Project Title: System Verification and Validation Plan for Software Engineering

Team 17, Track a Trace
Zabrain Ali
Linqi Jiang
Jasper Leung
Mike Li
Mengtong Shi
Hongzhao Tan

November 2, 2022

1 Revision History

Date	Version	Notes
November 1, 2022	1.0	Modified Abbreviations and Acronyms, General Information, Plan, System Test Description and the Appendix
November 2, 2022	1.1	Modified Plan and System Test Description

Contents

List of Tables

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
SRS	Software Requirement Specification
MG	Module Guide
MIS	Module Interface Specification
GPS	Global Positioning System
GIS	Geographic Information System
GERT	GIS-based Episode Reconstruction Toolkit
ArcGIS	A licensed GIS service used by GERT for data processing
PyERT	Python-based Episode Reconstruction Toolkit
CSV	Comma Separated Values
SHP	shapefile: A data format for spreadsheet
RCA	Route Choice Analysis
LU	Land Use
PAL	Potential Activity Locations
NFR	Non-Functional Requirement

This document will provide a detailed plan for verification and validation for PyERT. The following outline gives an overview of what is covered in this document:

- General Information is described at 3
- Plan is described at 4
- System Test Description is provided at 5
- Unit Test Description is provided at 6

3 General Information

3.1 Summary

The software being tested is called PyERT. PyERT is a software toolkit that aims to reverse engineer the GERT toolkit. PyERT is intended to re-implement the features in GERT that use ArcGIS Pro packages with open-source packages and libraries to make it fully open-source and independent from proprietary software like ArcGIS Pro. Some general functionalities of PyERT include pre-processing GPS points, classifying GPS points segments, generating alternative routes on transportation networks for given GPS points and adding information about activity locations.

3.2 Objectives

The objectives of verification and validation are to build confidence for the team in the correctness of our software, determine if the requirements are built correctly and that the design and development are following the requirements. Another goal is to demonstrate adequate usability of our product, showing that team is building what the clients truly needed and require.

3.3 Relevant Documentation

The documents that are relevant to verification and validation include the SRS (Ali et al., 2022), MG and MIS for the project.

4 Plan

This section will list the team members involved in the verification and validation process of the project and will describe the different verification and validation plans of the project. The following outline gives an overview of what is covered in this section:

- Verification and Validation Team is described at 4.1

- SRS Verification Plan is described at 4.2
- Design Verification Plan is described at 4.3
- Verification and Validation Plan Verification Plan is described at 4.4
- Implementation Verification Plan is described at 4.5
- Automated Testing and Verification Tools are described at 4.6
- Software Validation Plan is described at 4.7

4.1 Verification and Validation Team

1. Jasper Leung: Will focus on non-functional testing, and will lead the code review session.
2. Mike Li: Will focus on non-functional testing, and will handle the gathering of test users and distribution of the usability survey.
3. Zabrain Ali: Will focus on testing the extraction of activity locations, generation of alternative routes, and activity locations with additional information.
4. Hongzhao Tan: Will focus on functional testing, specifically the removal of redundant and outlier GPS data and classification of GPS points and segments.
5. Mengtong Shi: Will focus on functional testing, specifically the generation of CSV and SHP files, the generation of error messages, and the assignment of values to RCA variables; and will continuously focus on the SRS verification and code verification in the future.
6. Linqi Jiang: Will focus on functional testing, specifically the extraction and generation of activity locations.
7. Antonio Paez (Supervisor): Will help the team verify the design and VnV plan once the system is completed.

4.2 SRS Verification Plan

- Once the system is complete, a test group of users will be given the system and a set of sample inputs for the system. They will be asked to use one of the sample inputs to generate an output with the system. They will be asked to use the system on two devices. After they do this, they will be given a usability survey. This usability survey's questions will be based on some non-functional tests listed in section 5.2.
- For some non-functional tests, the developers will run the program with specific conditions and observe manually whether the output is correct.

4.3 Design Verification Plan

- Classmates will review the design documents of the system.
- The team will hold a review session with Dr.Paez when the system has been completed. During the review session with Dr.Paez, he will go over the design documents with the team and verify that everything described in the design was implemented properly.

