Running Head: Lab 3 – Traffic Wizard Prototype Test Plan

Lab 3 – Traffic Wizard Prototype Test Plan/Procedure

Sections 4, 5, and 6

Traffic Wizard – Blue Team

Old Dominion University

CS 411 - Brunelle

Last Modified: April 3, 2012

Version: 1.0

Table of Contents

1 Objectives

2 References

3 Test Plan

3.1 Testing Approach

3.2 Identification of Tests

3.3 Test Schedule

3.4 Fault Reporting and Data Recording

3.5 Resource Requirements

3.6 Test Environment

4 Test Responsibilities

The responsibilities for each team member during the prototype demonstration are outlined in Table 4. For the most part, team members with a certain realm of expertise will present the respective component of the prototype. The main presenters will be Andrew Crossman, Andrew McKnight, and Nick MacLeod, with Sujani Godavarthi, Binh Dong and Thomas Kennedy adding insight for their developed components.

Team Member	Responsibilities
Thomas Kennedy	Databases
Andrew Crossman	Presenter/Simulation Console Operator
Andrew McKnight	Presenter/Smartphone App
Nick MacLeod	Presenter/Algorithms
Sujani Godavarthi	Algorithms
Binh Dong	Hardware

Table 4: Test Responsibilities

5 Test Procedures

Test procedures for Traffic Wizard have been developed to ensure the functionality of the prototype is attained and correct. The test procedures are represented in a format that contains the category, subcategory, purpose, requirements covered, steps to test, and expected results. Each step in each test may either pass or fail, and a comment field is provided for tester analysis.

Test Category: Unit	**Description:** Traffic Wizard Database Schema and Interface		
Test Case: 1.1.1	**Case Name:** Database Structure Test	**Version:** 1.0	**Written By:** Thomas Kennedy
Requirements Fulfilled: 3.1.1.1, 3.1.1.2, 3.1.1.3	**Purpose:** Verify the structure of all tables and fields		
Setup Conditions: • MySQL is installed and all tables are implemented. • Driver Profile Database has been created. • Virtual Checkpoint Database has been created. • Speed Limit Database has been created. • Database Schemas are available.			

	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	Query the database to display the structure all tables in the Driver Profile Database.			All database table fields are displayed.
2	Visually verify that the retrieved fields correspond to the database design.			Driver Profile Database tables match the database schemas.
3	Repeat steps 1 and 2 for the Virtual Checkpoint Database.			Virtual Checkpoint Database tables match the database schemas.
4	Repeat steps 1 and 2 for the Speed Limit Database.			Speed Limit Database tables match the database schemas.

Test Category: Unit	**Description:** Test the aggregate speed function		
Test Case: 2.1.1	**Case Name:** TestAggregateSpeeds	**Version:** 1.0	**Written By:** Nicholas MacLeod
Requirements Fulfilled: 3.1.2.1	**Purpose:** To determine whether the aggregate speed function is working and if it is accurate.		
Setup Conditions: • Virtual Checkpoint Database must be set up and must allowed read/write access. • Must be able to receive or simulate checkpoint data			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Data comes in within checkpoint's specified time range			The old data and new data will be aggregated together and written to the database.
2	Data comes in after the checkpoint's specified time range			The new speed is written to the database.
3	Multiple user data is received for a checkpoint within one update			The new data will be aggregated together and the weights when aggregating with the old data will adjust based on the number of updates.
4	No new data received			Checkpoint speed should remain unchanged.

Test Category: Unit	**Description:** Check if the source code was written in Java or C++.		
Test Case: 2.2.1	**Case Name:** Source Code	**Version:** 1	**Written By:** Binh Dong
Requirements Fulfilled: 3.1.2.3.1	**Purpose:** To check if source code was written in Java or C++.		
Setup Conditions: • Need Source Code			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Open Source Code			Source Code should be written in C++ or Java.

Test Category: Unit	**Description:** Test Case to open the Virtual Checkpoint Database			
Test Case: 2.2.2	**Case Name:** Checkpoint Reallocation – Open Database	**Version:** 1	**Written By:** Binh Dong	
Requirements Fulfilled: 3.1.2.3.3	**Purpose:** To test the ability to open the Virtual Checkpoint Database.			
Setup Conditions: • Pass test case 2.2.1, Need a server, Need a client/smartphone or simulation console.				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Open Virtual Checkpoint Database			No errors returned.

Test Category: Unit	**Description:** Test Case to open the Virtual Checkpoint Database			
Test Case: 2.2.3	**Case Name:** Checkpoint Reallocation – Open Database	**Version:** 1	**Written By:** Binh Dong	
Requirements Fulfilled: 3.1.2.3.4	**Purpose:** To test the ability to open the speed limit database.			
Setup Conditions: • Pass test case 2.2.1, Need a server, Need a client/smartphone or simulation console.				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Open Speed Limit Database			No errors returned.

Test Category: Unit	**Description:** Test Cases to verify the Checkpoint Reallocation algorithm		
Test Case: 2.2.4	**Case Name:** Add Checkpoint	**Version:** 1	**Written By:** Binh Dong
Requirements Fulfilled: 3.1.2.3.7	**Purpose:** To test the ability to decrease the distance between two adjacent checkpoints as traffic conditions become heavy.		
Setup Conditions: • Pass test cases 2.2.1 – 2.2.3, Need a server, Need a client/smartphone or simulation console.			

	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	Select a Virtual Checkpoint			Virtual Checkpoint meta data will be displayed: Latitude, Longitude, Speed, Direction, and checkpoint condition.
2	Add Trigger			If the Checkpoint's condition is inactive or traffic heavy, the add checkpoint algorithm must be triggered.

Test Category: Unit	**Description:** Test Cases to verify the Checkpoint Reallocation algorithm		
Test Case: 2.2.5	**Case Name:** Delete Checkpoint	**Version:** 1	**Written By:** Binh Dong
Requirements Fulfilled: 3.1.2.3.8	**Purpose:** To test the ability to increase the distance between two adjacent checkpoints as traffic conditions become optimal.		
Setup Conditions: • Passed unit tests 2.2.1-2.2.4 Need a server, Need a client/smartphone or simulation console.			

	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	Select a Virtual Checkpoint			Virtual Checkpoint meta data will be displayed: Latitude, Longitude, Speed, Direction, and checkpoint condition.
2	Delete Trigger			If the Checkpoint's condition reads optimal traffic, the delete checkpoint algorithm must be triggered.

