**Better Than Waze**

**High Level Design**

**Submitted by**: BTW group:

Adam Elgressy

Anat Tetroashvili

Guy Rephaeli

Orel Kanditan.

Shay Michaeli

Sharon Hadar

## Supervised by: Tanya Brokhman-Linder

**Table of Contents (need to change after finishing)**

[Supervised by: Tanya Brokhman-Linder 1](#_Toc498849502)

[Abstract 3](#_Toc498849503)

[1. Introduction 3](#_Toc498849504)

[1.1 General Project Description 3](#_Toc498849505)

[1.2 Programming Environment 3](#_Toc498849506)

[2. Theoretical Background 3](#_Toc498849507)

[2.1 City Simulation Theory 3](#_Toc498849508)

[2.2 Graph Theory 4](#_Toc498849509)

[3. Basic System Functionalities 4](#_Toc498849510)

[4. Software Implementation 5](#_Toc498849511)

[4.1 Modules 5](#_Toc498849512)

[4.1.1 UI Module 5](#_Toc498849513)

[4.1.2 Data processing and navigation logic Module (BTW Logic) 5](#_Toc498849514)

[4.1.2.1 City Simulation 6](#_Toc498849515)

[4.1.3 DAL and data management Module (Data Access Layer) 6](#_Toc498849516)

[4.2 Top-Level View 7](#_Toc498849517)

[4.3 Main Menu 8](#_Toc498849518)

[4.3.1 User Interface 8](#_Toc498849519)

[4.3.2 Features 8](#_Toc498849520)

[5. References 9](#_Toc498849521)

# 

# Abstract

Simulating a city is a useful tool for future city planning and construction. To fully utilize the power of such simulations, traffic and transportation systems must be considered, thus providing the city planner the ability to pre-plan the roads for traffic optimization.

In this project, we aim to provide a tool for city simulation, and a navigation system inside the city for the purpose of time optimized driving.

# Introduction

## General Project Description

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.

In order to achieve these goals, BTW is giving the user the ability to:

1. **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.
2. **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.

## Programming Environment

Intellij IDEA – JAVA framework.

SQL Server.

# Theoretical Background

## City Simulation Theory

In the city simulation, we will use procedural modeling of road systems, that way we will be able to achieve the complex and anarchist nature, of many existing cities. With this, the user will be able to provide variables to better suit his needs, with limiting the number of roads/crosses, and adding a maximum/minimum limits to road sizes.

There will also be an option the build a plain grid city with given number of streets and avenues. To model that kind of city we will use a simple matrix with the given parameters size, and model the city accordingly.

## Graph Theory

The navigation task is most easily solved when modeling the problem as finding the shortest path in a graph. The most efficient algorithm for our needs is Dijkstra’s algorithm and its even more efficient variants: The A\* and IDA\* algorithms.

The A\* and IDA\* algorithms efficiency is a product of the heuristics they utilize, allowing them to consider the remaining length of a path, and not only its first part when calculating. The IDA\* algorithm can even utilize the heuristics to avoid traversing the whole graph.

# Basic System Functionalities

The main functionality that the system gives is finding the shortest way, from one point to another, in a given/rendom map. The input map can depict a place that exists in the real world, or, it can be made up by the user. The length of the way will be determined on the basis of the length of the roads and the load on the roads that contained in it.

