Test Report: yoGERT Toolbox

Team 19: Abeer Al-Yasiri, Nicholas Lobo,
Niyatha Rangarajan, Smita Singh,
Moksha Srinivasan, Longwei Ye

March 9, 2023

1 Revision History

Date	Version	Notes
Date 2023-03-08	1.0	Abeer: Section 3.3-3.4,3.11-3.12 ,Section 4, Section 6.1-6.5, Section 8.1, Section 9, Section 11
Date 2023-03-08	1.0	Longwei: Section 3.18, Section 8.2-8.4
Date 2023-03-08	1.0	Moksha: Sections 3.1-2, 3.17, 5, 6.8, 6.5, 8.1, 11
Date 2023-03-08	1.0	Nicholas: Sections 3.5-7, 3.9-10,3.15 , 6.9, 7.5, 8.1
Date 2023-03-08	1.0	Smita: Sections 3, 6.7,6.10,6.11
Date 2023-03-08	1.0	Niyatha: Sections 3, 6.6, 6.9, 8
Date 3	1.2	Notes

2 Symbols, Abbreviations and Acronyms

term	description
Activity Locations (ALs)	ALs are trip stops
ArcGIS	an online geographic information system software that is command line based system for manipulating and visualization data.
CSV/.csv	Comma Separated Values is a file type that contains large amounts of data separated by commas.
Episode	Session
GERT	GIS-based episode reconstruction toolkit
GIS	Geographical Information Systems
GPS	Global Positioning Systems
Mode Detection (MD)	Detection of transportation type being used
Route	Object path to get from position A to position B
Trace	Collection of GPS points for one travel behaviour/one user. Each trace is not connected to another.
REV1	Revision 1

symbols, abbreviations or acronyms – you can reference the SRS tables if needed

Contents

List of Tables

List of Figures

This document describes the results from completing the tests and evaluations described in the VnV plan. Some of the tests and evaluations were not completed by this revision but are on track to be completed for the next revision.

3 Functional Requirements Evaluation

3.1 R1

3.1.1 Transformation Module

The system successfully uploads a csv format file and notifies the user as seen in the Transformation module which creates a standardized Point object for map functions.

3.1.2 PreProcessing Module

The system allows users to upload various formats of GPS data in CSV file formats.

3.2 R2

3.2.1 PreProcessing Module

The system successfully processes GPS data points of multiple user traces.

3.3 R3

3.3.1 Mapping Module

The requirement is satisfied as the module creates a transferable HTML file format for interactive maps to be displayed on.

3.4 R4

3.4.1 Network Module

The requirement is satisfied as it handles GPS coordinates as an abstract data type and accepts GPS coordinate as inputs of type tuple.

3.4.2 ShortestRouteTrace Module

The requirement is satisfied as it handles GPS coordinates as an abstract data type.

3.4.3 ShortestRouteEpisode Module

The requirement is satisfied as it handles GPS coordinates as an abstract data type

3.4.4 Alternative Module

The requirement is satisfied as it handles GPS coordinates as an abstract data type

3.5 R5

3.5.1 Episode Generation Module

The requirement is satisfied as it stores the speed duration distance and status points for each trace

3.6 R6

3.6.1 Episode Generation Module

The requirement is satisfied as it extracts the trace into different types of episode which include stop, drive, and walk

3.7 R7

3.7.1 Episode Generation Module

The requirement is satisfied as it decomposes the episode trace into segments of type stop and trip

3.8 R8

3.8.1 Transformation Module

The system successfully identifies available activity locations given the csv from fetchactivitylocations function.

3.8.2 Fetch Activity Location Module

The system successfully fetches available activity locations for stop points given to the module

3.8.3 Point Module

Point Module is used to fetch the activity location from api

3.8.4 Activity Location Module

The system successfully create Activity Location Object with specified parameters

3.9 R9

3.9.1 Episode Generation Module

The requirement is satisfied as it generates the EBA variables

3.10 R10

3.10.1 Episode Generation Module

The requirement is satisfied as the generated EBA variables are stored in a csv

3.11 R11

3.11.1 Network Module

The requirement is satisfied as it creates the intermediate object used to build a route from. The object is directed edge graph that can be with routing algorithms.