Test Category: Unit	**Description:** Test coding language used in Route Matcher algorithm			
Test Case: 2.3.1.	**Case Name:** Algorithm RM Language Test	**Version:** 1.0	**Written By:** Andrew Crossman	
Requirements Fulfilled: 3.1.2.4.1.	**Purpose:** Verify that the Route Matcher algorithm is coded in either C++ or Java coding languages			
Setup Conditions: • Source code file for Route Matcher algorithm opened from server				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Visually inspect source code			Code is in C++ or Java

Test Category: Integration	**Description:** Test that Route Matcher algorithm can access Virtual Checkpoint Database			
Test Case: 2.3.2.	**Case Name:** Algorithm RM VC Database Connect Test	**Version:** 1.0	**Written By:** Andrew Crossman	
Requirements Fulfilled: 3.1.2.4.2.	**Purpose:** Verify that the Route Matcher algorithm is able to access the Virtual Checkpoint Database to find checkpoint GPS coordinates			
Setup Conditions: • Virtual Checkpoint Database Test Cases (1.2.1-1.2.X) passed • Virtual Checkpoint Database tables available to view • Server is loaded for operation, command line open				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Manually choose a virtual checkpoint from VC Database tables			Note chosen checkpoint's ID and GPS coordinates
2	Start Algorithm Tester from server command line			Each algorithm's name displayed to be selected for test
3	Select Route Matcher			Prompt for algorithm input appears (latitude and longitude coordinate)
4	Enter chosen checkpoint's latitude and longitude coordinates as input			Resulting ID for closest checkpoint to entered coordinates returned
5	Compare displayed ID with chosen checkpoint ID			ID's match

Test Category: Unit	**Description:** Test that Route Matcher algorithm accepts GPS coordinate data as input		
Test Case: 2.3.3.	**Case Name:** Algorithm RM Input Parameter Test	**Version:** 1.0	**Written By:** Andrew Crossman
Requirements Fulfilled: 3.1.2.4.3.	**Purpose:** Verify that the Route Matcher algorithm accepts two floating point values for coordinates as parameters		
Setup Conditions: • Virtual Checkpoint Database Test Cases (1.2.1 - 1.2.X) passed • Virtual Checkpoint Database tables available to view • Route Matcher Test Case 2.3.2. passed • Server is loaded for operation, command line open			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Manually choose a virtual checkpoint from VC Database tables			Note chosen checkpoint's ID and GPS coordinates
2	Start Algorithm Tester from server command line			Each algorithm's name is displayed to be selected for test
3	Select Route Matcher			Prompt for algorithm input appears (latitude and longitude coordinate)
4	Enter chosen checkpoint's latitude and longitude coordinates as input			Resulting ID for closest checkpoint to entered coordinates returned

Test Category: Unit	**Description:** Test that Route Matcher algorithm is able to return a checkpoint ID within the set proximity of given coordinates		
Test Case: 2.3.4.	**Case Name:** Algorithm RM Proximity Test	**Version:** 1.0	**Written By:** Andrew Crossman
Requirements Fulfilled: 3.1.2.4.4.	**Purpose:** Verify that the Route Matcher algorithm is able to return a checkpoint ID within 100 feet of provided coordinates		
Setup Conditions: • Virtual Checkpoint Database Test Cases (1.2.1 - 1.2.X) passed • Virtual Checkpoint Database tables available to view • Route Matcher Test Case 2.3.2 – 2.3.3 passed • Server is loaded for operation, command line open			

	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	Manually choose a virtual checkpoint from VC Database tables			Note chosen checkpoint's ID and GPS coordinates
2	Add or subtract 0.001 from latitude coordinate			Note new GPS coordinates
3	Start Algorithm Tester from server command line			Each algorithm's name is displayed to be selected for test
4	Select Route Matcher			Prompt for algorithm input appears (latitude and longitude coordinate)
5	Enter altered latitude and longitude coordinates as input			Resulting ID for closest checkpoint to entered coordinates returned
6	Compare displayed ID with chosen checkpoint ID			ID's match
7	Repeat Steps 2-6 with longitude instead of latitude			Same checkpoint ID returned as Step 5

Test Category: Unit	**Description:** Test that Route Matcher algorithm is able to return no checkpoint ID if input coordinates are too far			
Test Case: 2.3.5.	**Case Name:** Algorithm RM False Proximity Test	**Version:** 1.0	**Written By:** Andrew Crossman	
Requirements Fulfilled: 3.1.2.4.5.	**Purpose:** Verify that the Route Matcher algorithm is able to return that no checkpoint ID is within 100 feet of provided coordinates			
Setup Conditions: • Virtual Checkpoint Database Test Cases (1.2.1 - 1.2.X) passed • Virtual Checkpoint Database tables available to view • Route Matcher Test Case 2.3.2 – 2.3.4 passed • Server is loaded for operation, command line open				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Manually choose a virtual checkpoint from VC Database tables			Note chosen checkpoint's ID and GPS coordinates
2	Add or subtract 1 from latitude coordinate			Note new GPS coordinates
3	Ensure new GPS coordinates are not within 0.001 of another virtual checkpoint (check table)			If too close, repeat Step 2 with larger value If not within range, proceed to Step 4
4	Start Algorithm Tester from server command line			Each algorithm's name is displayed to be selected for test
5	Select Route Matcher			Prompt for algorithm input appears (latitude and longitude coordinate)
6	Enter altered latitude and longitude coordinates as input			Message stating "No checkpoint near given coordinates" displayed
7	Repeat Steps 2-6 with longitude instead of latitude			Same result as Step 6

Test Category: Unit	**Description:** Route Virtual Checkpoint parsing and analysis.		
Test Case: 2.4.1	**Case Name:** Route Analysis Accuracy Test	**Version:** 1.0	**Written By:** Thomas Kennedy
Requirements Fulfilled: 3.1.2.5.1	**Purpose:** Verify that the Route Analysis Algorithm properly validates a route against the Virtual Checkpoint Database.		
Setup Conditions: • Test 1.1.1 has been passed			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Enter a valid route for analysis.			A set of GPS coordinates will be parsed from the route data
2	Verify that the algorithm queries the database for Virtual Checkpoint information using the GPS coordinates values from step 1			All Virtual Checkpoints along the route specified in the previous step are returned.
3	Verify that Virtual Checkpoints have been returned.			Virtual Checkpoints are available.
4	Verify that the Virtual Checkpoint data is returned to the smartphone application			The smartphone receives a list of updated Virtual Checkpoint data.

