Text, logo

Description automatically generated

**SOUTHAMPTON SOLENT UNIVERSITY**

BSc (Hons) Computing

**Points of Interest Application**

Author : 10152420

Course title : Web Application Development (QHO540)

**Tutor’s name : Amjad Alam**

Module title : Web Application Development (Level 5)

**GitHub: *https://github.com/sarkersh/Web-Application-Development-COM518/tree/main/pointsOfInterestFinal***

**Introduction**

Thea aim of this report is to provide a brief technical overview and a user guide for the online points of interest website. The website allows users to look up information on places they might want to visit whilst on holiday.

**System Overview**

When a user visits the homepage, they are presented with a search box that allows them to enter a region they are interested in. This will submit the search to the backend server via ajax. The result is then rendered on the screen using DOM manipulation on the browser via javascript.

Graphical user interface, chart, treemap chart

Description automatically generated

Fig: *Find points of interest in London region*

The screenshot above shows the homepage with a list of interesting places to be found in London.

**Recommendation**

For each point of interest found the user can press the Recommend button to recommend the place. This information is submitted to the recommendation api on server via ajax.

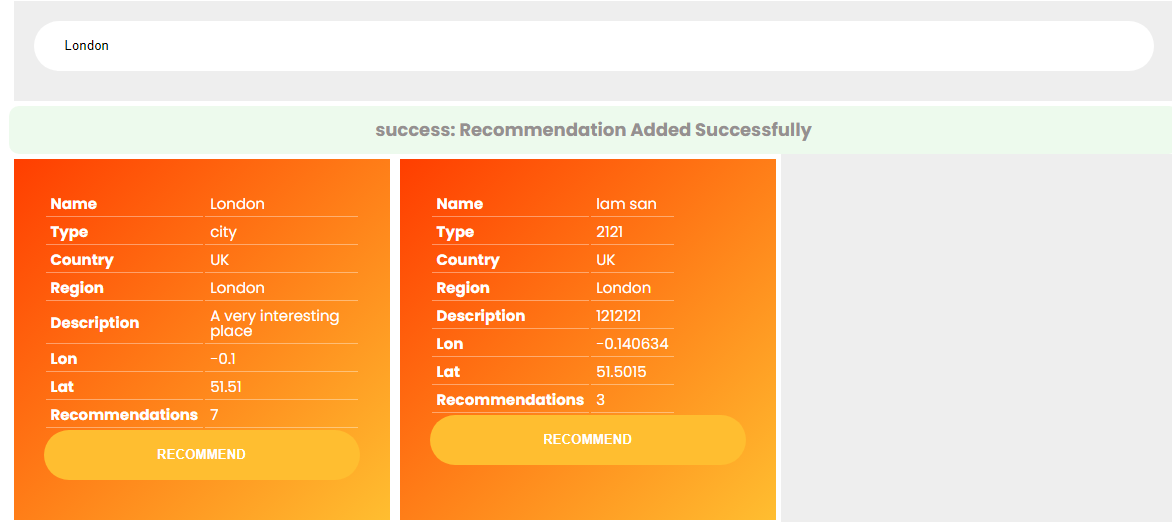


Fig: *Adding recommendations in London region(Task Part A3)*

As you can see the system presents a user-friendly feedback to let the user know if the request is submitted successfully.

**Map View**

The user can get a visual representation of the points of interest on a map view.

When the user searches for points of interest in a given region, the locations are presented on the map.

You can see the result of searching for points of interest in London. The locations are represented by markers on the map.