3.11.2 ShortestRouteTrace Module

The requirement is satisfied as it finds shortest route for a given trace input along with the option to customize how the module is called.

3.11.3 ShortestRouteEpisode Module

The requirement is satisfied as it finds shortest route for a given episode input along with the option to customize how the module is called.

3.12 R12

3.12.1 Alternative Module

The requirement is satisfied as it finds shortest route for a bus transportation type along with he option to customize how the module is called.

3.13 R13

3.13.1 Fetch Activity Location Module

The system successfully produces a output CSV file which includes a description of each activity location

3.13.2 Activity Location Module

The system successfully create Activity Location Object has description attribute called amenity

3.13.3 Point Module

The system outputs Point latitude and longitude for each stop point in activity locations csv

3.14 R14

3.14.1 Fetch Activity Location Module

The system successfully produces a output CSV file which includes a description of each activity location

3.14.2 Activity Location Module

The system successfully create Activity Location Object that can be stored in csv files

3.14.3 Activity Location Module

The system successfully ouputs point attributes in csv

3.15 R15

3.15.1 Episode Generation Module

The requirement is satisfied as it generates the the trip segments csv file for a given GPS data set.

3.16 R16

3.16.1 Transformation Module

The system successfully produces a standard output CSV file with the requested detail as part of the stats function in episodeGeneration. It displays number of mode changes, ping frequency and number of trips taken.

3.17 R17

3.17.1 PreProcessing Module

The system successfully identifies different traces from the user input CSV, segmenting input based on deviceID if ID is provided by the user.

3.18 R18

This requirement will be fulfilled by packaging all modules into a python library so that the user can call the functions integrated in his/her own code. **This will be statisfied and documented in the REV1 version of VnV-Report.**

4 Nonfunctional Requirements Evaluation

4.1 Usability

These NFRs are in the process of being evaluated by asking supervisor Dr. Paez to test the system out along with a testing guide and questionnaire. This review is tentatively scheduled for March 13th.

4.1.1 NFR1

This will be evaluated in REV1 by testing the toolbox as a python library.

4.1.2 NFR2

This will be evaluated in REV1 through the use testing done by Dr. Paez.

4.1.3 NFR3

This will be evaluated in REV1 through the use testing done by Dr. Paez especially when using the interactive maps outputted by the toolbox.

4.1.4 NFR4

This will be evaluated in REV1 through the use testing done by Dr. Paez.

4.1.5 NFR6

This will be evaluated in REV1 through the use testing done by Dr. Paez.

4.1.6 NFR8

It is satisfied because the system is able to process a request using previously requested GPS data upon user's request. Since all files created are transferrable, the user can continue or analyse data at any point.

4.1.7 NFR26

It is satisfied because any transferable entities or communication dialogues use no violate words. The system uses English language to stay consistent.

4.1.8 NFR27

Internally the toolbox ensures the any data is clean and ready to be passed for any analysis functions. On the other hand any external data is cleaned by the preprocessing module by ignoring the incorrect GPS coordinates.

4.2 Performance

The performance of the program is determined by the *Main Module* if it can:

- Correctly calls the functions in the sub-modules are return correct outputs
- Handles Stress Testing.

4.2.1 NFR9

It is satisfied since the run-time measured in Stress Testing is less than 6000 seconds, plus the edge cases also pass.

4.2.2 NFR11

It is satisfied as the manual testing done for the mapping module showed the exact behaviour when mapped with a reliable mapping tool (Google Maps). This enforces that the toolbox mapping is accurate to 80% because the human eye would notice irregularities when the accuracy percentage is below 80%.