4.4 Verification and Validation Plan Verification Plan

- Classmates will review the Verification and Validation Plan of the system.
- The team will keep track of all the tests listed in the document, and during testing, make sure that all of them are completed successfully.
- The team will hold a review session with Dr.Paez when the system has been completed. During the review session with Dr.Paez, he will go over the Verification and Validation document with the team and verify with the team that all the tests described in the VnV plan were completed successfully.

4.5 Implementation Verification Plan

Once the system is complete, the developers will hold a code review session and look for any errors. This session will be used for the static non-functional tests listed in section 5.2 (SR1, SR2, LR1). Any inconsistencies, errors, or non-functional code should be removed.

4.6 Automated Testing and Verification Tools

The unit testing framework, linter and coding standard the team will follow have been outlined in the Development Plan under sections 6 and 7.

4.7 Software Validation Plan

The team will hold a review session with Dr.Paez to verify that the requirements document has the right requirements. The team and Dr.Paez will walk through the SRS and review every requirement.

5 System Test Description

5.1 Tests for Functional Requirements

This section contains the tests for the Functional Requirements. The subsections for these tests were created based on the subsections of the Functional Requirements listed in the SRS (Ali et al., 2022). Each test was created according to the Fit Criterion of the requirements they were covering. Traceability for these requirements and tests can be found in the traceability matrix (5.3).

5.1.1 Generation of CSV and SHP Files

This subsection covers Requirement #1 of the SRS document (Ali et al., 2022), by testing that the system is able to read and generate CSV and SHP files.

1. test-FR1-1

   Control: Manual

   Initial State: The system is set up and ready to take over the user's input.

   Input: Valid GPS data

   Output: CSV and SHP files containing alternative routes, RCA variables with values and activity locations information

   Test Case Derivation: The data will be processed by the script is previously determined to be valid and should not cause the script running into any error. The alternative routes and activity location information output does not have to be correct.

   How this test will be performed: The test controller will be given a sample GPS data that has been made sure is valid. The controller will execute the script with the sample data as input and observe if the script can successfully generate CSV and SHP files or run into an error.

5.1.2 Removal of Invalid Points

This subsection covers Requirement #2 of the SRS document (Ali et al., 2022), by testing that the system is able to remove invalid GPS points from the user's input.

1. test-FR2-1

   Control: Manual

   Initial State: The system is set up and ready to take over the user's input.

   Input: GPS points where all points are valid

   Output: Valid GPS points without redundant points(points with the same coordinate) and outliers(speed $\geq$ 50m/s)

   Test Case Derivation: Because the sample data has been predetermined to have neither redundant point nor outlier. The system should not remove any points from the input GPS data.

   How this test will be performed: The test controller will execute the script with the input sample data which will be a CSV file containing GPS points, and the controller will observe from the generated output to see if any input point has been removed.

2. test-FR2-2

   Control: Manual

   Initial State: The system is set up and ready to take over the user's input.

Input: GPS points with redundant points, outliers and regular valid points

Output: Valid GPS points without redundant points(points with the same coordinate) and outliers(speed $\geq$ 50m/s)

Test Case Derivation: Because the sample data has been predetermined to have both redundant points and outliers. The system should remove the redundant points and outliers from the input GPS data.

How this test will be performed: The test controller will execute the script with the input sample data which will be a CSV file containing GPS points, and the controller will observe from the generated output to see if the predetermined redundant points and outliers have been removed.

3. test-FR2-3 Control: Manual

   Initial State: The system is set up and ready to take over the user's input.

   Input: GPS points with only redundant points

   Output: Valid GPS points without redundant points(points with the same coordinate) and outliers(speed $\geq$ 50m/s)

   Test Case Derivation:Because the sample data has been predetermined to have only redundant points. The system should remove all the points given in the input GPS data.

   How this test will be performed: The test controller will execute the script with the input sample data which will be a CSV file containing GPS points, and the controller will observe from the generated output to see if all the points given in the input data have been removed.