Test Category: Integration	**Description:** Route Analysis congestion calculation and aggregation		
Test Case: 2.4.2	**Case Name:** Route Analysis Data Test	**Version:** 1.0	**Written By:** Thomas Kennedy
Requirements Fulfilled: 3.1.2.5.2, 3.1.2.5.3	**Purpose:** To verify the calculation and communication of congestion data for a user specified route.		
Setup Conditions: • Tests 1.1.1 and 2.4.1 have been passed			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Enter a valid route for analysis.			The algorithm parses the route
2	Verify that the algorithm queries the Virtual Checkpoint Database and aggregates congestion data.			Congestion data is aggregated for current traffic information.
3	Verify that the algorithm has flagged outdated congestion data			The congestion information has been compiled for transmission.
4	Verify that the returned congestion data contains flags for all outdated Virtual Checkpoints without current data.			The returned data contains only current congestion data and flags for data that has been determined to be outdated.
5	Repeat steps 1 through 5 for a designed to trigger a split the congestion calculations into groups			The algorithm parses the route and splits the congestion calculations into groups.
6	Verify that the calculations have been divided into groups.			Verify that each group generates valid output (see step 4).
7	Verify that the groups return data in the appropriate order.			Each group has transmitted the congestion data. The congestion data arrives in order.

Test Category: Unit		**Description:** Test for code language used in the Blockage Finder algorithm.		
Test Case: 2.5.1		**Case Name:** Source Code	**Version: 1.0**	**Written By:** Sujani Godavarthi
Requirements Fulfilled: 3.1.2.6.1		**Purpose:** Implementing and checking the Blockage Finder algorithm is coded in C++/ Java coding languages.		
Setup Conditions: • Source code file for blockage finder algorithm to be supported in the server.				
	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	Checking source code			Code is in C++ or Java

Test Category: Integration		**Description:** Testing the user interface to be used on the server for Blockage Algorithm.		
Test Case: 2.5.2		**Case Name:** User Interface	**Version: 1.0**	**Written By:** Sujani Godavarthi
Requirements Fulfilled: 3.1.2.6.3		**Purpose**: Checking the user interface and being supported by the server.		
Setup Conditions: • Support Interface to be used by the server when requested access.				
	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	Checking user interface with the help of server.			Successful

Test Category: Integration	**Description:** Ensuring if information received is valid.		
Test Case: 2.5.3	**Case Name:** Accessing Information	**Version: 1.0**	**Written By:** Sujani Godavarthi
Requirements Fulfilled: 3.1.2.6.4	**Purpose**: Having the ability to access the Virtual Checkpoint Database		
Setup Conditions: • Virtual Checkpoint Database Test Cases (1.2.1-1.2.X) passed • Virtual Checkpoint database tables are available to view			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Virtual Checkpoint Database			If blockage applicable where in virtual checkpoints trigger for data which is being available from the VC Database.

Test Category: Integration	**Description:** Checking the location through Google Maps.		
Test Case: 2.5.4	**Case Name:** Geographical Area	**Version: 1.0**	**Written By:** Sujani Godavarthi
Requirements Fulfilled: 3.1.2.6.5	**Purpose:** Retrieving the latitude and longitude points of that particular region.		
Setup Conditions: • Virtual Checkpoint Database Test Cases (1.2.1-1.2.X) passed			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Geographical area			Blockage is certain in a region where in Google Maps and GPS coordinates are used to identify the location. Checkpoints are deleted if inactive for a defined time.

Test Category: Integration	**Description:** Virtual Checkpoints		
Test Case: 2.5.5	**Case Name:** Virtual Checkpoints	**Version: 1.0**	**Written By:** Sujani Godavarthi
Requirements Fulfilled: 3.1.2.6.6	**Purpose:** Clearing of blockages along the route with respect to Virtual Checkpoint.		
Setup Conditions: • Virtual Checkpoint Database Test Cases (1.2.1-1.2.X) passed			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Virtual Checkpoints			With the help of Virtual Checkpoints, the flow of traffic can be determined during the phase of blockages.
2	Identifying the virtual checkpoints along the road			VC noted in the respective database with latitude, longitude, speed, direction and checkpoint condition.
3	Select Route Matcher			The algorithm input appears to be latitude and longitude coordinates.
4	VC trigger			The data is being triggered to the Virtual Checkpoint and update of traffic.

Test Category: Integration	**Description:** Route Analysis algorithm		
Test Case: 2.5.6	**Case Name:** Route analysis along the chosen path.	**Version: 1.0**	**Written By:** Sujani Godavarthi
Requirements Fulfilled: 3.1.2.6.6	**Purpose**: Verify that the Route analysis algorithm properly validates a route.		
Setup Conditions: • Checking and identifying the appropriate virtual points along the road segment. • Virtual Checkpoint Database Test cases(1.2-1.2X) passed			

	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	Verify the algorithm works along the road where VC are already placed and use GPS coordinates.			Virtual checkpoints are available.
2	Checks the speed and information against the Speed Limit Database and the Virtual Checkpoint Database.			Successful
3	Verify that the VC data is transmitted to the smartphone application.			The smartphone receives a list of the updated Virtual Checkpoint Data.

Test Category: Unit	**Description:** Check Next Checkpoint Estimator calculations		
Test Case: 2.6.1	**Case Name:** Next Checkpoint Estimator calculations	**Version:** 1.0	**Written By:** Andrew McKnight
Requirements Fulfilled: 3.1.2.7.1, 3.1.2.7.3, 3.1.2.7.4	**Purpose:** Ensure the calculations performed by the algorithm are correct		
Setup Conditions: • Simulation Console is running • Client instance has been created on an iOS device and established connection to console • Client instance has received Trip object from console and a vector of coordinates describing the location and speed of the phone along the route • Client instance has begun a trip and passed initial checkpoint			

	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	Request a Location object from the LocationManager object			Callback method is invoked no more than *t*/10 milliseconds, where *t* was the time amount for the last checkpoint
2	Assert that the location obtained and speed are as defined in the coordinate vector			Location returned by LocationManager is equal to the expected location in the coordinate vector ±.01 miles
3	Calculate the time in milliseconds using the Euclidean distance formula between two points and the speed from LocationManager			Magnitude of return value is actual result to be determined by tester's calculations.
4	Determine sign of return value by heading of smartphone and expected heading			If expected heading and measured heading are =, sign is + (positive time until next checkpoint); Otherwise sign is – (negative time to next checkpoint because it has already passed)