4.2.3 NFR12

Even though the program handles stress testing, the edge case(i.e. process through 47.3 million data points) will be evaluated in REV1.

4.3 Operation and Environmental

Many operation and environmental tests depend on the completely packaged python library. As such, we do not have robust operation and environmental testing. Currently, we have tested that the library works and unit tests pass on Windows, OSX, and Linux operating systems.

4.3.1 NFR14

As we have tested that the toolbox works on different kinds of operating systems, this requirement is fulfilled.

4.4 Maintainability and Support

4.4.1 NFR15

This will be evaluated in REV1 after testing the toolbox as a library.

4.5 Security

The toolbox stores all the data analyzed locally, which keeps user data secure.

4.5.1 NFR16

It is satisfied as long as the user has the authority of editing the raw data, this requirement is always fulfilled and has the appropriate applications to open the data files.

4.5.2 NFR17

It is fulfilled since all files generated are transferrable and all the functionalities are accessible through function calls.

4.5.3 NFR18

It is satisfied as the mathematical computations of the system are reliable under the assumption that Python has reliable computation services.

4.5.4 NFR19

It is satisfied as the system only modifies files that the user passes in the function call by providing the file path.

4.5.5 NFR20

It is satisfied as the manual testing done for the mapping module showed the exact behaviour when mapped with a reliable mapping tool (Google Maps). This enforces that the toolbox mapping is accurate to 80% because the human eye would notice irregularities when the accuracy percentage is below 80%.

4.5.6 NFR21

It is satisfied because the toolbox displays the exceptions handled.

4.5.7 NFR22

This will be evaluated in REV1.

4.5.8 NFR23

Since all the data are stored locally on the user's computer only, this requirement is met.

4.5.9 NFR24

Since all the data are stored locally on the user's computer only, this requirement is met.

4.5.10 NFR25

This will be evaluated by REV1 through the use of Pytest and manual verification methods.

5 Comparison to Existing Implementation

The existing implementation of the GERT toolbox has no unit or integration testing, no code coverage metrics, and very little formal documentation. Integration testing of the original GERT toolbox is reliant on the ARCGIS software and there is no existing method to test each of the components of the toolbox.

Given the open source nature of the yoGERT toolbox. The yoGERT team ensured that each piece of functionality within the toolbox was accessible to developers and thoroughly documented and tested. This makes it easier for future developers to easily jump in and test their changes. There was a focus on scalability of the tests because when more open source developers contribute, the tests should be easily extendible to these new cases. Additionally, the developers used more traditional software testing methods such as statement and branch coverage to measure how robust the tests are and ensure they stay up to a certain metric.

Furthermore, compared to the implementation of the original toolbox, yoGERT masks less of the intermediate steps allowing users to explore intermediate data themselves. For example, rather than just generating activity locations, users can also view the data at each step of the way, easily understanding the prerequisites to generating activity locations. This transparency made testing extra important, because it parameterized and allows user input for many assumptions that the original toolbox made and hence opens the doors to lots of invalid/undefined inputs.

6 Unit Testing

6.1 Network Graph Module

Test Id	Inputs	Expected Values	Actual Values	Result
6.2.1.1	*"trace1.csv", "drive", False*	*networkx.classes.multidigraph.MultiDiGraph* object	*networkx.classes.multidigraph.MultiDiGraph* object	Pass
6.2.1.2	*"1_episode.csv"*	*networkx.classes.multidigraph.MultiDiGraph* object	*networkx.classes.multidigraph.MultiDiGraph* object	Pass
6.2.1.3	*"trace1.csv", "drive", False, (43.60902479751416, -79.69011484642793)*	3742825813	3742825813	Pass
6.2.1.4	*"trace1.csv", "drive", False*	"drive"	"drive"	Pass
6.2.1.5	*"trace1.csv", "drive", False, (43.59937567752286, -79.67924717546673)*	*OutOfBoundsCoordException*	*OutOfBoundsCoordException*	Pass
6.2.1.6	*"trace1.csv", "stop", False*	*InvalidModeException*	*InvalidModeException*	Pass
6.2.1.7	*"", "drive", False*	*EmptyFilePath- Exception*	*EmptyFilePath- Exception*	Pass