4. test-FR2-4

   Control: Manual

   Initial State: The system is set up and ready to take over the user's input.

   Input: GPS points with only outliers

   Output: Valid GPS points without redundant points(points with the same coordinate) and outliers(speed $\geq$ 50m/s)

   Test Case Derivation: Because the sample data has been predetermined to have only outliers. The system should remove all the points given in the input GPS data.

   How this test will be performed: The test controller will execute the script with the input sample data which will be a CSV file containing GPS points, and the controller will observe from the generated output to see if all the points given in the input data have been removed.

5.1.3 Division of GPS Points

This subsection covers Requirement #3 of the SRS document (Ali et al., 2022), by testing that the system is able to divide GPS points into 24-hour trajectories and further divide

those trajectories into clusters of adjacent points based on speed, distance, heading and change-in-heading thresholds.

1. test-FR3-1

   Control: Manual

   Initial State: The system is set up and ready to take the inputs.

   Input: Valid GPS points

   Output: GPS points with 24-hour trajectories, which are divided into clusters of adjacent points based on speed, distance, heading, and change in-heading thresholds

   Test Case Derivation: The system will first divide GPS points into 24-hour trajectories and then divide them into clusters of adjacent points based on speed, distance, heading, and change in-heading thresholds with input with valid GPS data.

   How this test will be performed: The test controller will be given a sample GPS data that has been made sure is valid, the controller will execute GPS points after proceeding with data.

5.1.4 Tag Creation for Valid GPS Points

This subsection covers Requirement #4 of the SRS document (Ali et al., 2022), by testing that the system is able to tag each valid input GPS point as a stationary (stop) point or a trip (moving) point.

1. test-FR4-1

   Control: Manual

   Initial State: The system has received valid GPS data and is ready to proceed.

   Input: Valid GPS data

   Output: GPS points are tagged with stop points or trip points correctly.

   Test Case Derivation: The system will tag GPS points with stop points and trip points with input with valid GPS data.

   How this test will be performed: The test controller will be given a sample GPS data that has been made sure is valid and predetermined tags on each of the GPS data points. The controller will execute the script with the sample data as input and observe if GPS points tagged by the script matches the predetermined tags.

5.1.5 Partition of GPS Trajectories

This subsection covers part of Requirement #5 of the SRS document (Ali et al., 2022), by testing that the system is able to partition valid GPS trajectories into segments.

1. test-FR5-1

   Control: Manual

   Initial State: System has divided the GPS points into trajectories and ready to proceed.

   Input: GPS trajectories

   Output: GPS Points Segments

   Test Case Derivation: The system will take over GPS trajectories and partition them into segments.

   How this test will be performed: The valid GPS points will proceed to the control, and the control will divide them into trajectories, and then they will be partitioned into segments by control.

5.1.6 Extraction of Trip Segments

This subsection covers Requirement #6 of the SRS document (Ali et al., 2022), by testing that the system is able to extract trip segments from valid GPS trajectories.

1. test-FR6-1

   Control: Manual

   Initial State: System has classified the GPS points segments partitioned from trajectories.

   Input: GPS trajectories with classified GPS points segments

   Output: Extract the trip segments(sequences of GPS points in travel episodes)

   Test Case Derivation: The system will need to look at the GPS trajectories and extract the trip segments because it is necessary when for generating alternative routes for the input GPS points.

   How this test will be performed: The test controller will input GPS trajectories and check if the system is able to extract trip segments from the data and then verify if the trip segments match the contents of a list of correct predetermined trip segments from the same GPS data.

5.1.7 Extraction of Activity Locations

This subsection covers Requirement #7 of the SRS document (Ali et al., 2022), by testing that the system is able to extract activity locations from valid GPS trajectories.

1. test-FR7-1

   Control: Manual

Initial State: System has classified the GPS points segments partitioned from trajectories.

Input: GPS Trajectories

Output: Extraction of activity locations (GPS points in stop episode or end points of trip segments)

Test Case Derivation: The system will need to look at the GPS trajectories and extract the activity locations because it is necessary when adding LU and PAL information to stationary GPS points.