Test Category: Unit	**Description:** Next Checkpoint Estimator route deviation test		
Test Case: 2.6.2	**Case Name:** Next Checkpoint Estimator deviation	**Version:** 1.0	**Written By:** Andrew McKnight
Requirements Fulfilled: 3.1.2.7.2	**Purpose:** Test the conditional branch in the algorithm that checks whether a user has deviated from a route		
Setup Conditions: • Simulation Console is running • Client instance has been created on an iOS device and established connection to console • Client instance has received Trip object from console and an actual path to travel, deviating before second checkpoint • Client instance has begun a trip and passed initial checkpoint • Driver has deviated from route according to its Trip object			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Obtain current location from CLLocationManager object			Hook method is called and CLLocation object is obtained. Coordinates should agree with any Console tracking variables.
2	Calculate distance to next checkpoint in trip.			Accurate, non-negative Euclidean distance calculated.
3	Determine cardinal direction from last checkpoint to next checkpoint.			Correct angle in [-180, 180] returned.
4	Obtain heading from CLLocation object			Heading stored in *double* variable.
5	Obtain cardinal directions of last checkpoint and next checkpoint			Directions stored in *double* variables.
6	Compare headings of the smartphone (S), individual checkpoints (C1, C2), and the pair of checkpoints (P) *			$S / [P / \text{abs}(C1-C2)] > S - (S / 5)$ and Next Checkpoint Estimator throws appropriate exception

Test Category: Unit	**Description:** Test Simulation Console Region Selection regions supported to be displayed		
Test Case: 3.1.1.	**Case Name:** Sim Console Region Support Test	**Version:** 1.0	**Written By:** Andrew Crossman
Requirements Fulfilled: 3.1.3.1.1. – 3.1.3.1.2.	**Purpose:** Verify that the Region Selection part of the Simulation Console has each of three region maps available as defined		
Setup Conditions: • Source code folder/files for Simulation Console opened • Region Selection requirement 3.1.3.1.1. available for view for boundary definitions			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Open \maps folder			
2	Visually inspect available map files			Three regions present: small_region, medium_region, large_region
3	Open small_region file			Verify boundaries match requirement from Google map image
4	Open medium_region file			Verify boundaries match requirement from Google map image
5	Open large_region file			Verify boundaries match requirement from Google map image

Test Category: Unit	**Description:** Test Simulation Console arrival and destination points for virtual drivers to enter during simulation runtime		
Test Case: 3.1.2.	**Case Name:** Sim Console Arrival and Destination Test	**Version:** 1.0	**Written By:** Andrew Crossman
Requirements Fulfilled: 3.1.3.1.4. – 3.1.3.1.5.	**Purpose:** Verify that all Simulation Console regions have entry and exit points for virtual drivers as defined in requirement		

Setup Conditions:

- Source code for Simulation Console opened
- Google Maps utility available
- Region Selection requirements 3.1.3.1.4. and 3.1.3.1.5. available for view for entry and exit point locations

	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	Open Regions.cs file			Source code for Region class opens
2	Locate SmallRegion class			
3	Observe GPS coordinate set within ArrivalPoints			Verify coordinates match locations in requirement 3.1.3.1.4. (using Google Maps)
4	Observe GPS coordinate set within DestinationPoints			Verify coordinates match locations in requirement 3.1.3.1.5. (using Google Maps)
5	Repeat Steps 2-4 for MediumRegion and LargeRegion classes			Results from Steps 2-4

Test Category: Unit	**Description:** Test Simulation Console Traffic Scenario Selection options are defined to represent all scenarios		
Test Case: 3.2.1.	**Case Name:** Sim Console Scenario Support Test	**Version:** 1.0	**Written By:** Andrew Crossman
Requirements Fulfilled: 3.1.3.2.1. – 3.1.3.2.2.	**Purpose:** Verify that the Traffic Scenario Selection part of the Simulation Console has all intended scenarios defined with specific attributes for runtime execution		
Setup Conditions: • Source code for Simulation Console opened • Traffic Scenario Selection requirements 3.1.3.2.1 and 3.1.3.2.2. available for view for scenarios			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Open Scenarios.cs file			Source code for Scenario class opens
2	Locate Scenario1 class			
3	Observe values for variables: trafficVolume, congestionRate, blockageRate			Verify variable values are as defined in requirement 3.1.3.2.1. for Scenario 1
4	Observe Arrival object value for variable arrivalRate			Verify variable value is as defined in requirement 3.1.3.2.2. for Scenario 1
5	Repeat Steps 2-4 for Scenarios 2-8			Results from Steps 2-4

Test Category: Unit	**Description:** Test Simulation Console Traffic Scenario Selection properties are scalable depending on chosen region		
Test Case: 3.2.2.	**Case Name:** Sim Console Scenario Scale Test	**Version:** 1.0	**Written By:** Andrew Crossman
Requirements Fulfilled: 3.1.3.2.3.	**Purpose:** Verify that the Traffic Scenario Selection part of the Simulation Console has scalability functions to support the 3 region sizes		
Setup Conditions: • Source code for Simulation Console opened			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Open Scenarios.cs file			Source code for Scenario class opens
2	Locate RegionUpdate function in base Scenario class			Public RegionUpdate function present and inheritable
3	Locate Scenario1 class			
4	Locate inherited RegionUpdate function			Inheritied virtual RegionUpdate function present
5	Observe code within brackets under statement “if regionSize == small”			RegionUpdate function alters these variables when called: trafficVolume, congestionRate, blockageRate, arrivalRate
6	Repeat Step 5 for medium and large regionSize statements			Results from Step 5, for specific region
7	Repeat Steps 3-6 for Scenarios 2-8			Results from Steps 3-6

Test Category: Integration	**Description:** Driver generation algorithm			
Test Case: 3.3.1	**Case Name:** Driver Generator	**Version:** 1.0	**Written By:** Andrew McKnight	
Requirements Fulfilled: 3.1.3.3.1, 3.1.3.3.4, 3.1.3.3.5	**Purpose:** Ensure that realistic proportions of drivers and users are generated, conforming to variable thresholds which can be changed by the user			
Setup Conditions: • Tester has console window open with access to an executable version of the algorithm				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Enter command to run the executable code including parameters specifying type of distribution, associated initial values for desired distribution, driver-user ratio, array of entry points, and simulation time lapse to run algorithm for			Execution runs for specified amount of simulation time
2	Assert that driver-user ratio is nearly equal			Driver-user ratio should be within 10% of specified ratio
3	Assert that driver generation volume is nearly equal across entry points			Each entry point should be no more than 1 standard deviation from a normal distribution of individual volumes
4	Assert that total number of drivers is appropriate for the region size and time lapse			Total volume of generated drivers must be within 1 standard deviation for specified distribution and parameters
5	Assert that order of insertion between entry points is interleaved enough			No more than .2 standard deviations worth of drivers may be generated from the same entry point uninterrupted by another entry point generation
6	Assert that order of destination points of generated drivers is interleaved enough			No more than .2 standard deviations worth of drivers may be generated with the same destination consecutively

