

COM518 Assignment: Points Of interest

Instructions

Project setup

```
npm install
```

Compiles and hot-reloads for development

```
nodemon index.js
```

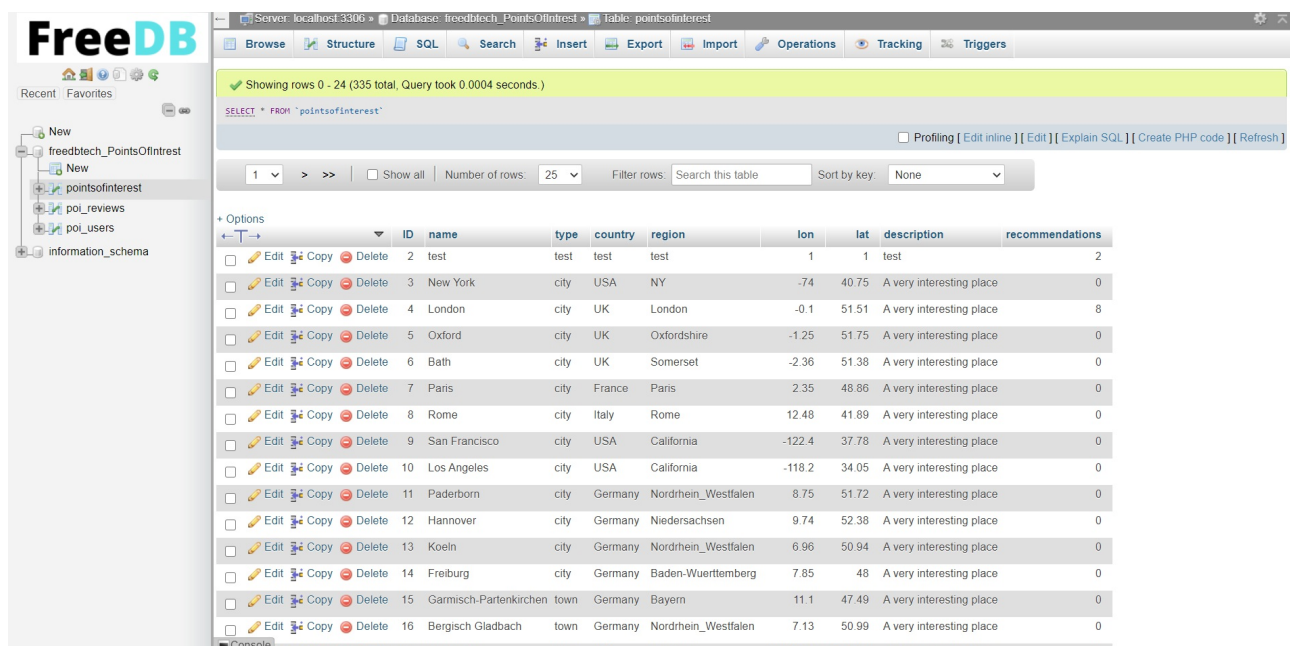
Login Credentials

Mail: **test@mail.com**
Password: **password**

Discussion on Development

Part A – Develop a very simple REST API

For this REST API rather than setting up a local MySQL server, I made use of a MySQL service (freedb.tech (www.freedb.tech)) and host the database. Proceeding to create and populate it after with the provided SQL file. This process would ideally make use of a .env file to configure the program accordingly and allow access, but for demonstration purposes, I have hardcoded everything in as well.



The screenshot shows the FreeDB web interface. On the left is a sidebar with a tree view of the database structure, including 'freedbtech_PointsOfInterest', 'poi_reviews', 'poi_users', and 'information_schema'. The main panel displays a table of points of interest. The table has columns: ID, name, type, country, region, lon, lat, description, and recommendations. It shows 16 rows of data, including locations like New York, London, Oxford, Bath, Paris, Rome, San Francisco, Los Angeles, Paderborn, Hannover, Koeln, Freiburg, Garmisch-Partenkirchen, and Bergisch Gladbach. Each row has interactive icons for Edit, Copy, and Delete. The interface also includes a search bar, a SQL query editor, and various toolbars for database management.

ID	name	type	country	region	lon	lat	description	recommendations
2	test	test	test	test	1	1	test	2
3	New York	city	USA	NY	-74	40.75	A very interesting place	0
4	London	city	UK	London	-0.1	51.51	A very interesting place	8
5	Oxford	city	UK	Oxfordshire	-1.25	51.75	A very interesting place	0
6	Bath	city	UK	Somerset	-2.36	51.38	A very interesting place	0
7	Paris	city	France	Paris	2.35	48.86	A very interesting place	0
8	Rome	city	Italy	Rome	12.48	41.89	A very interesting place	0
9	San Francisco	city	USA	California	-122.4	37.78	A very interesting place	0
10	Los Angeles	city	USA	California	-118.2	34.05	A very interesting place	0
11	Paderborn	city	Germany	Nordrhein_Westfalen	8.75	51.72	A very interesting place	0
12	Hannover	city	Germany	Niedersachsen	9.74	52.38	A very interesting place	0
13	Koeln	city	Germany	Nordrhein_Westfalen	6.96	50.94	A very interesting place	0
14	Freiburg	city	Germany	Baden-Wuerttemberg	7.85	48	A very interesting place	0
15	Garmisch-Partenkirchen	town	Germany	Bayern	11.1	47.49	A very interesting place	0
16	Bergisch Gladbach	town	Germany	Nordrhein_Westfalen	7.13	50.99	A very interesting place	0

In my design, I make use of a config file to store various keys and functions crucial to the program. They consist of passport configs, authentication functions and MongoDB keys all of which will later be used to complete further tasks.