How this test will be performed: The test controller will input GPS trajectories and check if the system is able to get activity locations from the data and then verify if the activity locations match the contents of a list of correct predetermined activity locations from the same GPS data.

5.1.8 Generation of Alternative Routes

This subsection covers part of Requirement #8 of the SRS document (Ali et al., 2022), by testing that the system is able to generate alternative routes for trip segments in SHP format with the correct digital road/pedestrian network.

1. test-FR8-1

   Control: Manual

   Initial State: System has classified the GPS points segments partitioned from trajectories.

   Input: Valid GPS Points

   Output: Alternative routes for trip segments in SHP format with digital road/pedestrian network

   Test Case Derivation: Because the system is developed to match input GPS points into transportation network. With the previously extracted trip segments, the system should be able to generate correct alternative routes on the transportation network for these segments.

   How this test will be performed: The test controller will use valid GPS data in a correct file format to test that an SHP file with digital road/pedestrian network is produced and matches the content of a predetermined SHP file with the correct digital road/pedestrian network.

5.1.9 Generation of Activity Locations with Additional Information

This subsection covers Requirement #9 of the SRS document (Ali et al., 2022), by testing that the system is able to generate SHP files for extracted activity locations with additional information given spatial data such as LU and PAL data in the user's input.

1. test-FR9-1

   Control: Manual

   Initial State: System has classified the GPS points segments partitioned from trajectories.

   Input: LU and PAL data

   Output: SHP files for extracted activity locations with additional information

   Test Case Derivation: The system will recognize the spatial data such as land use and potential activity locations using overlay analysis functions in GIS.

   How this test will be performed: The test controller will use specific spatial data to generate SHP files that matches a predetermined SHP file with the correct activity locations with additional information.

5.1.10 Generation of Error Messages

This subsection covers Requirement #10 of the SRS document (Ali et al., 2022), by testing that the system is able to generate descriptive messages to the user.

1. test-FR10-1

   Control: Manual

   Initial State: The system is set up and ready to take over user's input.

   Input: Invalid data format

   Output: Descriptive error message

   Test Case Derivation: The system will recognize that the data format is invalid and prompt an error message explaining why the system has an error.

   How this test will be performed: The test controller will try to use a file with an invalid data format.

2. test-FR10-2

   Control: Manual

   Initial State: The system is set up and ready to take over user's input.

   Input: Invalid file format

   Output: Descriptive error message

   Test Case Derivation: The system will recognize that the file is invalid and prompt an error message explaining why the system has an error.

   How this test will be performed: The test controller will try to use an invalid file format.

5.1.11 Classification of GPS Points Segments

This this subsection covers part of Requirement #11 of the SRS document (Ali et al., 2022), by testing that the system is able to classify the segments into stationary activity (stop) episodes and travel (moving) episodes.

1. test-FR11-1

   Control: Manual

   Initial State: System has partitioned trajectories into segments and ready to proceed.

   Input: Segments that are derived from GPS trajectories

   Output: Segments that are classified with stationary activity (stop) episodes or travel (moving) episodes

   Test Case Derivation: The system will take over Segments that are derived from GPS trajectories and classify them into stationary activity (stop) episodes and travel (moving) episodes.

   How this test will be performed: After trajectories are partitioned, the system will classify them into episodes. The test controller will check with predetermined classified GPS points and segments that the output of the system is correct.

5.1.12 Assignment of Values to RCA Variables

This subsection covers part of Requirement #12 of the SRS document (Ali et al., 2022), by testing that the system is able to assign values to RCA variables in CSV format for each of the alternative routes generated based on the provided road network attributes.

1. test-FR12-1

   Control: Manual

   Initial State: System has already generated alternative routes for trip segments.

   Input: Valid GPS Points

   Output: CSV formatted file with the correct RCA variables and values

   Test Case Derivation: Because the system has generated alternative routes for the input GPS data. It should be also to generate the correct RCA variables with correct values as the system is developed to do so.