Test Category: Integration	**Description:** Test Simulation Console Runtime Execution basic functionality in terms of defaults and execution		
Test Case: 3.4.1.	**Case Name:** Sim Console Runtime Defaults and Selections Test	**Version:** 1.0	**Written By:** Andrew Crossman
Requirements Fulfilled: 3.1.3.1.3. , 3.1.3.2.4. , 3.1.3.3.2. , 3.1.3.4.1. , 3.1.3.4.3. , 3.1.3.4.4. , 3.1.3.5.1. , 3.1.3.5.3. , 3.1.3.5.5. , 3.1.4.2.6.1. , 3.1.4.2.6.2. , 3.1.4.2.7.2. , 3.1.4.2.7.3. , 3.1.4.2.7.4. , 3.1.4.2.7.5.	**Purpose:** Verify that the Simulation Console requires a region, traffic scenario, and Traffic Wizard driver percentage be chosen before a simulation can be executed. Verify that all regions, scenarios, and percentages can be selected from Dashboard. Verify that Dashboard controls are functional as intended.		
Setup Conditions: • Simulation Console program is loaded for operation • Traffic Simulation window is launched			

Test Case Activity		Pass/Fail	Comments	Expected Result
1	Extend Dashboard from Traffic Simulation window			Dashboard extends
2	View Region Size drop-down box			*Small* region selected by default. Map for Small region already displayed (entire small_region map from case 3.1.1.)
3	Select *Medium* from Region Size drop-down box			Map for Medium region displayed (entire medium_region map from case 3.1.1.)
4	Select *Large* from Region Size drop-down box			Map for Large region displayed (entire large_region map from case 3.1.1.)
5	View Scenario drop-down box			*Scenario 1* selected by default
6	Select *Scenario 2* from Scenario drop-down box			*Scenario 2* selected
7	Repeat Step 6 for *Scenario 3* through *Scenario 8*			Result from Step 6

8	View Percentage TW Users drop-down box			*0%* selected by default
9	Select 1*0%* from Percentage TW Users drop-down box			*10%* selected
10	Repeat Step 9 for Percentage TW Users *20%* through *90%*			Result from Step 9
11	Set options to smallest case: Small region, Scenario 1, 0%			Options shown as selected in drop-down boxes
12	Click *Play* on the Dashboard			Simulation begins executing (virtual driver entities appear on map)
13	Let simulation run for 5 minutes			Virtual driver objects animate on map and do not overlap each other
14	Click *Pause* on the Dashboard			Simulation pauses in current state (virtual driver entities freeze animation on map)
15	Click *Play* on the Dashboard			Simulation resumes execution (virtual drivers continue animation)
16	Let simulation run for 15 minutes			Simulation ends execution at 15 minute mark (virtual driver entities freeze animation on map)
17	Click *Stop* on the Dashboard			Virtual driver entities disappear from map

<table>
<tr><td><b>Test Category:</b>
System</td><td colspan="3"><b>Description:</b> Test Simulation Console Runtime Execution for Scenario 1 to prove particular algorithms/performance for that scenario</td></tr>
<tr><td><b>Test Case:</b>
3.4.2.</td><td><b>Case Name:</b> Sim Console Scenario 1 Execution Test</td><td><b>Version:</b> 1.0</td><td><b>Written By:</b>
Andrew Crossman</td></tr>
<tr><td><b>Requirements Fulfilled:</b> 3.1.2.1.3. , 3.1.2.1.7. , 3.1.2.1.8. , 3.1.2.3.2. , 3.1.2.3.5. , 3.1.2.3.6. , 3.1.2.3.9. , 3.1.2.3.10 , 3.1.2.6.2. , 3.1.3.3.3. , 3.1.3.3.6. , 3.1.3.4.2. , 3.1.3.5.2. , 3.1.3.5.3. , 3.1.3.5.5. , 3.1.4.2.6.3. , 3.1.4.2.7.2.</td><td colspan="3"><b>Purpose:</b> Verify that the Simulation Console can execute a simulation of Scenario 1 that can show results of necessary algorithms and perform as expected. This test case is purposed at demonstrating a scenario with low congestion and a rare chance for blockages. This test case acts as the foundation for test cases 3.4.3. – 3.4.x, which run the other scenarios.</td></tr>
<tr><td colspan="4"><b>Setup Conditions:</b>
• Simulation Console program is loaded for operation
• Traffic Simulation window is launched
• Region Selection test case 3.1.1. passed (supported regions)</td></tr>
</table>

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Extend Dashboard from Traffic Simulation window			Dashboard extends
2	Select *Small* from Region Size drop-down box			Map for Small region displayed (small_region map from case 3.1.1.)
3	Select *Scenario 1* from Scenario drop-down box			
4	Select *70%* from Percentage TW Users drop-down box			
5	Click the *Debug* button on the Dashboard			Debug window appears beneath map with no initial text
6	Click *Play* on the Dashboard			Simulation begins executing (virtual driver entities appear on map)
7	Click *Pause* on the Dashboard after simulation begins			Simulation activity freezes in current state
8	Notate current status of properties within region: number of checkpoints, status of checkpoints, current			*Take screenshot of initial status if necessary

	blockages			
9	Click *Play* on the Dashboard to resume simulation			Simulation activity resumes (virtual driver entities continue animation)
10	Let the simulation run for 5 minutes			Virtual driver entities animate across the roads on the region map as simulation time advances. Virtual checkpoints change traffic status and re-allocate. Debug window displays internal exchanges of information.
11	Click *Pause* on the Dashboard			Simulation activity freezes in current state. Debug window stops reporting.
12	Observe reported lines in Debug window			At least one instance of: - A virtual checkpoint receives speed and time input from a virtual driver - A new speed and update time is returned to a virtual checkpoint to change status - Re-allocation of checkpoints occurs to reflect lessened traffic congestion (checkpoints spread apart more, report that database is updated with new locations) - Virtual checkpoint de-activated (greyed out) due to lack of input
13	Click *Stop* on the Dashboard			Simulation ends execution (virtual driver entities disappear from map). Debug window clears text.