For the rest of the project, I made use of an MVC development stack as it separates and isolates key services (controllers) from views and allows for a better working environment.

Initially, to ensure the functionality of my database, I started by creating a route with the suffix /poi which would involve an ajax call is made to my database that displays all of the data in a JSON format. This would, later on, act as a basis for all my other calls.

```
{
  "data": [
    {
      "id": 2,
      "name": "test",
      "lon": -74,
      "lat": 40.71
    },
    {
      "id": 3,
      "name": "New York",
      "lon": -74,
      "lat": 40.71
    },
    {
      "id": 4,
      "name": "London",
      "lon": 0,
      "lat": 51.51
    },
    {
      "id": 5,
      "name": "Oxford",
      "lon": 1.26,
      "lat": 51.75
    },
    {
      "id": 6,
      "name": "Bath",
      "lon": 2.35,
      "lat": 51.38
    },
    {
      "id": 7,
      "name": "Paris",
      "lon": 2.33,
      "lat": 48.86
    },
    {
      "id": 8,
      "name": "Rome",
      "lon": 12.5,
      "lat": 41.9
    },
    {
      "id": 9,
      "name": "San Francisco",
      "lon": -122.42,
      "lat": 37.77
    },
    {
      "id": 10,
      "name": "Los Angeles",
      "lon": -118.24,
      "lat": 34.05
    },
    {
      "id": 11,
      "name": "Paderborn",
      "lon": 8.45,
      "lat": 51.71
    },
    {
      "id": 12,
      "name": "Hannover",
      "lon": 9.75,
      "lat": 52.27
    },
    {
      "id": 13,
      "name": "Koeln",
      "lon": 6.96,
      "lat": 50.94
    },
    {
      "id": 14,
      "name": "Freiburg",
      "lon": 7.84,
      "lat": 48.06
    },
    {
      "id": 15,
      "name": "Garmisch-Partenkirchen",
      "lon": 10.83,
      "lat": 47.54
    },
    {
      "id": 16,
      "name": "Bergisch Gladbach",
      "lon": 7.1,
      "lat": 50.99
    },
    {
      "id": 17,
      "name": "Petersfeld",
      "lon": 8.5,
      "lat": 50.99
    },
    {
      "id": 18,
      "name": "Lyndhurst",
      "lon": 10.83,
      "lat": 50.99
    },
    {
      "id": 19,
      "name": "New Milton",
      "lon": 1.15,
      "lat": 50.99
    },
    {
      "id": 20,
      "name": "Southampton",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 21,
      "name": "Hedge End",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 22,
      "name": "Hythe",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 23,
      "name": "Alton",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 24,
      "name": "Portsmouth",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 25,
      "name": "Ringwood",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 26,
      "name": "Fordingbridge",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 27,
      "name": "Havant",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 28,
      "name": "Andover",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 29,
      "name": "Farnborough",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 30,
      "name": "Emsworth",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 31,
      "name": "Tadley",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 32,
      "name": "Waterlooville",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 33,
      "name": "Odiham",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 34,
      "name": "Hook",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 35,
      "name": "Eastleigh",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 36,
      "name": "South Hayling",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 37,
      "name": "Fleet",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 38,
      "name": "Gosport",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 39,
      "name": "Basingstoke",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 40,
      "name": "Bordon",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 41,
      "name": "Whitchurch",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 42,
      "name": "Lymington",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 43,
      "name": "Blackwater",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 44,
      "name": "Totton and Eling",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 45,
      "name": "Hartley Wintney",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 46,
      "name": "Winchester",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 47,
      "name": "Chandlers Ford",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 48,
      "name": "Bishops Waltham",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 49,
      "name": "Aldershot",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 50,
      "name": "New Alresford",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 51,
      "name": "Fareham",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 52,
      "name": "Ludgershall",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 53,
      "name": "Marlborough",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 54,
      "name": "Salisbury",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 55,
      "name": "Mokring",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 56,
      "name": "Staines-upon-Thames",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 57,
      "name": "Farnham",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 58,
      "name": "Godalming",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 59,
      "name": "Camberley",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 60,
      "name": "Egham",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 61,
      "name": "Guildford",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 62,
      "name": "Chertsey",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 63,
      "name": "Littlehampton",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 64,
      "name": "Petworth",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 65,
      "name": "Midhurst",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 66,
      "name": "Chichester",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 67,
      "name": "Selsey",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 68,
      "name": "Bognor Regis",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 69,
      "name": "Arundel",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 70,
      "name": "Vine Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 71,
      "name": "Fox and Hounds/Lone Barn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 72,
      "name": "Jolly Sailor",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 73,
      "name": "Millers Pond",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 74,
      "name": "The Royal Oak",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 75,
      "name": "The Langley Tavern",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 76,
      "name": "The Bridge",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 77,
      "name": "Salmon Leap",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 78,
      "name": "Cleveland Bay",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 79,
      "name": "Fleming Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 80,
      "name": "Dog and Crook",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 81,
      "name": "Stoneham Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 82,
      "name": "Travellers Rest",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 83,
      "name": "The Forest Home",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 84,
      "name": "The Cottage",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 85,
      "name": "The Swan",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 86,
      "name": "Chamberlayne Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 87,
      "name": "The Manor",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 88,
      "name": "Spike Island",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 89,
      "name": "Big Cheese",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 90,
      "name": "Fox and Hounds",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 91,
      "name": "The Obelisk",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 92,
      "name": "Hare and Hounds",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 93,
      "name": "The Rising Sun",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 94,
      "name": "Stones",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 95,
      "name": "Humble Plumb",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 96,
      "name": "Peartree Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 97,
      "name": "Red Lion",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 98,
      "name": "The Earl of Locksley",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 99,
      "name": "Gardeners Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 100,
      "name": "The Ark",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 101,
      "name": "The Cricketers",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 102,
      "name": "The Sun Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 103,
      "name": "The Chamberlayne Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 104,
      "name": "The Grantham Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 105,
      "name": "The Bent Brief",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 106,
      "name": "The Litten Tree",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 107,
      "name": "Percy Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 108,
      "name": "Good Companion",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 109,
      "name": "Exford Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 110,
      "name": "Man Fare Pub",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 111,
      "name": "Man Fare Harvester",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 112,
      "name": "Bishop Blaize",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 113,
      "name": "The Monks",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 114,
      "name": "Orange Rooms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 115,
      "name": "The Pilgrim Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 116,
      "name": "The Anchor Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 117,
      "name": "The Winstons",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 118,
      "name": "The Blue Keys",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 119,
      "name": "The Bellesmoor",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 120,
      "name": "Chillworth Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 121,
      "name": "The Malvern",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 122,
      "name": "Ice House",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 123,
      "name": "Baddesley Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 124,
      "name": "The Platform Tavern",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 125,
      "name": "The Mountbatten",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 126,
      "name": "The Hop Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 127,
      "name": "The Queens Head",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 128,
      "name": "The Bold Forester",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 129,