   How this test will be performed: The test controller will use valid GPS points to generate an alternative route for trip segments. This will then have a CSV file produced and the test controller will verify with a predetermined CSV file with the correct RCA values that the newly generated CSV file is correct.

5.2 Tests for Nonfunctional Requirements

This section contains tests for Nonfunctional Requirements. The subsections for these tests were created based on the subsections of the Nonfunctional Requirements listed in the SRS (Ali et al., 2022). Each test was created according to the Fit Criterion of the requirements they were covering. The traceability for these requirements and tests can be found in the traceability matrix (5.3).

5.2.1 Look and Feel

This subsection's tests covers all Look and Feel Requirements listed in the SRS (Ali et al., 2022), by testing that it is easy for users to figure out the input for the system and follow the system.

1. LF1

   Type: Non-Functional, Dynamic, Manual

   Initial State: The system is installed and run.

   Input/Condition: The user will be asked if the system's interface is easy to follow.

   Output/Result: At least 90% of the test users for the system will say that the system's interface is easy to follow

   How this test will be performed: A test group of users will be given the system and a set of sample inputs for the system. They will be asked to use one of the sample inputs to generate an output with the system. After they do this, they will be given a usability survey (see section 7.2). In the usability survey, the users will be asked if the system's interface is easy to follow.

2. LF2

   Type: Non-Functional, Dynamic, Manual

   Initial State: The system is installed and run.

   Input/Condition: The user will be asked if the system's interface makes it easy to determine the required input.

   Output/Result: At least 90% of the test users for the system will say that the system's interface makes it easy to determine the input.

   How this test will be performed: A test group of users will be given the system, and a set of sample inputs for the system. They will be asked to use one of the sample inputs to generate an output with the system. After they do this, they will be given a usability survey. In the usability survey, the users will be asked if the system's interface makes it easy to determine the required input.

3. LF3

   Type: Non-Functional, Dynamic, Manual

   Initial State: The system is installed and run.

   Input/Condition: The user will be asked if the system's interface gave a clear indication of what stage the system was in while the system was generating an output

   Output/Result: At least 90% of the test users for the system will say that they were able to tell what stage the system was in from the system's interface while the system was generating output.

   How this test will be performed: A test group of users will be given the system, and a set of sample inputs for the system. They will be asked to use one of the sample inputs to generate an output with the system. After they do this, they will be given a usability survey. In the usability survey, the users will be asked if the system's interface gives a clear indication of the stage of the system.

5.2.2 Usability and Humanity

This subsection's tests covers all Usability and Humanity requirements listed in the SRS (Ali et al., 2022), by testing that it is easy for users to download the system, testing if the users can generate output, and testing that the system can display instructions.

1. UH1

   Type: Non-Functional, Dynamic, Manual

   Initial State: The system is available for download online. The user installs the system.

   Input/Condition: The user will be asked their programming experience level, and if they were able to install the system without asking for help.

   Output: At least 90% of the users with little to no programming experience will say that they were able to install the system without asking for help.

   How this test will be performed: A test group of users will be given the system, and a set of sample inputs for the system. They will be asked to use one of the sample inputs to generate an output with the system. After they do this, they will be given a usability survey. The user will be asked for their programming experience level. They will also be asked if they needed to ask for help while downloading the system.

2. UH2

   Type: Non-Functional, Dynamic, Manual

   Initial State: The system is installed and run.

   Input/Condition: The user will be asked if they were able to generate an output with given sample inputs.

Output/Result: At least 95% of the test users for the system will say that they were able to generate an output after sending in a sample input.

How this test will be performed: A test group of users will be given the system, and a set of sample inputs for the system. They will be asked to use one of the sample inputs to generate an output with the system. After they do this, they will be given a usability survey. In the usability survey, the users will be asked if they were able to generate an output with the given sample inputs.

3. UH3

Type: Non-Functional, Dynamic, Manual

Initial State: The system is set up and ready to take the user's input.

Input/Condition: The developers will run the program with a help flag in the input.

Output/Result: There will be a help menu displayed on the system's interface

How this test will be performed: The developers will run the program with a help flag in the input, and confirm that there is a help menu displayed when it is ran with that flag.

