**Better Than Waze**

**High Level Design**

**Submitted by**: BTW group- Anat Tetroashvili, Shay Michaeli, Guy Rephaeli, Adam algressy, Sharon Hadar, Orel Kanditan.

**Supervised by: Tanya**

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

[Abstract 3](#_30j0zll)

[1. Introduction 4](#_3znysh7)

[1.1 General Project Description 4](#_2et92p0)

[1.2 Programming Environment 4](#_tyjcwt)

[2. Theoretical Background 4](#_1t3h5sf)

[2.1 Section 1 4](#_4d34og8)

[2.1.1 Sub Section 5](#_26in1rg)

[2.2 Section 2 5](#_lnxbz9)

[3. Basic System Functionalities 5](#_35nkun2)

[4. Software Implementation 5](#_1ksv4uv)

[4.1 Modules 5](#_44sinio)

[4.2 Top-Level View 5](#_2jxsxqh)

[4.3 Main Menu 5](#_z337ya)

[4.3.1 User Interface 5](#_3j2qqm3)

[4.3.2 Features 5](#_1y810tw)

[4.3.3 User Interface 5](#_4i7ojhp)

[5. References 5](#_2xcytpi)

# 

# 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 (may need to change the name)

## 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 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.

# Software Implementation

This is the programmer manual: Up to you – like:

## Modules

Add Modules diagram here

### User Module (may need to change the name)

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.

### Geo-Json Reading\Manipulation Module (must change the name)

### City Simulation

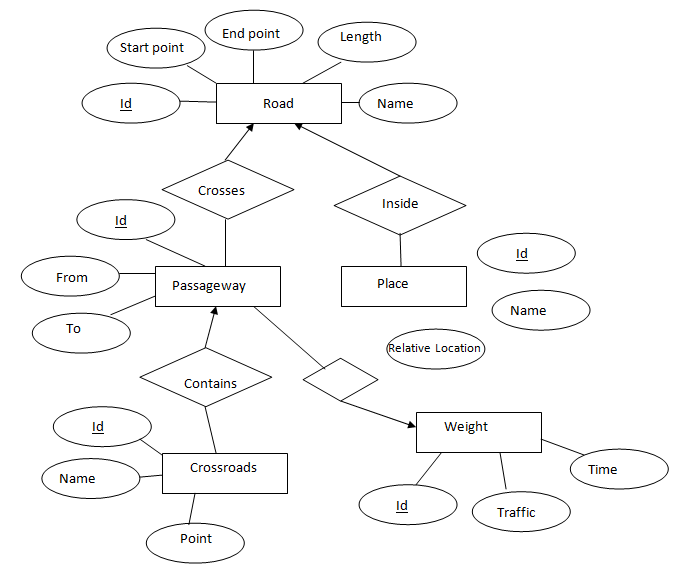
### Data Base

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:



### Navigation

The navigation module consumes the data from the data base. From that data, several graph representations of the roads are built, and then graph traversing algorithms are applied to find shortest paths in the graphs.

### Additional Module???

## 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: self-location, roads, special locations, intersections, traffic loads and behavior, traffic lights, etc. This later added info will also for translation in the input tier. any change of the map by the user will be translated to geojason, and the will be sent to the "Data Tier" and to the "Map Representation Tier".

**Map Representation tier:** consists of three main parts:

The map will be represented in geojason file. Every change to the map made by the user, or by the backend, will be translated to geojason and then will be seen on the map.

* 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 geojason 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 geojason file.
* User Input: all input added by the user after the first map initialization will be translated to geojason 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 geojason 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. the server will be notified of the data changes, then it will send the change to the "Map Representaion Tier" to be shown on the map.

**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

### Features

# References

* Papers and links