6.2 Shortest Route Trace Module

Test Id	Inputs	Expected Values	Actual Values	Result
6.2.3.1	*NetworkGraph object, "1_episode.csv"*	*ShortestRouteTrace, 30*	*ShortestRouteTrace, 30*	Pass
6.2.3.2	*NetworkGraph object, "1_episode.csv", "distance"*	*InvalidWeightException*	*InvalidWeightException*	Pass

6.3 Shortest Route Episode Module

Test Id	Inputs	Expected Values	Actual Values	Result
6.2.2.1	*NetworkGraph object, "1_episode.csv"*	*ShortestRouteEpisode, 26*	*ShortestRouteEpisode, 26*	Pass
6.2.2.2	*NetworkGraph object, "1_episode.csv", "time", True, 25*	*ShortestRouteEpisode, 28*	*ShortestRouteEpisode, 28*	Pass
6.2.2.3	*NetworkGraph object, "1_episode.csv", "time", False*	*ShortestRouteEpisode, 1*	*ShortestRouteEpisode, 1*	Pass
6.2.2.4	*NetworkGraph object, "1_episode.csv", "distance"*	*InvalidWeightException*	*InvalidWeightException*	Pass

6.4 Alternative Route Module

Test Id	Inputs	Expected Values	Actual Values	Result
6.2.4.1	"trace1.csv"	*ShortestRouteTrace, 30*	*ShortestRouteTrace, 30*	Pass
6.2.4.2	"trace1.csv", "distance"	*InvalidWeightException*	*InvalidWeightException*	Pass

6.5 Mapping Module

Test Id	**Inputs**	**Expected State**	**Actual State**	**Result**
6.2.5.1	*NetworkGraph object, ShortestRouteTrace object, ”test_TraceRouteMap.html”*	Creation of test_TraceRouteMap.html file	Creation of test_TraceRouteMap.html file	Pass
6.2.5.2	*NetworkGraph object, ShortestRouteEpisode object, ”test_EpisodeRouteMap.html”*	Creation of test_EpisodeRouteMap.html file	Creation of test_EpisodeRouteMap.html file	Pass
6.2.5.3	*NetworkGraph object, ShortestRouteTrace object, ”test_AlternativeRouteMap.html”*	Creation of test_AlternativeRouteMap.html file	Creation of test_AlternativeRouteMap.html file	Pass
6.2.5.4	”trace-activityLocation.csv”, ”stops.csv”, ”test_ActivityLocationMap.html”	Creation of test_ActivityLocationMap.html file	Creation of test_ActivityLocationMap.html file	Pass
6.2.5.5	”1_episode.csv”, ”test_EpisodeMap.html”	Creation of test_EpisodeMap.html file	Creation of test_EpisodeMap.html file	Pass