Test Category: System	**Description:** Test Simulation Console Runtime Execution for Scenario 8 to prove particular algorithms/performance for that scenario		
Test Case: 3.4.3.	**Case Name:** Sim Console Scenario 8 Execution Test	**Version:** 1.0	**Written By:** Andrew Crossman
Requirements Fulfilled: 3.1.2.1.3. , 3.1.2.1.7. , 3.1.2.1.8. , 3.1.2.3.2. , 3.1.2.3.5. , 3.1.2.3.6. , 3.1.2.3.10 , 3.1.2.6.2. , 3.1.2.6.7. , 3.1.3.3.3. , 3.1.3.3.6. , 3.1.3.4.2. , 3.1.3.5.2. , 3.1.3.5.3. , 3.1.3.5.4. , 3.1.3.5.5. , 3.1.4.2.6.3. , 3.1.4.2.7.2.	**Purpose:** Verify that the Simulation Console can execute a simulation of Scenario 8 that can show results of necessary algorithms and perform as expected. This test case is purposed at demonstrating a scenario with much congestion and a high chance for blockages. This test case uses test case 3.4.2. as a basis for proving other algorithms first.		

Setup Conditions:

- Simulation Console program is loaded for operation
- Traffic Simulation window is launched
- Region Selection test case 3.1.1. passed (supported regions)
- Runtime Execution test case 3.4.3. passed (system test)

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Extend Dashboard from Traffic Simulation window			Dashboard extends
2	Select *Small* from Region Size drop-down box			Map for Small region displayed (small_region map from case 3.1.1.)
3	Select *Scenario 8* from Scenario drop-down box			
4	Select *20%* from Percentage TW Users drop-down box			
5	Click the *Debug* button on the Dashboard			Debug window appears beneath map with no initial text
6	Click *Play* on the Dashboard			Simulation begins executing (virtual driver entities appear on map)
7	Click *Pause* on the Dashboard after simulation begins			Simulation activity freezes in current state
8	Notate current status of properties within region:			*Take screenshot of initial status if necessary

	number of checkpoints, status of checkpoints, current blockages			
9	Click *Play* on the Dashboard to resume simulation			Simulation activity resumes (virtual driver entities continue animation)
10	Let the simulation run for 5 minutes			Virtual driver entities animate across the roads on the region map as simulation time advances. Virtual checkpoints change traffic status and re-allocate. Debug window displays internal exchanges of information.
11	Click *Pause* on the Dashboard			Simulation activity freezes in current state. Debug window stops reporting.
12	Observe reported lines in Debug window			At least one instance of: - A virtual checkpoint receives speed and time input from a virtual driver - A new speed and update time is returned to a virtual checkpoint to change status - Re-allocation of checkpoints occurs to reflect increased traffic congestion (checkpoints moved closer together, report that database is updated with new locations) - A new checkpoint is added during re-allocation due to increased traffic congestion - A blockage is reported at some location and displayed on the map as a red rectangle
13	Click *Stop* on the Dashboard			Simulation ends execution (virtual driver entities disappear from map). Debug window clears text.

Test Category: Unit	**Description:** Test Simulation Console Runtime Execution to display and distinguish normal virtual drivers from virtual drivers using Traffic Wizard		
Test Case: 3.4.4.	**Case Name:** Sim Console Virtual Driver Type Test	**Version:** 1.0	**Written By:** Andrew Crossman
Requirements Fulfilled: 3.1.3.3.3. , 3.1.3.3.6. , 3.1.3.4.2. , 3.1.3.4.5. , 3.1.3.5.2. , 3.1.3.5.3. , 3.1.3.5.5. , 3.1.4.2.7.2.	**Purpose:** Verify that the Simulation Console can generate two types of virtual drivers: normal drivers (without the ability to learn of traffic conditions), and TW users (with the ability to learn of conditions and re-route if necessary)		
Setup Conditions: • Simulation Console program is loaded for operation • Traffic Simulation window is launched			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Extend Dashboard from Traffic Simulation window			Dashboard extends
2	Select *50%* from Percentage TW Users drop-down box			
3	Click *Play* on the Dashboard			Simulation begins execution (virtual driver entities appear on map)
4	Let the simulation run for 5 minutes			Virtual driver entities animate across the roads on the region map as simulation time advances.
5	Click *Pause* on the Dashboard			Simulation activity freezes in current state
6	Observe Traffic Simulation window in paused state			Two different colors of virtual drivers present on the map (white is normal, blue is a TW user)
7	Click *Stop* on the Dashboard			Simulation ends execution (virtual driver entities disappear from map)

<table>
<tr><td><b>Test Category:</b> Integration</td><td colspan="3"><b>Description:</b> User login credential checking</td></tr>
<tr><td><b>Test Case:</b> 4.1.1</td><td><b>Case Name:</b> Login</td><td><b>Version:</b> 1.0</td><td><b>Written By:</b> Andrew McKnight</td></tr>
<tr><td><b>Requirements Fulfilled:</b> 3.1.4.1.1.1, 3.1.4.1.1.2, 3.1.4.1.1.3, 3.1.4.1.1.4</td><td colspan="3"><b>Purpose:</b> Ensure that only authorized users are able to access the main user interface functionality of the application</td></tr>
<tr><td colspan="4"><b>Setup Conditions:</b>
• Simulation Console is running
• Smartphone application opened
• Cellular signal is present
• “Login” button is disabled</td></tr>
</table>

	Test Case Activity	Pass/Fail	Comments	Expected Result
1	Input username in username field and password in the password field, both with invalid characters			Login button remains disabled due to invalid input; message appears describing error
2	Change user/pass inputs to valid inputs but invalid credentials			Login button is enabled
3	Click login button			Access is denied; message appears describing error
4	Change user/pass to completely valid credentials			Access is granted and user is taken to main screen

Test Category: Unit	**Description:** New Trip Creation process evaluation		
Test Case: 4.2.1	**Case Name:** New Trip	**Version:** 1.0	**Written By:** Andrew McKnight
Requirements Fulfilled: 3.1.4.1.2.1 – 3.1.4.1.2.7	**Purpose:** Ensure the process of New Trip Creation runs correctly or fails gracefully		

Setup Conditions:

- Smartphone application opened
- Cellular signal is present
- Login attempt successfully completed
- New Trip button pressed on main screen
- Next button is disabled