5.2.3 Performance

This subsection's tests covers all Performance requirements listed in the SRS (Ali et al., 2022), by testing the processing time, the data size limit of the system input, and that the system can be used at any date/time.

1. PR1

Type: Non-Functional, Dynamic, Manual

Initial State: The system is set up and ready to take the user's input.

Input/Condition: The system is called with a CSV data file with 1,000,000 GPS points.

Output: The system returns on output in at most 88 seconds.

How this test will be performed: The developers will run the system and use a data file containing 1,000,000 GPS points as the input. The difference between the start and end time of processing the data will be output. If the resulting difference is 88 seconds or less, this test will be considered a success.

2. PR2

Type: Non-Functional, Dynamic, Manual

Initial State: The system is set up and ready to take the user's input.

Input/Condition: The system is called with a data file of 50 million GPS points.

Output: The system successfully generates an output for an input data set containing 50 million data points.

How this test will be performed: The developers will run the system and use a data file containing 50 million data points as the input. They will verify manually that an output is generated, and that the output is valid.

3. PR3

Type: Non-Functional, Dynamic, Manual

Initial State: The system is set up and ready to take the user's input.

Input/Condition: The system is ran with different date and time settings.

Output: The system generates the same output when ran at 2 different dates/times.

How this test will be performed: The developers will run the system using a sample input. They will then close the system, change the date and time of the computer system, re-open the system, and run it again with the same sample input. They will verify manually that the system has generated the same output.

5.2.4 Operational and Environmental

This subsection's tests covers all Operational and Environmental requirements listed in the SRS (Ali et al., 2022), by testing that the system can be used with any operating system that has Python, doesn't install external packages, and properly lists all required packages.

1. OE1

Type: Non-Functional, Dynamic, Manual

Initial State: The system is installed and run.

Input/Condition: The user will be asked for their operating system and if they were able to run the system without issue.

Output: All users using Windows and Linux will report that they are able to run the system without any errors.

How this test will be performed: A test group of users will be given the system, and a set of sample inputs for the system. They will be asked to use one of the sample inputs to generate an output with the system. After they do this, they will be given a usability survey. The user will be asked for their operating system, and if they ran into any issues with the program.

2. OE2

   Type: Non-Functional, Dynamic, Manual

   Initial State: The system is downloaded in a virtual environment with the required packages installed.

   Input/Condition: The system will be run in the virtual environment without downloading any external software.

   Output: The system returns a valid output.

   How this test will be performed: The developers will create a virtual environment with the required packages of PyERT and run the system. They will manually verify that a output is returned without any other external software needed.

3. OE3

   Type: Non-Functional, Dynamic, Manual

   Initial State: The system is downloaded in a virtual environment.

   Input/Condition: The system will be run in the virtual environment after the user installs the system's dependent packages given by the system.

   Output: The system returns a valid output.

   How this test will be performed: The developers will create a virtual environment and download the system. They will manually verify that the system contains a dependency list of all the required packages and install them. The developers will run the system after the packages are installed and verify that a valid output is returned.

5.2.5 Maintainability and Support

This subsection's tests covers the Maintainability and Support Requirement 3 (MS3) listed in the SRS (Ali et al., 2022), by testing if the system is portable. Requirements MS1 and MS2 are not covered by tests, since they are Git requirements and our group felt that tests would not be appropriate.

1. MS1

   Type: Non-Functional, Dynamic, Manual

   Initial State: The system is set up and ready to take the user's input.

   Input/Condition: The user will be asked to run the system on two separate devices.

   Output: At least 90% of the users will report in the usability survey that using the system behaved the same and was similar on both devices.

   How this test will be performed: A test group of users will be given the system, and a set of sample inputs for the system. They will be asked to use one of the sample

inputs to generate an output with the system. They will be asked to use the system on two devices. After they do this, they will be given a usability survey. The user will be asked for their experience with the system on both devices.