6.6 Transformation Module

Test Id	**Inputs**	**Expected State**	**Actual State**	**Result**
6.2.8.1	*Incorrect filepath not to the stop.csv passed to stoprelated function*	InvalidFilePath exception thrown	InvalidFilePath exception thrown	Pass
6.2.8.2	*FilePath to stop GPS data passed to stoprelated function*	List of Point type Objects generated	List of Point type Objects generated	Pass
6.2.8.3	*FilePath to episode GPS data passed to episoderelated function*	List of Point type Objects generated	List of Point type Objects generated	Pass
6.2.8.4	*FilePath to trace GPS data passed to tracerelated function*	List of Point type Objects generated	List of Point type Objects generated	Pass
6.2.8.5	*FilePath to trace GPS data passed to summarymode function*	string value of most frequent mode of travel	string value of most frequent mode of travel	Pass
6.2.8.6	*Incorrect filepath not to the trace.csv passed to tracerelated function*	InvalidFilePath exception thrown	InvalidFilePath exception thrown	Pass
6.2.8.7	*Incorrect filepath not to the episode.csv passed to episoderelated function*	InvalidFilePath exception thrown	InvalidFilePath exception thrown	Pass
6.2.8.8	*Incorrect filepath not to the trace.csv passed to summarymode function*	InvalidFilePath exception thrown	InvalidFilePath exception thrown	Pass
6.2.8.9	*Incorrect input error message for convertActivityLocation(ActvityLoactionList) is generated*	InvalidInput Error exception thrown	InvalidInout Error exception thrown	Pass
6.2.8.10	*Incorrect input error message for convertActivityLocation(ActvityLoactionList) is generated*	list of [point lat, point lon,[nested list of activity locations attributes]] is returned	list of [point lat, point lon,[nested list of activity locations attributes]] is returned	Pass

6.2.8.11	*Incorrect file error message for convertActivityCSVis generated*	Invalid file error is returned	lInvalid file error is returned	Pass
6.2.8.12	*Activity objects in the right format to be used are created by convertActivityCSV*	list of activity location objects are returned	list of activity location objects are returned	Pass
6.2.8.13	*Incorrect input error message for convertListToActivityLocationObject(activityLocationList) is generated*	InvalidInput Error exception thrown	InvalidInout Error exception thrown	Pass
6.2.8.14	*ist Activity objects of activity location type are generated*	list Activity objects of activity location type are generatedn	list Activity objects of activity location type are generated	Pass

6.7 Fetch Activity Location Module

Test Id	Inputs	Expected Values	Actual Values	Result
6.2.7.1	*”stops.csv”, ”trace/trace1/trace-activityLocation.csv”*	*trace-activityLocation.csv*	*”trace/trace1/trace-activityLocation.csv”*	Pass
6.2.7.2	*””, ”trace/trace1/trace-activityLocation.csv”*	*”Input file is invalid”*	*”Input file is invalid”*	Pass
6.2.7.3	*”stops.csv”, ”trace/trace1/trace-activityLocation.csv”*	*No error message in output*	*No error message in output*	Pass
6.2.7.4	*”stops.csv”, ””*	*”Output file is invalid”*	*”Output file is invalid”*	Pass
6.2.7.5	*””, ”trace/trace1/trace-activityLocation.csv”*	*No error message about output file*	*No error message about output file*	Pass
test-ST-10	*”stops.csv”, ”trace/trace1/trace-activityLocation.csv” (server is also unavailable)*	Log warning in output displaying ”Stop point activity locations not found due to sever being unavailable”)	Log warning in output displaying ”Stop point activity locations not found due to sever being unavailable”	Pass
6.2.7.6	*”stops.csv”, ”trace/trace1/trace-activityLocation.csv”*	trace/trace1/trace-activityLocation.csv generated with correct file content	trace/trace1/trace-activityLocation.csv generated	Pass

6.8 PreProcessing Module

Test Id	Inputs	Expected Values	Actual Values	Result
6.2.9.1	*trace1.csv test_ExpectedInput*	*trace0.csv in test_ExpectedInput*	*trace0.csv in test_ExpectedInput*	Pass
6.2.9.2	*trace1.csv test_DMSInput*	*trace0.csv without DMS values in test_DMSInput*	*parsing error*	Fail
6.2.9.3	*trace1_InvalidLL.csv test_InvalidLL*	*trace0.csv in test_InvalidLL without invalid rows*	*trace0.csv in test_InvalidLL without invalid rows*	Pass
6.2.9.4	[78°55’44.29458”N, 124° 4’ 58” W]	[78.92897071666667, 124.08277777777778]	[78.92897071666667, 124.08277777777778]	Pass
6.2.9.5	*trace1_InvalidLL.csv test_InvalidLL*	InvalidInputException	InvalidInputException	Pass