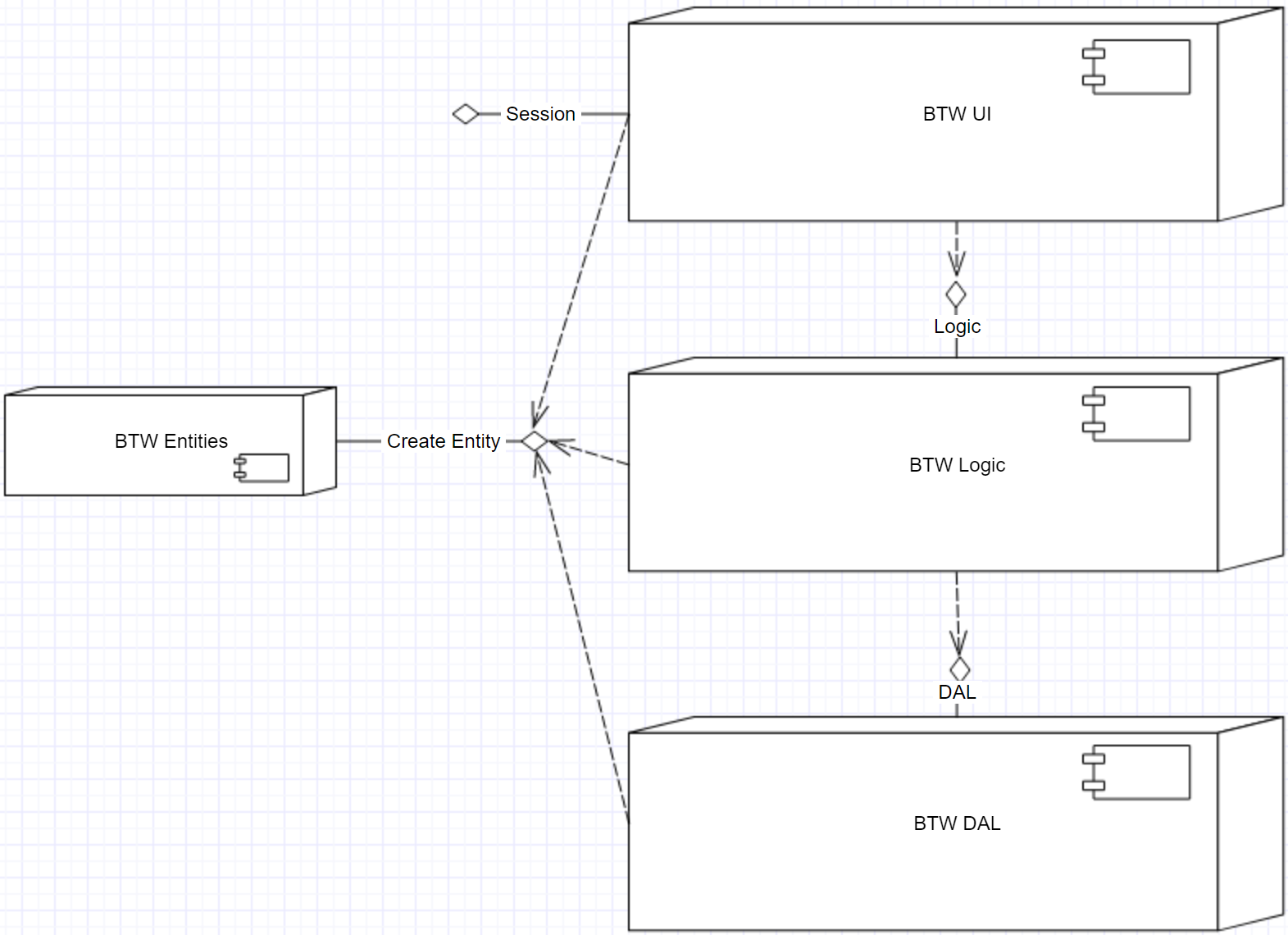
The system will produce graphical simulation of the inserted map.

The customer will be able to insert data on roads (number of roads, length of roads, number of intersections, etc.), and the system will create a random map that matches the characteristics.

The user will be able to re-access the maps he and other users have previously entered into the system, as well as update and modify them with new data.

# Software Implementation

## Modules



### UI Module

The user can provide a Geo-Json file, or choose to simulate a city. This module provides the user interface for these functionalities, and outputs information for the use of the other modules.

### Data processing and navigation logic Module (BTW Logic)

This module is responsible for conversions between the various formats (Java code, Geojson code and SQL lines), and algorithm logic. In this module, the input data processing will occur, then, the data will convert into the appropriate format for maintaining in the database. Moreover, this module includes the navigation algorithms and calculations made for the calculation of the shortest way.

### City Simulation

Part of the logic is responsible for simulating maps. Aside from using a "ready to use" city, we give the option to simulate a new road map through procedural generation of road maps, which are going to have more "messy" characteristics, like many European cities. while you will also be able to simulate a grid road map, like many American cities.

In a simulation, the user can give to the system many parameters, to better suit the simulated city to his needs, parameters such as: upper limits to number of roads/crosses, minimum\maximum desired length of a road.

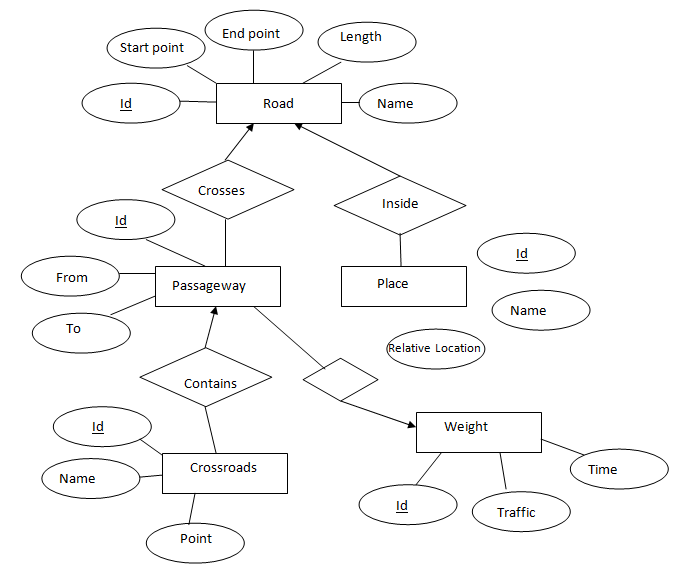
### DAL and data management Module (Data Access Layer)

To keep the system’s information, an SQL based database will be used.