      "name": "Anchor Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 130,
      "name": "The Hinkler",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 131,
      "name": "The Bullseye",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 132,
      "name": "Foresters Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 133,
      "name": "Longhead Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 134,
      "name": "Bald Faced Stag",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 135,
      "name": "The Old House at Home",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 136,
      "name": "William IV",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 137,
      "name": "Wellington Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 138,
      "name": "The Freemantle",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 139,
      "name": "Englishman",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 140,
      "name": "Freemantle Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 141,
      "name": "The Heath (Table Table)",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 142,
      "name": "Budes Lea",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 143,
      "name": "Bricklayers Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 144,
      "name": "The Ship",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 145,
      "name": "Rising Sun",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 146,
      "name": "Waterloo Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 147,
      "name": "Turfcutters Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 148,
      "name": "TGI Fridays",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 149,
      "name": "Ship Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 150,
      "name": "Frankie and Bennys",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 151,
      "name": "The Hunters",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 152,
      "name": "Southampton Arms",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 153,
      "name": "Prince Consort",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 154,
      "name": "Zen",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 155,
      "name": "The Red Lion",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 156,
      "name": "The Wagon Works",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 157,
      "name": "Linden Tree",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 158,
      "name": "The King Rufus",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 159,
      "name": "The Village Bells",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 160,
      "name": "King George",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 161,
      "name": "Ennios",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 162,
      "name": "La Regatta",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 163,
      "name": "The Manor House",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 164,
      "name": "The Plough",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 165,
      "name": "Grande Caf  ",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 166,
      "name": "La Cantina",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 167,
      "name": "Eleenagles",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 168,
      "name": "The Fitzhugh",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 169,
      "name": "Banana Wharf",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 170,
      "name": "The Otter",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 171,
      "name": "The Croft (Closed)",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 172,
      "name": "Sunrise Balti and Tandoori",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 173,
      "name": "Nashua",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 174,
      "name": "Windhover Manor",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 175,
      "name": "The New Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 176,
      "name": "Grove Tavern",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 177,
      "name": "Firehouse",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 178,
      "name": "Glen Bar",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 179,
      "name": "Regents Park",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 180,
      "name": "Osbourne",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 181,
      "name": "The Cricketers",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 182,
      "name": "Key and Anchor",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 183,
      "name": "Prezzo Romey",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 184,
      "name": "The Olive Tree",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 185,
      "name": "Que Pasa",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 186,
      "name": "Yatess",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 187,
      "name": "Namaste Kerala",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 188,
      "name": "The Stile",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 189,
      "name": "The Rover",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 190,
      "name": "Brass Monkey",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 191,
      "name": "Table",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 192,
      "name": "The Margherita",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 193,
      "name": "The Hiltonbury Farmhouse",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 194,
      "name": "Jolly Sailor",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 195,
      "name": "Finicki Food Company",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 196,
      "name": "Encore",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 197,
      "name": "Aroma",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 198,
      "name": "El Rancho",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 199,
      "name": "The Standing Order (D Wetherspoon)",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 200,
      "name": "Bellinis",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 201,
      "name": "The London Hotel",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 202,
      "name": "Peking Phoenix",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 203,
      "name": "Nashua",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 204,
      "name": "The Tavern",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 205,
      "name": "The Tudor Rose",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 206,
      "name": "Thai Cottage",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 207,
      "name": "The Ketch Rigger",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 208,
      "name": "Boomerang",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 209,
      "name": "Ye Olde Whyte Hart",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 210,
      "name": "The Riverside Bar and Restaurant",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 211,
      "name": "The Cinnamon Bay",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 212,
      "name": "The Malt and Hops",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 213,
      "name": "The Station",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 214,
      "name": "The Welcome Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 215,
      "name": "The New Clock Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 216,
      "name": "Unit Nightclub",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 217,
      "name": "Capers",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 218,
      "name": "The Griffin",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 219,
      "name": "Shooting Star",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 220,
      "name": "Giddy Bridge",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 221,
      "name": "Avondale House",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 222,
      "name": "Plume of Feathers",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 223,
      "name": "The Joiners",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 224,
      "name": "Nicks Restaurant",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 225,
      "name": "Park Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 226,
      "name": "The Swan Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 227,
      "name": "The Galley Restaurant",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 228,
      "name": "The Roebuck",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 229,
      "name": "White Horse",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 230,
      "name": "The Thai Corner Restaurant",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 231,
      "name": "Royal Bengal",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 232,
      "name": "The Bugle",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 233,
      "name": "The King and Queen",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 234,
      "name": "The Victory Inn",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 235,
      "name": "Rat and Parrot",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 236,
      "name": "Goblets",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 237,
      "name": "Bitterne Balti",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 238,
      "name": "Charlie Chans",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 239,
      "name": "Nazs Cuisine",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 240,
      "name": "Purbani Tandoori Restaurant and Take Away",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 241,
      "name": "South Garden Chinese Restaurant",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 242,
      "name": "Water Margin",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 243,
      "name": "The Master Builder",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 244,
      "name": "Sanjha Restaurant",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 245,
      "name": "Test",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 246,
      "name": "Test",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 247,
      "name": "Chalcraft Distribution Park",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 248,
      "name": "harbour",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 249,
      "name": "Britannia Wharf",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 250,
      "name": "Port of Southampton",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 251,
      "name": "Blackdown",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 252,
      "name": "Pikes Peak",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 253,
      "name": "Cauterets",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 254,
      "name": "Dartmoor",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 255,
      "name": "The Crown",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 256,
      "name": "Costa Coffee",
      "lon": 1.33,
      "lat": 50.99
    },
    {
      "id": 257,
      "name": "Costa Coffee",
      "lon": 1.33,
      "lat": 50.99
    }
  ]
}
```