6.9 Episode Generation Module

Test Id	Inputs	Expected Values	Actual Values	Result
6.2.6.1	*trace_1.csv in test_csv*	*Creation of trace.csv in test_trace*	*Creation of segments.csv in test_trace*	Pass
6.2.6.2	*trace.csv in test_trace*	*Creation of segments.csv in test_trace*	*Creation of segments.csv in test_trace*	Pass
6.2.6.3	*segments.csv in test_trace*	*Creation of stops.csv in test_trace*	*Creation of stops.csv in test_trace*	Pass
6.2.6.4	*stops.csv in test_trace, time_tolerance*	*stops.csv in test_trace*	*stops.csv in test_trace*	Pass
6.2.6.5	*stops.csv in test_trace, distance_tolerance*	*stops.csv in test_trace*	*stops.csv in test_trace*	Pass
6.2.6.6	*stops.csv in test_trace*	*episode in test_trace*	*episode.csv in test_trace*	Pass
6.2.6.7	*test_trace folder*	*summarymode.csv in test_trace*	*summarymode.csv in test_trace*	Pass
6.2.6.8	*test_trace folder*	*stats.csv in test_trace*	*stats.csv in test_trace*	Pass

6.10 Activity Location Module

Test Id	Inputs	Expected Values	Actual Values	Result
6.2.10.1	*ActivityLocation(”Lemon Bar”, 43.651504, -79.386657, ”Juice”)*	*Activity Location Object*	*Activity Object*	Pass
6.2.10.2	*ActivityLocation(”Lemon Bar”, 43.651504, -79.386657, ”Juice”)*	*Lemon Bar*	*Lemon Bar*	Pass
6.2.10.3	*ActivityLocation(”Lemon Bar”, 43.651504, -79.386657, ”Juice”)*	*43.651504*	*43.651504*	Pass
6.2.10.4	*ActivityLocation(”Lemon Bar”, 43.651504, -79.386657, ”Juice”)*	*-79.386657*	*-79.386657*	Pass
6.2.10.5	*ActivityLocation(”Lemon Bar”, 43.651504, -79.386657, ”Juice”)*	*”Juice”*	*”Juice”*	Pass
6.2.10.6	*ActivityLocation(”Lemon Bar”, 43.651504, -79.386657)*	*”None”*	*”None”*	Pass

6.11 Point Module

Test Id	Inputs	Expected Values	Actual Values	Result
6.2.11.1	*Point(43.651605, -79.386759,"17:22:02", "mode.DRIVE")*	*Point Object*	*Point Object*	Pass
6.2.11.2	*Point(43.651605, -79.386759,"17:22:02", "mode.DRIVE")*	*43.651605*	*43.651605*	Pass
6.2.11.3	*Point(43.651605, -79.386759,"17:22:02", "mode.DRIVE")*	*-79.386759*	*-79.386759*	Pass
6.2.11.4	*Point(43.651605, -79.386759,"17:22:02", "mode.DRIVE")*	*"17:22:02"*	*"17:22:02"*	Pass
6.2.11.5	*Point(43.651605, -79.386759,"17:22:02", "mode.DRIVE")*	*"mode.DRIVE"*	*"mode.DRIVE"*	Pass
6.2.11.6	*Point(43.651605, -79.386759)*	*Point Object where pointOjbect.time = None and pointObject.mode = None*	*Point Object where pointOjbect.time = None and pointObject.mode = None*	Pass

7 Changes Due to Testing

7.1 Network Graph Module

- New exceptions to handle system failure for boundary cases and unexpected inputs.

7.2 Shortest Route Trace Module

- Descriptive exception print statements.

7.3 Shortest Route Episode Module

- Handling different inputs when customization parameters are inputted
- Descriptive exception print statements.

7.4 Mapping Module Module

- Fixing bugs when formatting text to be displayed on the interactive object.
- Handling variations of mapping different routes using on type of function call.