The data is stored in SQL Server and contains information about roads, crossroads, traffic information, streets and places on the map.

Retrieving the data from the databases will be done using Transact-SQL.

DB Design:



## Top-Level View

The system architecture will be divided into four main tiers. This separation provides the ability to implement each part independently. The four tiers are:

**Client Tier:** This tier includes the user interface: windows, menus, buttons, presenting of maps, location and calculated paths. The interface will allow the user to ask for the quickest path for a chosen destination, from any source location on the map. The interface will also allow the user to enter some details about the map such as: roads, special locations, intersections, traffic loads and behavior, traffic lights, etc. This later added info will also for translation in the input tier.

**Input tier:** consists of three main parts:

* Map Input: a representation of an existing map will be inserted as an input. After processing the connections between streets, roads, intersections, central locations and traffic lights locations, an output file will be produced, in a form of a geojson file. The produced file will be sent to the data section for more processing and later, storing it on the server.
* Map Randomization: a non-existing map will be randomized to be used as our city. an implementation to map randomization will be provided by the project's team. The output will be sent for data processing and will also be a geojson file.
* User Input: all input added by the user after the first map initialization will be translated to geojson form, and the output will be sent to be updated at the data tier.

**Data Tier:** data received from the input tier in the form of geojson file will be processed and stored in an SQL server. The data will be stored in a way that simplified the connections among the roads, locations, intersections, traffic lights and the traffic behavior on the map. The data will be queried from the backend tier for paths calculations purposes. Data base allow to query for all roads, central locations, intersections, average hourly traffic loads on a road, traffic light timing and cyclicality, road's entrances and exits. In time, More data will be updated and added to server by the user or by the backend tier.

**Backend Tier:** in that section, we will do all the heavy calculations. finding the shortest path on the map between a source and a destination, considering the traffic loads, traffic rules and paths length. Changing traffic loads will be updated in the data tier.

## Main Menu

### User Interface

First of all, you will need to choose a map for this session of the system:

* Choose a file in your device that represent a map, which complies with the format the system uses.
* Use the city simulation option, which will give you the ability to choose the style of city: European style/grid city. In addition you will be able to set parameters as you see fit for the desired number of roads/crosses, and minimum/maximum length of road.

After the city is set, you will be able to give our system a starting position, time of departure, and destination. The system will show you on a map, the route which takes the least time to arrive to your destination.

### Features

* Simulating different traffic congestions level in existing cities infrastructure, to forecast the infrastructure needs of the future.
* Simulating new road system designs, to check levels of traffic congestion the design can hold, and how well does it hold the traffic.
* Navigating from point A, to point B, in the shortest time possible, considering various variables, such as: the average traffic during the trip's course, real-time information from other BTW users near the projected course, and traffic lights.

# References

* Map represenations, Stanford: <http://theory.stanford.edu/~amitp/GameProgramming/MapRepresentations.html>
* International Standard for geographic data files : <https://www.iso.org/standard/54610.html>
* Create amazing road maps:

<https://craft.io/?gclid=Cj0KCQjw4eXPBRCtARIsADvOjY1gVIbvvoizgq-bmOx0dlm6A3XXazXiErp_pU4tJehGua8lL7dsRfUaAgpZEALw_wcB>

* NCBI GEO:

<https://www.ncbi.nlm.nih.gov/geo/info/faq.html>

* GEO documentation

<https://www.ncbi.nlm.nih.gov/geo/info/>

* GeoTools: open source JAVA library providing tools for geospatial data:

<https://github.com/geotools/geotools>

<http://geotools.org/>

<http://docs.geotools.org/latest/userguide/tutorial/quickstart/eclipse.html>

* link to article about cities simulation as souspatial networks:

<https://books.google.co.il/books?id=AJU4DwAAQBAJ&lpg=PA201&ots=ibORQdPh70&dq=City%20simulation%20as%20a%20graph&pg=PA201#v=onepage&q=City%20simulation%20as%20a%20graph&f=false>

* paper on navigation algoritms:

<https://arxiv.org/pdf/1504.05140.pdf>

* OSM to geoJson:

<https://github.com/tyrasd/osmtogeojson>

* geoJson graphic simulator:

<http://geojson.io>

* MIT project about cities representation:

<https://github.com/t-mw/citygen>

* Open source project about procedural city modeling:

<https://github.com/phiresky/procedural-cities>