Better Than Waze

Test Plan

Submitted by:

Adam Elgressy

Anat Tetroashvili

Guy Rephaeli

Orel Kanditan

Shay Michaeli

Sharon Hadar

Supervised by:

Tanya Brokhman-Linder

Table of Contents

1. Introduction

1.1 Project overview

Better Then Waze (BTW) is a geographic system project. BTW designed to help planning an efficient roads system, and to find the best paths between locations in the system.
To achieve these goals, BTW is giving the user the ability to:

a. **Simulate a city** – a real roads system with junctions, streets and locations. The user will give the specification, and BTW will simulate the desired city.
b. **Pick the fastest way between two locations**. By keeping information about heavy traffic for each junction and road – BTW will know how to find the best path between two points and display the directions to the user.

All cities are first represented through the Geo-Json file format, enabling a uniform yet robust representation of the city and traffic information. The input\simulated city is then stored in SQL DB, from which information can be easily extracted when needed.
Using the information stored in the DB, a city can be represented as a very detailed graph, on which graph algorithms can be applied to navigate efficiently.

2. Test strategy

2.1 Test objectives

The objective of the test is to verify that the functionality of BTW works according to the specifications. The final product of the test is twofold:

- A production-ready software;
- A set of stable test scripts that can be reused for Functional test execution.

2.2 Test assumptions

The unit tests will assume each module is an independent piece of software and test them accordingly. The integration tests will assume each module works independently and will assume we have a working internet connection for database communication.

2.3 Scope and Levels of Testing

User level: check all interfaces are correctly represented. Check all desired paths are received

Map display representation level: check all maps are correctly represented and sent to be displayed. Check a calculated path is also represented correctly on the map. Check the possibility to update the display when there is a change in the database content.

Data base level: check all the data is consistent with the map input, and with the current state of the roads. Check the possibility to update the information in the data base.

Logic level: check all the calculated paths are correct, check all the information collected from the data base is represented in a sufficient way in java. Also check the possibility to update this representation over time.

2.3.1 Functional Test

PURPOSE: Functional testing will be performed to check the functions of application. The functional testing is carried out by feeding the input and validates the output from the application.

METHOD: The test will be performed according to Functional scripts/Test procedures with well-defined PASS/FAIL criteria.

TestRandomizedMap: check a random map can be created by demand. check it can calculate several random paths on it. Check that information can be added and removed from it. Try to display it on the screen. If all previous actions can be performed, then return SUCCESS otherwise return fail.

TestGivenMap: check a given map can be readed from the input. check it can calculate several random paths on it. Check that information can be added and removed from it. Try to despaly it on the screen. If all previous actions can be performed, then return SUCCESS otherwise return fail.

TestDataBaseConnection: check that a connection to the database can be established. If yes, return a SUCCESS, else return FAIL.

TestDataBaseQuerryCorrectness: try query several simple examples from a small and well known database. Check if the results are consistent with the small data base content. The test receives as an input the final state expected from the data base and compare the test results to it. If consistency exists, return SUCCESS, else return FAIL.

TestDataBaseUpdateCorrectness: try update several predefined examples from a small and well known database. Try to make the result look like a well known expected transform. The test receives as an input the final state expected from the data base after the transformation and compare the test results to it. If consistency exists, return SUCCESS, else return FAIL.

TestGraphCreationForLogic: get as input a predefined small database, and make a graph representation of the data structures in java. Print the data structures created and check the consistency to the input. If consistency exists, return SUCCESS, else return FAIL.

TestLogicFindRightPaths: get as input a graph representation in java of a small well known database. Try to find predefined shortest paths by sing the graph representation. If found all paths correctly return SUCCESS, else return FAIL.

2.3.2 User Acceptance Test (UAT)

PURPOSE: this test allows the end users to complete one final review of the system prior to deployment. The user will enter a well-known map (of his own city) and will search a well-known path (for example from his home to work) but with an alternative path existing on the map.to ensure only the correct rout is available; the user will give all the road on the map not on the desired path a higher traffic load factor. Then he will check if the rout returned from the system is the correct one, which means the shortest.

Another test will be performed with the above conditions, but now the well-known path will get a high traffic load factor, and the rest of the map will get a significantly lower traffic load. Here we want to receive the alternative path, and not the shortest.

After setting the correct traffic load for each test, more tests will be to take off manually some roads from the shortest and the alternative paths on the map and see if the received path is changed according to the roads taken. The possibilities are:

- When the traffic load on the shortest path is lowest, erase from the map a road from this path and see if the new path calculated is making sense, or even that the new path is not considering the road that was taken.
- When the traffic load on the shortest path is lowest, erase from the map a road not from this path, and see that the result stay the same.
- When the traffic load on the shortest path is highest, erase from the map a road from this path and see if the new path calculates is still the alternative path.
- When the traffic load on the shortest path is highest, erase from the map a road from the alternative path and see that the received path is changed' or cannot be calculated.

One more test is to see it recognize when the source and destination are the same. Should just say the user has arrived. An error will be a path the require the user to move from his location.