7.5 Episode Generation Module

- Folder creation for each method has an exception to catch duplicate file creation
- Added helper function for episode generation to create trace and all other sub-folders together

7.6 Fetch Activity Location Module

- Updated to account for input file path error (with descriptive error message)
- Updated to account for output file path error (with descriptive error message)
- Updated to account for server unavailable error(with descriptive error message)

7.7 PreProcessing Module

- Updated to account for multiple IDs in an input trace.
- Updated regular expression for DMS detection (needs to be further updated).
- Updated to account for user inputted trace folder

7.8 Main Module

- Main Module has been removed since the user can directly call the functions in the sub-modules. Some pieces of code have been moved to the sub-modules.

8 Automated Testing

8.1 Unit Tests - PyTest

All unit test cases were implemented using the PyTest library. This allowed for clear formatted reports and the ability to easily check coverage metrics.

8.2 Stress Testing

As mentioned in *Section 4.2,* the Stress Testing is used to determine to performance of the program. We test the functions in each module under heavy loads via an encapsulated file(i.e. main.py) to stimulate that the user directly call the functions.

The test case is set as the following: for each function in the code file, the automated test run 10 times and record: total run-time, average run-time, and the number of correct outputs.

Notes: the run-time is fulfilled by python's built-in timeit.default_timer() method.

8.2.1 Results for *preProcessing* module

Currently this section has not been covered. This test will be performed in REV1.

8.2.2 Results for *episodeGeneration()*

The test file contains 3830 travel points and a total number of 12 calcualted stop points in the corresponding generated stop.csv file. The result of running 10 times is:

- 10/10 cases successfully calculate 10 stop points and stored in the stops.csv file.
- Total run-time for 10 cases: 3.7565 seconds.
- Average run-time for 1 case: 0.3757 seconds.

8.2.3 Results for *findActivityLocations()*

The test file(stops.csv) has 3 stop points in downtown area of San Jose, California, USA, with a searching radius of 500 metres. The *findActivityLocations()* function should return more than 50 possible nearby-locations. The result after running the test case is at the following:

- 10/10 cases successfully generated all possible nearby-locations
- Total run-time for 10 cases: 20.0785 seconds.
- Average run-time for 1 case: 2.0079 seconds.

However, it should be noted that this function highly depends on the public server we are pulling responds(i.e. information of the nearbyy-location). In other words, if the server is under heavy load, the performance may become poorer or even fail to response.

8.2.4 Results for *generateShortestEpisodeRoute()*

For this testing, we use the file that contains a total number of 324 travel episodes and the result is:

- 10/10 cases successfully the *NetworkGraph* and *ShorestRouteEpisode* type of data
- Total run-time for 10 cases: 275.2367 seconds.
- Average run-time for 1 case: 27.5237 seconds.

8.2.5 Results for *generateShortestTraceRoute()*

By using the test file that has 476 trace points, and the trace-mode set as "drive", the result is at the following:

- 10/10 cases successfully the *NetworkGraph* and *ShorestRouteTrace* type of data
- Total run-time for 10 cases: 860.8367 seconds.
- Average run-time for 1 case: 86.0837 seconds.

8.2.6 Results for *generateAlternativeRoute()*

Similar to *Section 8.2.4,* we used the same trace file for testing this function. The result is listed below:

- 10/10 cases generates *AlternvativeRoute* type of data correctly.
- Total run-time for 10 cases: 1158.3500 seconds
- Average run-time for 1 case: 115.8350 seconds

8.2.7 Summary of Stress Testing

8.3 Integration Testing

As the goal of Integration Testing is to determine whether the modules are integrated properly, the encapsulated functions in main.py file, which connects all the sub-modules, should tested through Integration Testing. By referring to *Section 8,* as all the functions in the file pass Stress Testing, this indicates that all modules for the program are integrated properly and working as a group, therefore the Integration Testing should be considered as a pass.