5.2.6 Security

This subsection's tests covers Security Requirements 1 and 2 listed in the SRS (Ali et al., 2022), by testing that the system does not access the user's information or external files. Requirements SR3 was not covered by tests, since it is a Git requirement and our group felt that a test would not be appropriate.

1. SR1

   Type: Non-Functional, Static, Manual

   Initial State: An initial working copy of the code for the system is completed.

   Input/Condition: The developers will be given the system's code to review.

   Output: The developers will confirm that there is nothing in the system that accesses a user's personal information.

   How this test will be performed: Once the system is complete, the developers will hold a code review session. The developers will confirm that there is no code that accesses the user's personal information.

2. SR2

   Type: Non-Functional, Static, Manual

   Initial State: An initial working copy of the code for the system is completed.

   Input/Condition: The developers will be given the system's code to review.

   Output: The developers will confirm that the code does not access files outside the user-provided inputs and files included in Python and the system.

   How this test will be performed: Once the system is complete, the developers will hold a code review session. The developers will confirm that there is no code that accesses any external files.

5.2.7 Legal

This subsection's tests covers the Legal Requirement listed in the SRS (Ali et al., 2022), by testing that all packages used to not require licenses.

1. LR1

   Type: Non-Functional, Static, Manual

Initial State: An initial working copy of the code for the system is completed.

Input/Condition: The developers will be given the system's code to review.

Output: The developers will confirm that the code does not use packages or libraries that require licenses.

How this test will be performed: Once the system is complete, the developers will hold a code review session. They will confirm that there are no packages or libraries used in the code that require licenses.

5.3 Traceability Between Test Cases and Requirements

Test ID	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
test-FR1-1	×											
test-FR2-1		×										
test-FR2-2		×										
test-FR2-3		×										
test-FR2-4		×										
test-FR3-1			×									
test-FR4-1				×								
test-FR5-1					×							
test-FR6-1						×						
test-FR7-1							×					
test-FR8-1								×				
test-FR9-1									×			
test-FR10-1										×		
test-FR10-2										×		
test-FR11-1											×	
test-FR12-1												×

Table 1: **Functional Requirements Traceability**

NFR ID	LF1	LF2	LF3	UH1	UH2	UH3	PR1	PR2	PR3	OE1	OE2	OE3
LF1	×	×										
LF2			×									
UH1				×								
UH2					×							
UH3						×						
PR1							×					
PR2								×				
PR3									×			
OE1										×		
OE2											×	
OE3												×

Table 2: **Non-Functional Requirements Traceability Part 1**

NFR ID	MS1	SR1	SR2	LR1
MS1				
MS2				
MS3	×			
SR1		×		
SR2			×	
SR3				
LR1				×

Table 3: **Non-Functional Requirements Traceability Part 2**

6 Unit Test Description

6.1 Unit Testing Scope

This section cannot be completed yet. It will be completed after the Module Guide and Module Interface Specification are created.

6.2 Tests for Functional Requirements

N/A

6.3 Tests for Nonfunctional Requirements

N/A

6.4 FR1-1n Test Cases and Modules

N/A

References

Zabrain Ali, Linqi Jiang, Jasper Leung, Mike Li, Mengtong Shi, and Hongzhao Tan. Software requirements specification for software engineering: Pyert, 2022. URL https://github.com/paezha/PyERT-BLACK/blob/main/docs/SRS/SRS.pdf.

7 Appendix

7.1 Symbolic Parameters

N/A

7.2 Usability Survey Questions?

The following questions listed below would be delivered on Google Forms and presented to the volunteer testers when testing the game. They consist of multiple choice followed by some written questions.

1. How experienced are you with programming?
    (a) Little to no experience
    (b) Moderately experienced
    (c) Very experienced
2. What operating system do you use?
    (a) Windows
    (b) Mac
    (c) Linux
3. Were you able to install the system?
    (a) Yes
    (b) No
4. Was the system's interface easy to follow and understand?
    (a) Yes
    (b) No
5. Did the system's interface make it easy to determine the correct input type?
    (a) Yes
    (b) No
6. While it was generating an output, did the system make it easy to determine the current stage of the system?
    (a) Yes
    (b) No
7. Were you able to generate an output with one of the given sample inputs?