1. Look up all points of interest in a given region

For this search functionality, I started by creating a route that takes in a form input

```

async function ajaxAdd() {
  const body = {
    name: document.getElementById("name").value,
    type: document.getElementById("type").value,
    country: document.getElementById("country").value,
    region: document.getElementById("region").value,
    lat: document.getElementById("lat").value,
    lon: document.getElementById("lon").value,
    description: document.getElementById("description").value,
  };
  const response = await fetch("/poi/post", {
    method: "POST",
    headers: { "Content-type": "application/json" },
    body: JSON.stringify(body),
  });
}

```

The SQL logic for this code inserts data from the requests body and populates the newly added record accordingly. Storing the response as a JSON value and sending it back to be used.

```

INSERT INTO pointsofinterest
(name, type, country, region, lon, lat, description, recommendations)
VALUES
(?, ?, ?, ?, ?, ?, ?, ?),
[poi.name, poi.type, poi.country, poi.region, poi.lon, poi.lat, poi.description, 1]

```

□ Edit Copy Delete 1112 My Test Type Test UK London 1 1 Description Test 1

3. Recommend a point of interest. This API endpoint should read in the POI ID and increase the number of recommendations by one for that POI

This ajax call is present in my region search functionality, I added a button to each result where, if clicked, a call is made to the recommendation service that increments the results corresponding table with 1. It accomplishes this by searching for the corresponding record based on the id.

```

async function ajaxRecommend(id) {
  const response = await fetch(
    `http://localhost:8080/poi/recommend/${id}`
  );
}

```

For this SQL statement, the recommendation of the provided req parameter ID is updated by incrementing with 1.

```

UPDATE pointsofinterest
SET recommendations = recommendations + 1
WHERE ID='${id}'

```

Troubles Part A

The troubles I faced with this kind of approach was deviating from what we were taught and isolating each of the ajax calls into its route and respective file. I also didn't fully understand how to access the route parameter which required me to look into it. However, I was able to successfully fill in my gaps of knowledge and produce 3 working calls.