Check all the buttons provided by the main menu can be pressed and lead to the right menu. After that check that every menu can resume the main menu by the "resume" button.

METHOD: Will be performed manually by team members according to written test cases.

3. Validation and Defect Management

It is the responsibility of the tester to open the defects, link them to the corresponding script, assign an initial severity and status,

It is the responsibility of the developer to retest after a fix is provided and close the defect.

Defects will be categorized according to the following severity status:

Severity	Impact
1 (Critical)	• This bug is critical enough to crash the system, cause file corruption, or cause potential data loss • It causes an abnormal return to the operating system (crash or a system failure message appears). • It causes the application to hang and requires re-booting the system.
2 (High)	• It causes a lack of vital program functionality with workaround.
3 (Medium)	• This Bug will degrade the quality of the System. However there is an intelligent workaround for achieving the desired functionality - for example through another screen. • This bug prevents other areas of the product from being tested. However other areas can be independently tested.
4 (Low)	• There is an insufficient or unclear error message, which has minimum impact on product use
5 (Cosmetic)	• There is an insufficient or unclear error message that has no impact on product use.

4. TEST ENVIRONMENT

The test running environment is a working computer connected to the internet (for approaching the database server), with enough memory to hold a current zoom of the map representation. The code will be first checked on the IntelliJ IDE using JUnit, and afterwards on the Technion server. Requires a virtual machine running at least java 8 and above.

5. Test cases

5.1 Function tests

5.1.1 Data-Base Tests:

Objective	**Enter:**	**Exit:**	**Defect Categorization:**
Insert Road information	Use SQL query to insert new road with legal values.	Road table affected, new tuple appears.	Critical – tuple isn't created, wrong values appear.
Insert Crossroads information	Use SQL query to insert new Crossroads with legal values.	Crossroads table affected, new tuple appears.	Critical – tuple isn't created, wrong values appear.
Insert Passageway information	Use SQL query to insert new Passageway with legal values.	Passageway table affected, new tuple appears.	Critical – tuple isn't created, wrong values appear.
Insert Weight information	Use SQL query to insert new Weight with legal values.	Weight table affected, new tuple appears.	Critical – tuple isn't created, wrong values appear.
Insert Place information	Use SQL query to insert new Place with legal values.	Place table affected, new tuple appears.	Critical – tuple isn't created, wrong values appear.
Can’t create illegal Crossroads	1. Use SQL query to insert new crossroads. 2. Supply illegal	Crossroads table isn’t affected.	Critical – Old data table affected. High – tuple is

	Passageways id.		added to the table.
Can't create illegal Passageway	1. Use SQL query to insert new Passageway. 2. Supply illegal crosses roads id.	Passageway table isn't affected.	Critical – Old data table affected. High – tuple is added to the table.
Can't create illegal Place	1. Use SQL query to insert new Place. 2. Supply illegal road id.	Place table isn't affected.	Critical – Old data table affected. High – tuple is added to the table.
Retrieve Road Information	Use SQL query to get road data by road id.	Road table isn't affected. Correct information received.	Critical – Old data table affected. High – received wrong information.
Retrieve Crossroads Information	Use SQL query to get Crossroads data by id.	Crossroads table isn't affected. Correct information received.	Critical – Old data table affected. High – received wrong information.
Retrieve Passageway Information	Use SQL query to get Passageway data by id.	Passageway table isn't affected. Correct information received.	Critical – Old data table affected. High – received wrong information.
Retrieve Place Information	Use SQL query to get Place data by Place id.	Place table isn't affected. Correct information received.	Critical – Old data table affected. High – received wrong information.

Can't get information with wrong id	Use SQL query to receive road information, with unknown id.	Road table isn't affected. Error message – unknown road.	Critical – Old data table affected. High – received wrong information.
DB knows how to get information from JSON	1. Create legal json file with system information. 2. Use DB functionality to keep the information from the json file.	The json file is recognized correctly, all data spread and inserted into the correct tables by columns.	Critical – Old data table affected. High – received wrong information.
Get maps from DB	Use SQL query to receive map information by map id.	The correct json file describing the map id received.	Critical – Old data table affected. High – received wrong information.

5.1.2 Location Tests:

Objective	**Enter:**	**Exit:**	**Defect Categorization:**
Create Graph	All graph information from the DB.	A graph that represents the city correctly.	Critical – graph isn't created; created graph does not represent the city correctly.
Find Coordinates	Enter coordinates to get the crossroad corresponding to them.	Crossroad object located in the specified coordinates.	Critical – object isn't returned, wrong object is returned.
Do Not Find Illegal Coordinates	Enter Illegal coordinates.	Error message, specifying the illegal coordinates	Critical – Crossroad object is returned.
Find Location Name	Enter location name to	Road object	Critical – object

	get the road corresponding to it.	containing the specified location.	isn't returned, wrong object is returned.
Do Not Find Illegal Location	Enter nonexisting location name.	Error message, specifying the illegal location name.	Critical – Road object is returned.
Find Address	Enter address to get the road corresponding to it.	Road object containing the specified address.	Critical – object isn't returned, wrong object is returned.
Do Not Find Illegal address	Enter nonexisting address.	Error message, specifying the illegal address.	Critical – Road object is returned.

