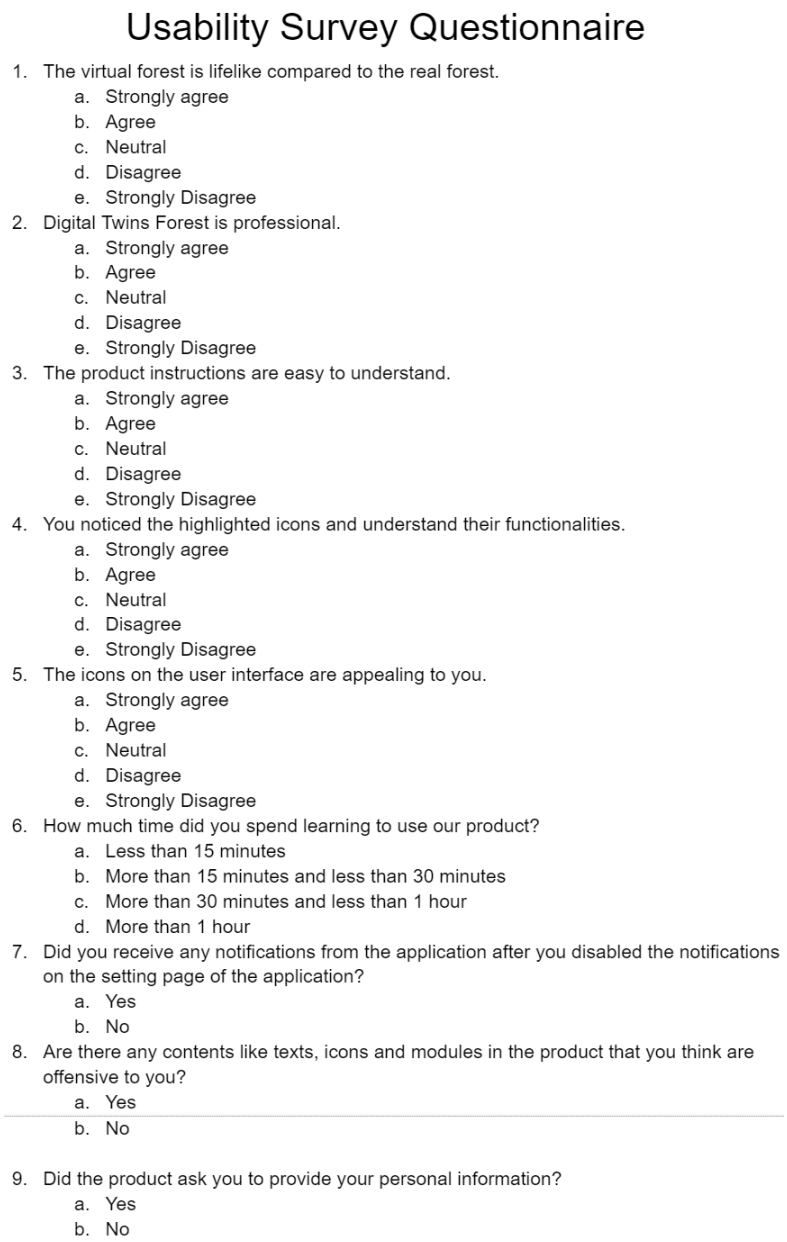
7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