Part B – Develop a simple AJAX-based JavaScript front-end

For this project, I decided to make use of EJS as a view engine, as it allows me to make components that can be reused for multiple pages, and centralize my external scripts and links in the header of my project. Each page is linked and connected through a custom ejs component called navbar, which I added to allow easy navigation between pages.

The project makes use of bootswatch, a bootstrap variant, to display data in its own unique style, making it look nice and removing the need for custom css throughout my project. This also provides a unified and dignified look to the project, lending itself to clear consistency throughout for users.

4. Search Page

This page allows users to search for all points of interest in a given region. The user can enter a region using the provided form the entire page consists of. After the system makes use of JavaScript to communicate with the systems REST API finding and returning all points of interest in that region. The results are added to a list and appended accordingly using a foreach function.

HOME

SEARCH

ADD

REVIEW

LOGOUT

Hampshire

SEARCH

523 Name: Petersfield Recommendations: 2	RECOMMEND
524 Name: Lyndhurst Recommendations: 0	RECOMMEND
525 Name: New Milton Recommendations: 0	RECOMMEND
526 Name: Southampton Recommendations: 1	RECOMMEND
527 Name: Romsey Recommendations: 0	RECOMMEND
528 Name: Hedge End Recommendations: 0	RECOMMEND
529 Name: Hythe Recommendations: 0	RECOMMEND
530 Name: Alton Recommendations: 0	RECOMMEND
531 Name: Portsmouth Recommendations: 0	RECOMMEND
534 Name: Havant Recommendations: 0	RECOMMEND
535 Name: Andover Recommendations: 0	RECOMMEND
537 Name: Emsworth Recommendations: 0	RECOMMEND

5. Add POI page

This page would allow the user to enter point of interest details into a provided form, using JavaScript to communicate with my web API it adds POI accordingly. The results are then sent and displayed on my database once completed. It consists of a form with multiple required inputs that need to be filled out before any requests can be made.

HOME

SEARCH

ADD

REVIEW

LOGOUT

Details

My Test

Type Test

UK

London

Coordinates (lng & lat)

00.1

00.1

Description:

Description Test

SUBMIT

1112

My Test

Type Test

UK

London

1

1

Description Test

1

6. Recommend button

When processing the search results, I create a "Recommend" button for each result that, when clicked, sends an AJAX POST request to the REST API done in task 3. This increments the results accordingly and would allow the user to recommend the POI for others to see.

HOME

SEARCH

ADD

REVIEW

LOGOUT

London

SEARCH

4 | Name: London | Recommendations: 8

RECOMMEND

HOME

SEARCH

ADD

REVIEW

LOGOUT

London

SEARCH

4 | Name: London | Recommendations: 9

RECOMMEND

Troubles Part B

As this is my first time using EJS for a project of this scale, I struggled setting up my components properly the first time, eventually learning that all pages can be embedded in the layout view. From there on I was able to successfully create the views and have them flow into one another.

Part C – Adding simple error-checking

7. Error-checking

By making use of the required variable I can ensure that all fields of the task2s form are filled in by the user, ensuring that if any of the POI details are blank the appropriate message is sent back to the client. If something were to go wrong during the creation process I have made the system so that an alert displaying the appropriate error code is returned in case of a problem with the creation call.

```
try {
  res.json(await pointsOfInterestPost.create(req.body));
} catch (err) {
  console.error(`Error while creating points of interest`, err.message);
  alert(err);
}
```

Troubles Part C

There were no real struggles with the error checking addition as I could simply require all inputs and try-catch my ajax calls to output the relevant errors.

Part D – Adding a map

8. Leaflet Map

Using leaflet I was able to add an OpenStreetMap map to the page, which can be used accordingly. I the map to never render copies and to Show itself in its entirety upon mounting. Moreover, the popup functionality enables users to see lat and lng of the position they clicked on in a popup.