5.1.3 Navigation Tests:

Objective	**Enter:**	**Exit:**	**Defect Categorization:**
Create Route	Start and finish points.	A legit route between the two points.	Critical – route is not returned; output route is invalid Medium – Route isn't optimal
Do not Create nonexisting Route	Start and finish points.	Error message specifying the start and finish points	Critical – output route is returned
Calculate route timing	Start and finish points.	A legit route between the two points, and the time it takes to pass the route.	High – estimated time is not returned; output route and time are not compatible.
Find greedy optimal route	Start and finish points.	A legit and short route between the two points, and the time it	High – there exists a point on the route, from

		takes to pass the route.	which the suggested route was not optimal.

5.1.4 City Generation Tests:

Objective	**Enter:**	**Exit:**	**Defect Categorization:**
Create default grid city	-	A functional 6x6 blocks grid city road map, with the length of the block borders are about 1 mile. No load on the streets	Critical – city is not returned. High –The city that returns doesn't comply with the default
Create grid city with number of streets parameter	Parameters to create the grid city, namely: - Number of streets	A functional grid city road map, according to the number of streets parameter No load on the streets	Critical – city is not returned. High –The city that returns doesn't comply with the parameters
Create grid city with number of avenues parameter	Parameters to create the grid city, namely: - Number of avenues	A functional grid city road map, according to the number of avenues parameter No load on the streets	Critical – city is not returned. High –The city that returns doesn't comply with the parameters
Create grid city with street length parameter	Parameters to create the grid city, namely: - Street length	A functional grid city road map, according to the street length parameter	Critical – city is not returned. High –The city that returns doesn't comply

		No load on the streets	with the parameters
Create grid city with avenue length parameter	Parameters to create the grid city, namely: - Avenue length	A functional grid city road map, according to the avenue length parameter No load on the streets	Critical – city is not returned. High –The city that returns doesn't comply with the parameters
Create random city with default parameters	-	A functional city road map, according to the default parameters of the random city generator, **NOT** the grid generator No load on the streets	Critical – city is not returned. High –The city that returns doesn't comply with the default parameters

5.1.5 Geo-Json Handling Tests:

Objective	**Enter:**	**Exit:**	**Defect Categorization:**
Create random map	Properties for the random map to include	GeoJson file generation	Critical- GeoJson file for random map is not created, illegal file is created
Do not create random map with illegal input	Illegal properties for random map	Error message, specifying the illegal input	Critical- GeoJson file for random map is created
Build data structures from a given GeoJson	GeoJson file representing	Complete data structures for the use	Critical- data structures

file, containing the properties of the given city	city	of the program	doesn't create, or creation of incorrect data structures
Do not Build data structures from a given GeoJson file, containing illegal properties of map	GeoJson file representing city with illegal properties	Error message, specifying the illegal property	Critical- creation of incorrect data structures

5.2 UAT Tests

5.2.1 Usability Tests:

Objective	**Enter:**	**Exit:**	**Defect Categorization:**
Input City Representation	Geo-Json file specifying a city	Graphic representation of the city.	Critical – no graphical representation is shown; graphic representation and input city are incompatible
City representation Error Handling	Incoherent Geo-Json file	Error message describing incoherencies.	Critical – the application gets stuck\shuts down; a graphical representation is shown
City Generation and representation	Parameters describing some desired city characteristics	Graphic representation of the generated city.	Critical – no graphical representation is shown; graphic representation and input parameters are incompatible

City Generation Error Handling	Incompatible city characteristics	Error message describing the incompatible parameters.	Critical – the application gets stuck\shuts down; a graphical representation is shown
City representation and navigation	Geo-Json file specifying a city, and start and end points for navigation	Graphic representation of the generated city and navigation route.	Critical – no graphical representation is shown; graphic representation and input parameters are incompatible
City representation and illegal navigation	Geo-Json file specifying a city, and illegal start and end points for navigation	Graphic representation of the generated city and error message specifying the illegal points for the navigation task.	Critical – no graphical representation is shown; a navigation route is shown; the application gets stuck\shuts down.

5.2.2 Performance Tests:

Objective	**Enter:**	**Exit:**	**Defect Categorization:**
City Creation Performance	Geo-Json file specifying a city	Graphic representation of the city	Medium – city creation takes more than an hour
City Generation Performance	Parameters describing some desired city characteristics	Graphic representation of the city	Medium – city generation takes more than a few hours
Navigation	Start and finish points	Graphic representation of the	Medium – route calculation takes

Performance	(city is already initialized)	navigation route in the city	more than a few minutes
Route Quality	Start and finish points (city is already initialized)	Graphic representation of the navigation route in the city	Medium – route takes much longer to drive than the optimal route