N/A

7.2 Usability Survey Questionnaire



Usability Survey Questionnaire

1. The virtual forest is lifelike compared to the real forest.
    a. Strongly agree
    b. Agree
    c. Neutral
    d. Disagree
    e. Strongly Disagree
2. Digital Twins Forest is professional.
    a. Strongly agree
    b. Agree
    c. Neutral
    d. Disagree
    e. Strongly Disagree
3. The product instructions are easy to understand.
    a. Strongly agree
    b. Agree
    c. Neutral
    d. Disagree
    e. Strongly Disagree
4. You noticed the highlighted icons and understand their functionalities.
    a. Strongly agree
    b. Agree
    c. Neutral
    d. Disagree
    e. Strongly Disagree
5. The icons on the user interface are appealing to you.
    a. Strongly agree
    b. Agree
    c. Neutral
    d. Disagree
    e. Strongly Disagree
6. How much time did you spend learning to use our product?
    a. Less than 15 minutes
    b. More than 15 minutes and less than 30 minutes
    c. More than 30 minutes and less than 1 hour
    d. More than 1 hour
7. Did you receive any notifications from the application after you disabled the notifications on the setting page of the application?
    a. Yes
    b. No
8. Are there any contents like texts, icons and modules in the product that you think are offensive to you?
    a. Yes
    b. No
9. Did the product ask you to provide your personal information?
    a. Yes
    b. No

Figure 1: Usability Survey Questionnaire

7.3 Pictures

The following is a sample turn page button:

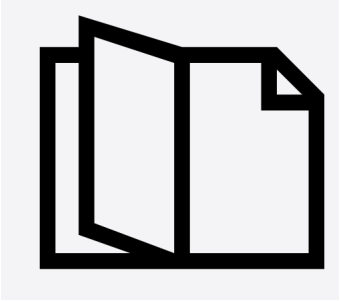


Figure 2: Turn Page Button

The following is a sample minimize button:

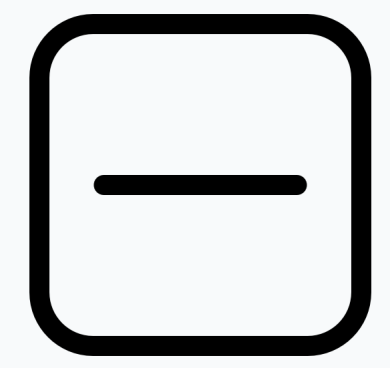


Figure 3: Minimize Button

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?

The knowledge and skills the team need to acquire to complete the verification and validation of our project are listed below:

- dynamic testing knowledge
- static testing knowledge
- automatic and manual testing knowledge
- the use of Unity
- design of questionnaire and user interview

Jiacheng is responsible for dynamic testing knowledge. The possible approaches to acquire it include online tutorials and the experience of the mandatory testing course we took last semester. He decided to pursue this knowledge because it is essential for the validation of the product, and the correctness has to be proved in the process of execution. Furthermore, he is familiar with dynamic testing knowledge.

Tingyu is responsible for static testing knowledge. Possible approaches to acquire this kind of knowledge could be the experience of a testing course we took, searching online, discussing in the group and talking to experts. Static testing is significant in the verification process, and it would include technical or informal review, inspection, walkthrough, static code review, etc. As Tingyu is also responsible for organizing

regular meetings and taking the script, he would love to study static testing knowledge and put static testing into our meetings as a part of the discussion. He would control and manage the process of static testing.

Yichen is responsible for automatic and manual testing knowledge. Knowledge about automatic and manual testing can be gained from the online materials, experience in the testing course, and more. The knowledge of automatic testing will mainly apply to parametric modelling and data storage. And manual testing will be used on the other part, as our product relies on the user interface, and we have to check the overall results manually. Yichen decided to pursue this kind of knowledge because she will take care of the overall effect of the user interface and make modifications along with the testing results.

Bowen is responsible for the use of Unity. The possible methods of gaining knowledge of Unity include online tutorials, experience from past courses, and talking to experts. Bowen will work as the main developer for the modelling part and has one year of CO-OP experience working on Unity, and that is why he will pursue this skill.

Junhong is responsible for the design of the questionnaire and user interview. The possible approaches to gaining this skill include learning in human-computer interaction class, searching online, and learning about successful cases we can find. This skill is essential because we would love to know how the user feels using our product to get a full view of the usability and user experience. Moreover, this can only be achieved by asking our users. Junhong decided to pursue this one because he already deeply understands this skill and has performed many independent projects with questionnaires.