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Type name of existing route (case insensitive) and arbitrary addresses in starting and ending address fields			“Next” button remains disabled; Error message is displayed
2	Type name not already assigned to other trip on smartphone			Next button becomes enabled
3	Touch “Next” button			Screen advances to the departure time picker, which is initialized to the current time; “Next” button is enabled
4	Touch “Next” button			Screen advances to Notification Method screen; all options are initially selected; “Next” button is enabled
5	Switch all options off and back on			Operation should proceed as expected- sliders move to off positions and back to on positions
6	Touch “Next” button			Screen advances to Primary Route Screen; Error message shows if error returned from Google Geocoding API service, otherwise all possible routes are listed and overlaid on map; “Finish” button is disabled
7	Touch all route list entries one by one			Corresponding route overlay is redrawn in bold blue lines

8	Touch “Finish” button			Screen advances to main screen
9	Touch “Current Trips” button on main screen			Screen advances to list of current trips; newly created trip should be last on the list
10	Touch newly created trip			Screen advances to trip detail screen; primary route overlaid in bold blue, other routes in thin red lines; other details match input values in earlier steps

Test Category: Unit	**Description:** Tests the Route Tracer functionality.		
Test Case: 4.3.1	**Case Name:** Route Tracer Operation	**Version:** 1.0	**Written By:** Andrew McKnight
Requirements Fulfilled: 3.1.4.1.3.1 – 3.1.4.1.3.3	**Purpose:** Test the functionality of the Route Tracer screen to ensure that illegal start/stop presses are prevented		

Setup Conditions:

- User must have logged in.
- User must have begun new trip creation (1) –OR-
- User must have navigated to route tracer from main screen (2)
- GPS signal must be present
- “Stop” button is disabled

Test Case Activity		Pass/Fail	Comments	Expected Result
1	Attempt to press stop button.			It is disabled so nothing happens.
2	Press the start button.			Execution of Route Tracer algorithm commences. Start button becomes disabled.
3	Press start button an arbitrary amount of times after first occasion.			Execution of RouteTracer continues unaffected.
4	Press stop button.			Execution ceases and the location data is transmitted to the server. Screen advances to either “Route Tracer Finished” screen (if setup condition (2) is fulfilled) or to the next step in new trip creation (if setup condition (1) is fulfilled)

Test Category: Unit	**Description:** Edit Trip process evaluation		
Test Case: 4.4.1	**Case Name:** New Trip	**Version:** 1.0	**Written By:** Andrew McKnight
Requirements Fulfilled: 3.1.4.1.4.1, 3.1.4.1.4.2	**Purpose:** Ensure the process of editing a Trip runs correctly or fails gracefully		
Setup Conditions: • Smartphone application opened • Cellular signal is present • Login attempt successfully completed • At least one trip has been previously created • Edit Trip button pressed on main screen			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Select arbitrary trip from list of existing trips			Screen advances to trip detail screen
2	Touch "Edit" button on bottom of screen			Screen advances to screen identical to first screen of new trip creation
3	Run through test case 4.2.1, changing at least one data point on each screen			All tests pass normally
4	Touch Edit Trip button from main screen			Screen advances to list of current trips
5	Touch list item corresponding to the edited trip			Screen advances to trip detail screen. All changed details are reflected in the information displayed

Test Category: Unit	**Description:** End of Trip process evaluation		
Test Case: 4.5.1	**Case Name:** End of Trip	**Version:** 1.0	**Written By:** Andrew McKnight
Requirements Fulfilled: 3.1.4.1.5.1, 3.1.4.1.5.2	**Purpose:** Ensure the End of Trip process of runs correctly and unobtrusively to the user		

Setup Conditions:

- Simulation Console is running and has socket connection to smartphone app
- Smartphone application opened
- Cellular signal is present
- Smartphone has received trip object and drive vector from simulation console
- Smartphone is in drive mode and has passed the last checkpoint
- App view changes to End Trip Screen
- Audible message is played describing the end of the trip

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Touch edit trip			Screen advances to start of trip creation/edit series
2	Touch “Back” button			Screen reverts to end trip screen
3	Touch “done” button			Screen advances to main screen

Test Category: Unit	**Description:** Delay notification process evaluation		
Test Case: 4.6.1	**Case Name:** New Trip	**Version:** 1.0	**Written By:** Andrew McKnight
Requirements Fulfilled: 3.1.4.1.6.1 – 3.1.4.1.6.4	**Purpose:** Ensure the delay notification process runs correctly and unobtrusively to the driver		
Setup Conditions: • Simulation Console is running and connected to smartphone app through socket • Smartphone application opened • Cellular signal is present • Login attempt successfully completed • New Trip button pressed on main screen • Next button is disabled			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Send Alert object from console to smartphone			• If application is running, o If app is in still mode, advances screen to delay notification screen o If app is in drive mode, also plays audible alert • Otherwise, alerts are sent via text/email/push notification as specified by the test
2	Assert that time is not negative number and all other information is correct as compared to trip object			All info is identical between delay notification screen and simulation console state and trip object

Test Category: Unit	**Description:** Test Simulation Console interface Main Menu to ensure that all features are accessible		
Test Case: 5.1.1.	**Case Name:** Sim Console GUI Main Menu Test	**Version:** 1.0	**Written By:** Andrew Crossman
Requirements Fulfilled: 3.1.4.2.1.1. – 3.1.4.2.1.4.	**Purpose:** Verify that the Simulation Console Main Menu interface has accessible buttons/tabs for every feature of the Simulation Console and that they access the appropriate window		
Setup Conditions: • Simulation Console program is loaded for operation			

	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	Visually inspect Simulation Console Main Menu			Traffic Wizard logo is displayed. Buttons for each of four features is displayed (as well as an Exit button): - Driver Profile Demo - Route Create/Edit Demo - Route Tracer Demo - Traffic Simulation
2	Click *Driver Profile Demo* button			Driver Profile Demo window opens
3	Click *Back* to return to Main Menu			Main Menu is displayed as before
4	Repeat Steps 2-3 for *Route Create/Edit Demo, Route Tracer Demo,* and *Traffic Simulation*			Result from Step 2 for respective window
5	Click *Exit* button			Simulation Console program closes

Test Category: Unit	**Description:** Driver Profile Demonstration		
Test Case: 5.2.1	**Case Name:** Driver Profile Database	**Version:** 1.0	**Written By:** Thomas Kennedy
Requirements Fulfilled: 3.1.4.2.3.1	**Purpose:** Verify that features of Driver Profiles have been implemented correctly		

Setup Conditions:

- Test 1.1.1 has been passed

	Test Case Activity	Pass/Fail	Comments	Expected Result
1	Open a connection to the Driver Profile Database			A connection has been opened to the database.
2	Query the Drive Profile Database for all tuples in all tables.			All rows from the database have been returned.
3	Visually Verify that all table entries have been returned.			All entries have been returned.