(a) Yes

(b) No

8. If you used the system while on a separate device, was your experience with the system different on both devices? Write the differences below or leave this section blank if there were none.

(a) Yes

(b) No

9. Did you run into any issues/errors while running the system? Write them below if you did, leave this blank if you didn't.

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?

Knowledge and Skills

- To perform unit testing, the team will need to acquire skills with Pytest.
- The team will need to acquire dynamic testing knowledge for testing the requirements properly.
- The team will need to acquire GiHub Actions' usage to run automatic tests during CI/CD.
- To acquire static testing knowledge, the team will need to learn how to effectively hold a code review sessions and develop an effective method of verifying that requirements are met in the code.
- The team will need to acquire domain specific knowledge in order to fully understand the inputs and outputs of the functional tests.
- The team will need to learn how to set up the proper environment for the system and test controller to correctly perform functional and non-functional tests.

Approaches

- Pytest
  - Research online on how to use Pytest.
  - Take online courses related to Pytest.
  - Create Pytest projects in our own time.

  Zabrain - The approach I will choose for acquiring knowledge on how to create tests with Pytest will be to create Pytest projects in my own time. This is the approach I will select because the best form of learning how to use different testing frameworks is by trying to create tests with them. The knowledge and experience I will gain by

practicing writing tests using Pytest will greatly benefit the performance of our system and will allow the group to have a better understanding on how to successfully create good functional tests for this project.

- Dynamic testing
  - Research online on the best method to perform dynamic testing.
  - Consult the professor and ask questions about dynamic testing.

  Mengtong - The approach I will choose for acquiring dynamic testing knowledge will be researching online for good dynamic testing techniques. This is the approach I will select because there are a lot of in-depth resources about dynamic testing techniques online, and doing some research to find the best-fit method to perform dynamic testing will help the group in the upcoming testing process.

- GitHub Actions
  - Research online on how to use GitHub Actions.
  - Re-watch the tutorial for the 4G06 course on how to use GitHub Actions.

  Hongzhao - This is a good approach since the tutorial of the course can teach about how to build automatic test using GitHub Actions with self-explained examples presented by the instructors and the examples can be followed easily, and the online resources about GitHub Actions could help on some use cases of Git that the tutorial has not been covered but needed by the project. Building automatic tests with GitHub Actions can let the implementation be tested each time when submitting new changes to the repository to make sure that the new changes do not lead to bugs for the system as a whole.

- Effectively holding code review sessions
  - Research online on how to effectively hold code review sessions.
  - Consult with our supervisor, Dr. Antonio Paez, and gather information on what our team should be specifically looking for during our code review session.

  Mike - The approach I will choose will be to research online on how to effectively hold code review sessions. This is a good approach as searching online for tips on how to hold code review sessions will reveal many tips and in-depth resources for code review sessions that we previously did not know. Making our code review sessions more efficient will allow us to constantly review our code and ensure the code base quality is up to high standard at a faster pace.

- Domain-specific knowledge:
  - Consult with our supervisor, Dr.Antonio Paez.
  - Do research online on technologies similar to GERT.
  - Read the in-depth documentation provided for GERT.

Jasper - The approach I will choose for acquiring domain-specific knowledge is reading the in-depth documentation provided for GERT. This is the approach I will select because reading the code documentation help out group understand the exact inputs and outputs of the system. This knowledge will help greatly when performing testing. It will also contain information that cannot be found from observation or from discussion with the supervisor.

- Set up the proper environment
  - Consult with our supervisor, Dr. Antonio Paez.
  - Research online to set up a proper method to setup a testing environment.
  - Read and understand documentation of GERT.

  Linqi Jiang - The approach I will choose to set up a proper environment for testing is to read and understand the documentation provided within GERT. This is the most effective way for this approach since the documentation will contain knowledge and instructions for team members to build the proper testing environment for requirements. Building proper is very important for the testing process since different testing environment may lead different unexpected errors to the final results, by understanding documentation within the GERT, the process for building the testing environment for team will be easier and much more precise.