Bike Point Data Application

1. Introduction

The Bike Point Application is built to handle data from the Bike Point API within the Transport for London system. This app initializes a database (transport_in_london.sqlite), pulls data from an external API (https://api.tfl.gov.uk/bikepoint), exports data to a JSON file, and creates an interactive HTML map to visualize bike points. The project strengths in type safety, concurrency, and precise syntax to ensure a maintainable and robust solution.

2. How to Compile and Run the Application

In the pre-requisites Install Haskell (GHC) and Stack Ensure SQLite is installed.

Setup:

- Clone the repository : **git clone https://github.qmul.ac.uk/ec24099/haskell-project**
- Navigate to the project directory : **cd haskell-project**
- Pull the latest Code : **git pull origin main**

Run the Haskell Project: Assuming the project uses Stack, here are commands for both tools:

- To install dependencies and compile the code use the following:
  stack update
  stack build
  stack run –
- Commands

Command	**Description**
stack run -- create	Initialize the database and create bike related tables.
stack run -- loaddata	Download bike point data from api and store it in the database.
stack run – dumpdata	Export bike point data to a JSON file.
stack run -- <query> <query_arguments>	run some queries on the database (your choice)
stack run -- map	Creates HTML map of bike points.

- Queries Example Commands

`stack run -- bikes <station-name>` This query gets the number of available bikes at a station.
Example query command: **` stack run -- bikes "River Street , Clerkenwell" `**

`stack run -- nearby <lat> <lon> <radius>` This query finds the bike points within radius (km) of coordinates.
Example query command: **` stack run -- nearby 51.529163 -0.10997 1`**

`**stack run -- bikePoint <stationName>**` This query finds the bike points details for given station.
Example query command: ` **stack run -- bikePoint "River Street , Clerkenwell"** `

`**stack run -- checkDataForNoOfBikes <noOfBikes>** ` This query finds the bike points within radius (km) of coordinates.
Example query command: ` **stack run -- checkDataForNoOfBikes 13** `

3. Design Choices

Haskell
Haskel is known for its functional paradigm and type system and, which ensures code maintainability. Type safety reduces runtime errors, especially when handling structured data like JSON or database records.
SQLite Database:

SQLite
SQLite was selected for its lightweight integration. It is sufficient for managing the relatively small dataset provided by the Transport of London API.

JSON Serialization:
The **aeson** library was used for **JSON** parsing(decoding) and encoding due to its performance and effortless integration with Haskell's data types.

Leaflet.js
Leaflet.js was chosen for its ease of use and extensive mapping capabilities. It provides a smooth way to visualize data geographically.

Division of Responsibilities
The application is modularized into five discrete modules (Parse, Database, Fetch, Lib, and Types), each handling a specific aspect of the application, making it easier for extension and maintenance.

4. Map Feature

This map visualizes bike points with markers, each displaying the bike point's name and coordinates. The map dynamically calculates its center based on the average latitude and longitude of all points.

Challenges:
1. Calculating the map's center dynamically required precise averaging of coordinates.
2. Ensuring bike point names were clean and secure to prevent HTML injection involved precise string handling.
3. Creating JavaScript code dynamically within Haskell

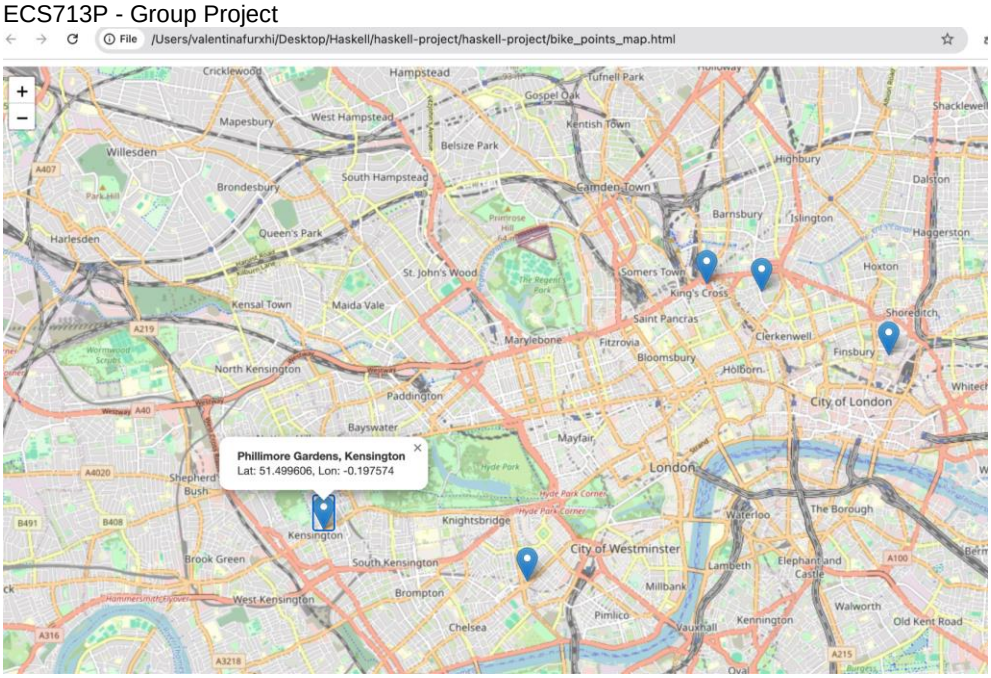

Fig1 : Bike Points in map

id	url	lat	lon	common_name
BikePoints_1	/Place/BikePoints_1	51.529163	-0.10997	River Street , Clerkenwell
BikePoints_2	/Place/BikePoints_2	51.499606	-0.197574	Phillimore Gardens, Kensington
BikePoints_3	/Place/BikePoints_3	51.521283	-0.084605	Christopher Street, Liverpool Street
BikePoints_4	/Place/BikePoints_4	51.530059	-0.120973	St. Chad's Street, King's Cross
BikePoints_5	/Place/BikePoints_5	51.49313	-0.156876	Sedding Street, Sloane Square

Fig 2. Some data from Bike Points table