When searching for a region the map makes an ajax call and gathers all the data to loop through and add markers dynamically on the map using a foreach method. This ensures that all relevant POIs are displayed and accessible to the user

```
const map = L.map("map").setView([40.505, -0.0], 2);

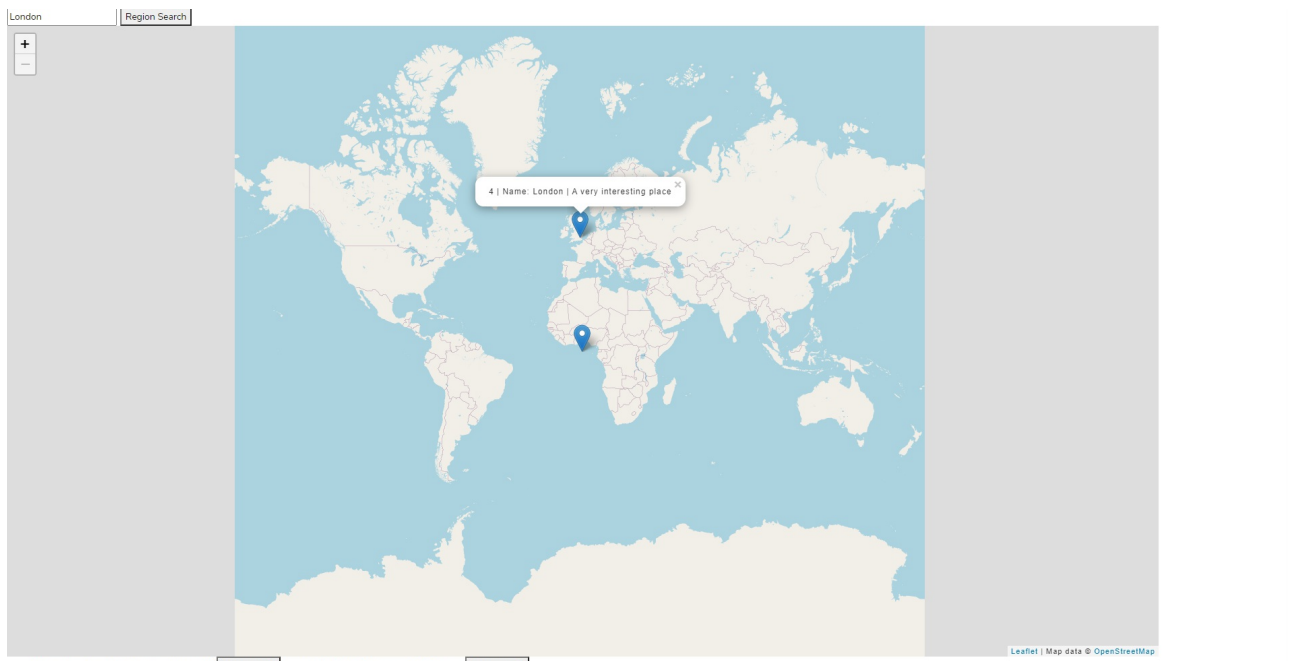
L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
  maxZoom: 18,
  minZoom: 2,
  attribution:
    "Map data &copy; " +
    '<a href="http://openstreetmap.org">OpenStreetMap</a>',
  id: "examples.map-i875mjb7",
  noWrap: true
}).addTo(map);

var popup = L.popup();

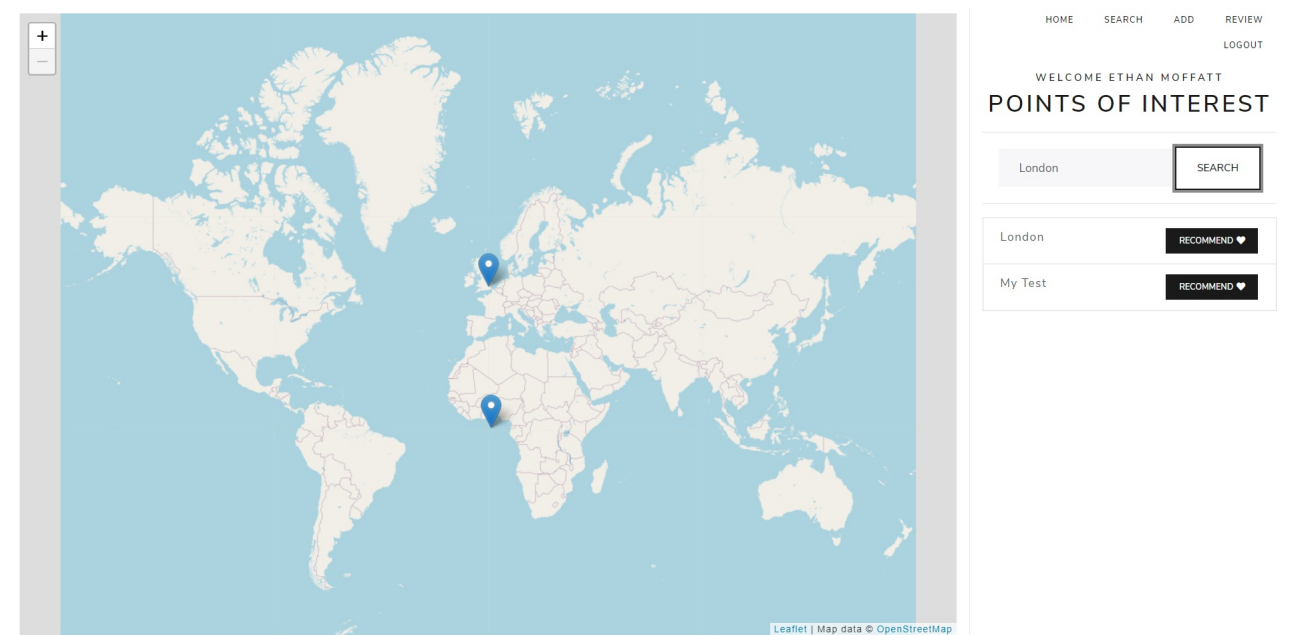
function onMapClick(e) {
  popup
    .setLatLng(e.latlng)
    .setContent('Add map at ' + e.latlng.toString())
    .openOn(map);
}

map.on("click", onMapClick);
```

In addition to the map, whenever the search form is inputted with a valid value, the results are displayed as markers on the map. When clicked on the point of interest name and description appear as a popup. This is accomplished by taking the Json data gotten from the Ajax call and looping through it with a foreach function, creating markers accordingly.