8.4 Regression Testing

First of all, the changes of the Code are listed in *Section 7.*

During the processing of changing the code, comparing to the output between two versions, the expected outputs matches the actual outputs. In other words, the stability of the program is remain unchanged during this process.

9 Trace to Requirements

Test Cases	**Covered FR and NFRS**
Stress Testing, Integration Testing	NFR9, NFR11, NFR12, NFR18, NFR19, NFR25, NFR27
Regression Testing	NFR4, NFR9, NFR11, NFR12, NFR18, NFR19, NFR21, NFR24, NFR25

FR/T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
6.2.1.1				X							X							
6.2.1.2				X							X							
6.2.1.3				X							X							
6.2.1.4				X							X							
6.2.1.5				X							X							
6.2.1.6				X							X							
6.2.1.7				X							X							
6.2.4.1				X							X							
6.2.4.2				X							X							
6.2.4.3				X							X							
6.2.4.4				X							X							
6.2.3.1				X							X							
6.2.3.2				X							X							
6.2.4.1				X								X						
6.2.4.2				X								X						
6.2.5.1			X															
6.2.5.2			X															
6.2.5.3			X															
6.2.5.4			X															
6.2.5.5			X															
6.2.6.1					X													
6.2.6.2							X											
6.2.6.3							X											
6.2.6.4									X									
6.2.6.5									X									
6.2.6.6									X									
6.2.6.7															X			
6.2.6.7																X		
6.2.7.1														X				
6.2.7.2								X										
6.2.7.3								X										
6.2.7.4														X				
6.2.7.5														X				
6.2.7.6														X				
6.2.8.1																		
6.2.8.2																		
6.2.8.3																		
6.2.8.4																	X	
6.2.8.5																X		
6.2.8.6																		

FR/T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
6.2.8.7																		
6.2.8.8																		
6.2.8.9								X										
6.2.8.10								X										
6.2.8.11								X										
6.2.8.12								X										
6.2.8.13								X										
6.2.8.14								X										
6.2.9.1	X	X	X	X													X	
6.2.9.2	X	X	X	X														
6.2.9.3	X	X	X	X														
6.2.9.4				X														
6.2.9.5		X																
6.2.10.1								X					X	X				
6.2.10.2			X					X					X	X				
6.2.10.3			X					X					X	X				
6.2.10.4			X					X					X	X				
6.2.10.5								X					X	X				
6.2.10.6								X					X	X				
6.2.11.1								X			X	X	X					
6.2.11.2								X			X	X	X	X				
6.2.11.3								X			X	X	X	X				
6.2.11.4								X			X	X	X	X				
6.2.11.5								X			X	X	X	X				
6.2.11.6								X			X	X	X					

10 Trace to Modules

All unit and stress tests are organized by module.

11 Code Coverage Metrics

The pytest-cov plugin was used to test coverage metrics. The team chose to test both statement and branch coverage, as this allowed developers to see if tests accurately represented all paths within the code. The goal stated in earlier documents was 80% coverage, initial tests do not meet this metric for branch coverage. However, the team plans to create more tests before the final demonstration to ensure key metrics have been achieved.

Test Name	**Coverage Metric**	**Result %**
NetworkGraph	Statement — branch	94% — 56%
ShortestRouteTrace	Statement — branch	94% — 100%
ShortestRouteEpisode	Statement — branch	96% — 87%
AlternativeRoute	Statement — branch	100% — 12%
Fetch Activity Location	Statement — branch	89% — 87%
PreProcessing	Statement — branch	87% — 84%
EpisodeGeneration	Statement — branch	TBD(Debugging for code coverage) — TBD(Debugging for code coverage)
Transformation	Statement — branch	TBD(Debugging for code coverage) — TBD(Debugging for code coverage)
Activity Location	Statement — branch	100% — 100%
Point	Statement — branch	100% — 100%