Test Category: Unit	**Description:** Driver Profile Demonstration		
Test Case: 5.2.2	**Case Name:** Driver Profile Screenshots	**Version:** 1.0	**Written By:** Thomas Kennedy
Requirements Fulfilled: 3.1.4.2.3.2	**Purpose:** Verify that features of Driver Profile Demonstration utilizes appropriate GUI screenshots		

Setup Conditions:

- Traffic Wizard smartphone application GUI screenshots are available

	Test Case Activity	Pass/Fail	Comments	Expected Result
1	Open the Simulation Console and navigate to the Driver Profile Demo.			The Driver Profile Demo is on screen.
2	Visually inspect the GUI screenshots on the page and compare to the smartphone screenshots.			The GUI screens match.

Test Category: Unit	**Description:** Driver Profile Demonstration		
Test Case: 5.2.3	**Case Name:** Driver Profile Main Menu	**Version:** 1.0	**Written By:** Thomas Kennedy
Requirements Fulfilled: 3.1.4.2.3.2	**Purpose:** Verify that features of Driver Profile Demonstration allows access to the main menu		
Setup Conditions: • Tests 5.3.1 has been passed			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Open the Simulation Console and navigate to the Driver Profile Demo.			The Driver Profile Demo is on screen.
2	Click the Main Menu Button			The Main Menu is displayed.

Test Category: Unit	**Description:** Must describe all fields required for creating a new route manually as outlined in Requirement 3.1.4.1.3.		
Test Case: 5.3.1	**Case Name:** Create/Edit	**Version:** 1	**Written By:** Binh Dong
Requirements Fulfilled: 3.1.4.2.4.1	**Purpose:** To ensure the functionality of the Route Create / Edit portion of the Simulation Console.		
Setup Conditions: • Simulation Console			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Open “Route Create / Edit Demo”			Route Create / Edit GUI loads.
2	Start creating a route			Route creation GUI loads.
3	Create a route			User inputs a route
4	Save a route			Route saves.
5	Load a route			To ensure if the saved route was saved.
6	Edit route			User edits previously saved route.
7	Repeat steps 4-5			To ensure if edited route saved.

Test Category: Unit	**Description:** Must use smartphone app GUI from Requirement 3.1.4.1 as foundation for images.		
Test Case: 5.3.2	**Case Name:** Simple	**Version:** 1	**Written By:** Binh Dong
Requirements Fulfilled: 3.1.4.2.4.2	**Purpose:** Route Create / Edit GUI must be intuitive, robust and non-distracting.		
Setup Conditions: • Need Source Code			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Visual Check			GUI should not be distracting. GUI should conform to Requirement 3.1.4.1.

Test Category: Unit	**Description:** Must be able to return to main Menu at any time.		
Test Case: 5.3.3	**Case Name:** Anytime Main Menu	**Version:** 1	**Written By:** Binh Dong
Requirements Fulfilled: 3.1.4.2.4.3	**Purpose:** To check the ability to return to the main menu at any time.		
Setup Conditions: • Need Source Code			

	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	Press main menu			GUI should load the main menu. This must happen any time.

Test Category: Integration	**Description:** Route Tracer demo		
Test Case: 5.4.1	**Case Name:** Route Tracer demo	**Version:** 1.0	**Written By:** Andrew McKnight
Requirements Fulfilled: 3.1.4.2.5.1 – 3.1.4.2.5.3	**Purpose:** Show the functionality of the Route Tracer works as expected and returns correct results		
Setup Conditions: • Simulation console is running • Smartphone application is opened and tester has logged in and selected Route Tracer from the main screen			

	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	See Test case 4.3.1 for Route Traeer usage and testing			All steps in test case 4.3.1 pass
2	Inspect data sent to server for route matching			Data should accurately describe all locations, speeds, and headings along routes at points where readings were taken
3	Press button to go to main screen			Simulation console returns to main screen

Test Category: Unit	**Description:** Test Simulation Console interface Dashboard to ensure that it becomes visible when extended		
Test Case: 5.6.1.	**Case Name:** Sim Console GUI Dashboard Access Test	**Version:** 1.0	**Written By:** Andrew Crossman
Requirements Fulfilled: 3.1.4.2.7.1.	**Purpose:** Verify that the Simulation Console Dashboard is accessible from the Traffic Simulation window and that it is visible when extended for controls		
Setup Conditions: • Simulation Console program is loaded for operation • Traffic Simulation window is launched			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Extend Dashboard from Traffic Simulation window (click on down arrow)			Dashboard extends and overlaps part of the top of the currently displayed region map
2	Collapse Dashboard (click on up arrow)			Dashboard collapses to the top of the region map – only visible as a bar with down arrow to be extended again

Test Category: Unit	**Description:** Test Simulation Console interface Dashboard to ensure that a simulation has to be stopped before returning to the Main Menu		
Test Case: 5.6.2.	**Case Name:** Sim Console GUI Dashboard Return Test	**Version:** 1.0	**Written By:** Andrew Crossman
Requirements Fulfilled: 3.1.4.2.7.6 , 3.1.4.2.7.7.	**Purpose:** Verify that the Simulation Console is unable to return to the Main Menu when a simulation is running and that it is able to return when there is no simulation running		
Setup Conditions: • Simulation Console program is loaded for operation • Traffic Simulation window is launched			

	Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**
1	Extend Dashboard from Traffic Simulation window			Dashboard extends
2	Click *Play* on the Dashboard			Simulation begins execution (virtual driver entities appear on the map)
3	Attempt to click *Back* to return to the Main Menu			Button is greyed out. Window does not exit Traffic Simulation and simulation continues execution
4	Click *Pause* on the Dashboard			Simulation activity freezes in current state
5	Attempt to click *Back* to return to the Main Menu			Button is greyed out. Window does not exit Traffic Simulation and simulation remains paused
6	Click *Stop* on the Dashboard			Simulation ends execution (virtual driver entities disappear from map)
7	Click *Back* to return to the Main Menu			Main Menu window appears

6 Traceability Requirements

The Traceability Matrix shows the relationship between the test cases and the requirements covered by each. Each requirement has at least one corresponding test case. The matrix can be found at http://cs.odu.edu/~411blue/?page=collaboration#lab3