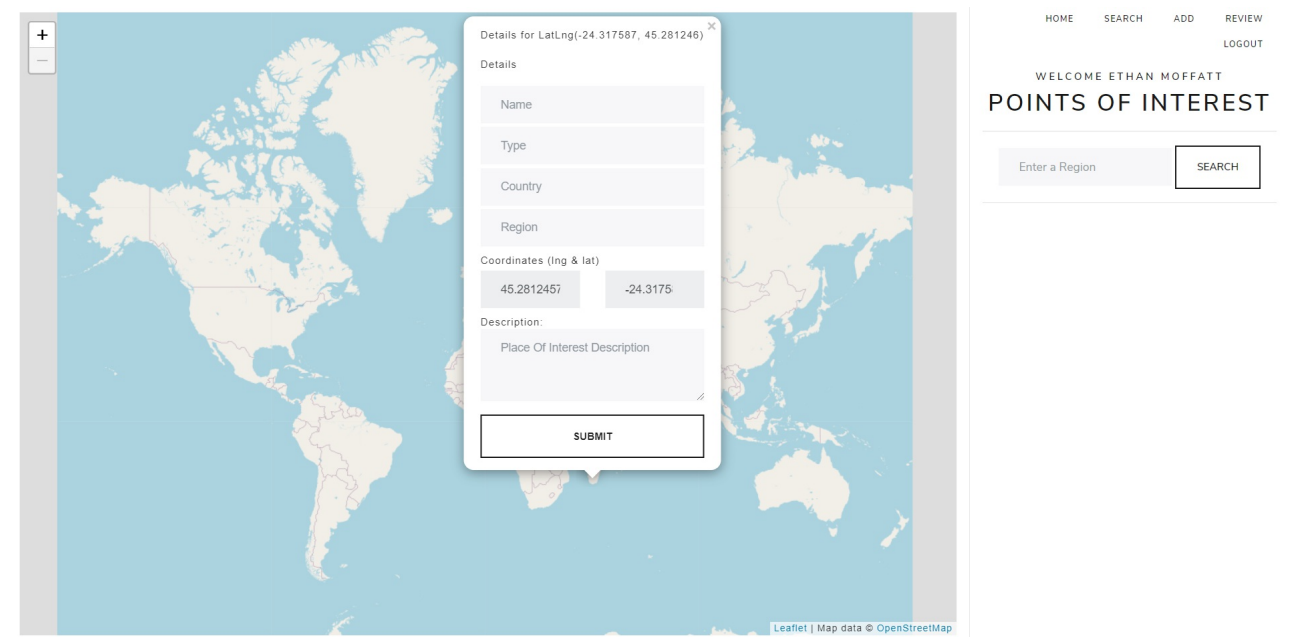
At this point, having assembled all the basic functionality of my project, I opted to create a suitable and professional-looking interface where all previous functionality would be combined into a singular coherent page. This page lays out the necessary features in an organized fashion.



The Search functionality now displays the data as seen in task 1 and 4 on the map and also in its own contained list on the right, with recommending functionality enabled for each.

9. Add a POI on the map

When you click on the map at a particular location a popup opens prompting the user to enter the POI details. The form is identical to the one in task 2 and 5 with the same error checking methods. Once the submit button is the data is sent over to your REST API and adds a new POI.



Troubles Part D


I have had extensive previous experience dealing with Mapbox, another service that provides maps similar to leaflet., meaning the gist was easy to comprehend and elaborate on. However, I did initially struggle with correctly adding POI locations as I had to do extensive research on how to get the lat lng of a point once clicked. Eventually, I discovered how to and could dynamically add them to the generated form application.

Part E – logins and sessions

10. Implement a session-based login system

For the login system, I immediately made use of passport with MongoDB as opposed to the MySQL alternative stated here. This allowed me more control over the system as a whole and a more secure login system as a whole. Allowing users to access pages and requests only if they were authenticated on the system. From the login page, users are able to input their email and password to access the system, however, they can also sign up if need be in a secure way that would not compromise their credentials, as passwords are hashed and encrypted on creation.

Login

 LOGIN


Email

Password

LOGIN

No Account? [Register](#)

Sign Up

 REGISTER

Name

Email

Password

Confirm Password

Have An Account? [Login](#)

When a user logs in successfully, a message containing the user's name appears to confirm that they have successfully signed into the application.

HOME SEARCH ADD REVIEW
 LOGOUT

WELCOME TESTY MCTESTERSON
 POINTS OF INTEREST

11. Ensure login enabled

Due to my use of a passport for a login system, I am able to make requests and pages only available to logged-in users. This is done by adding the `ensureAuthenticated` method to my routes. A simple yet effective method that allows me to ensure only those people who have authenticated can access the Database

```
router.get('/region', ensureAuthenticated, (req, res) => res.render('pages/region'));
router.get('/add', ensureAuthenticated, (req, res) => res.render('pages/addPointsOfInterest'));
router.get('/map', ensureAuthenticated, (req, res) => res.render('pages/map'));
router.get('/review', ensureAuthenticated, (req, res) => res.render('pages/review'));
```

Troubles Part E

I've used passport multiple times before so didn't struggle with its implementation, but did however have some trouble trying to implement it into the SQL database due to the user information. Hence i decided to let the system use mongoose to model a user object which was sent to a MongoDB server.

Part F – Implementing a review system

12. Review API

This allows clients to review a POI by reading in the POI ID POSTing data from the request body. It does this through a simple form and ensures that the review is not blank, only to be accessed when a user is logged in.





Description:

Enter an id

Place Of Interest review

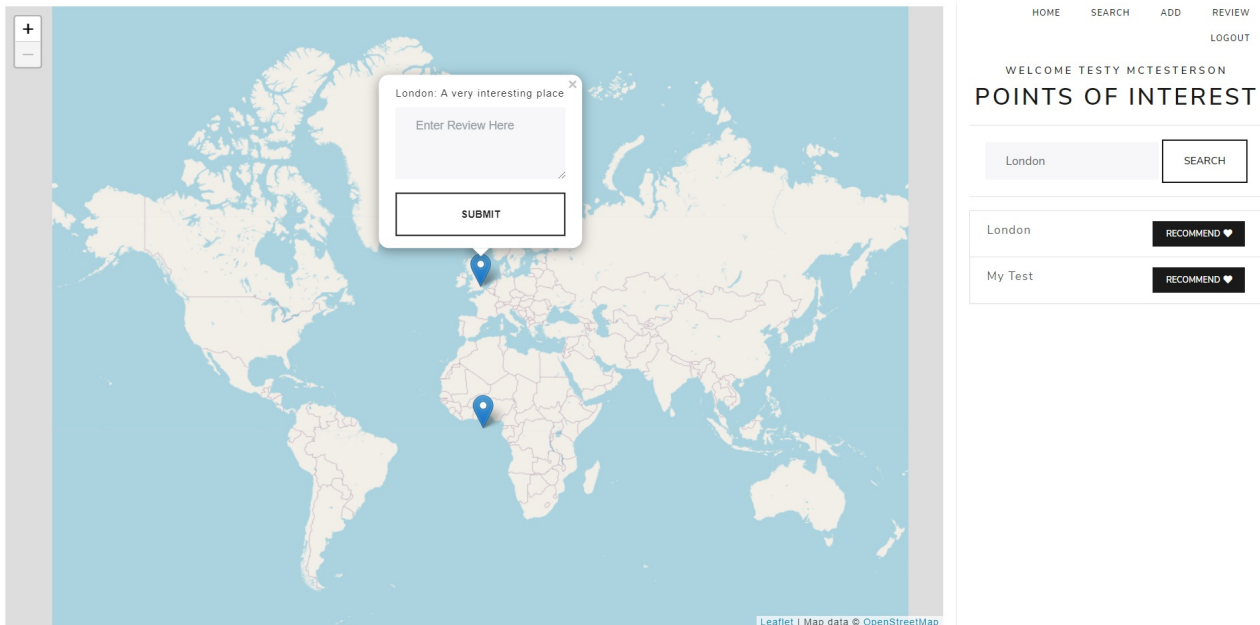
SUBMIT

+ Options

				id	poi_id	review
<input type="checkbox"/>		Edit		Copy		Delete
				2	4	Bit Dusty
<input type="checkbox"/>		Edit		Copy		Delete
				3	4	Yee

13. Add a review box to the popup

As an extension on task a, whenever a user was to search a location and click on its corresponding popup, the system displays a review option beneath it. This allows the user to enter a review sending it along with the POI ID to the recently created review REST API.



Part F Troubles

There were no troubles here as it was as simple as copying the POI adding request and amending the SQL statement to add data to another table. Hence it worked as intended from the start

Part G - Extension

- creating a well-structured Node application with models, controllers and routes and using ECMAScript 6 classes;
 - I have made use of all the above and more to ensure my program works as intended and is easily readable and understandable through OOD.
- using Passport for authentication rather than express-session;
 - From the start, I made use of MongoDB and passport to make a secure login system that can be used for the system as appropriate
- creating a professional-looking site, including handling errors in a user-friendly, "real-world" way.
 - My site uses a modern and minimalist look designed to give a professional feeling throughout use
 - All my errors are presented in a user-friendly way that does not take away from the site.