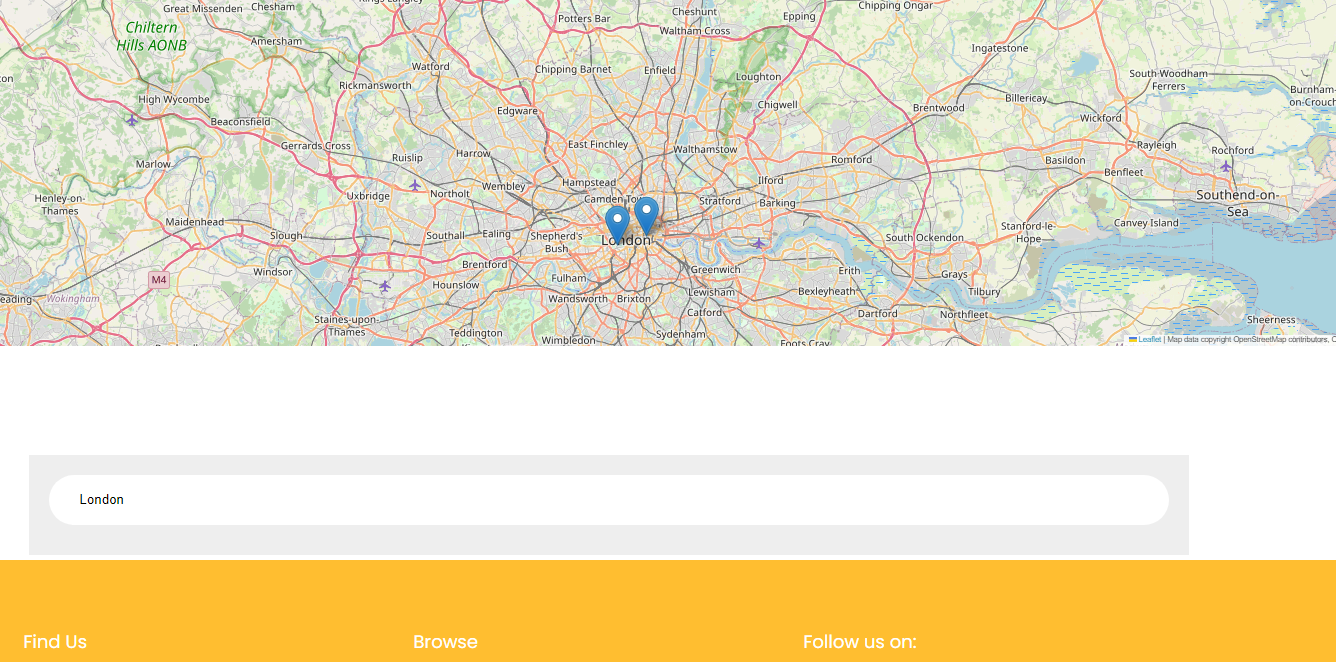


Fig: *Map view in London region(Task PartA1)*

**User Reviews**

Users can leave a review by clicking the location marker and entering their review. This data is sent to the reviews api on the server.

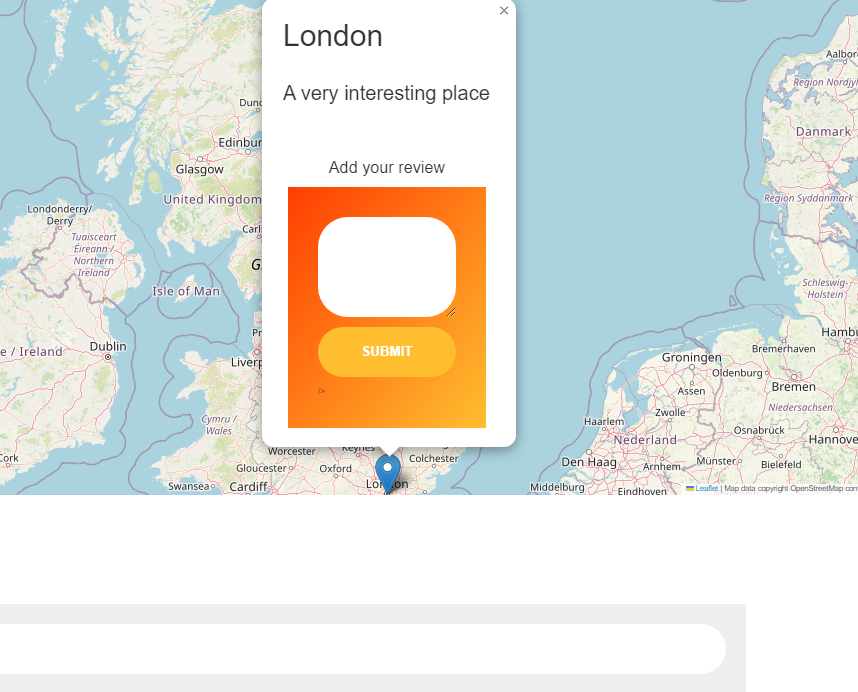


Fig: *User reviews form to add their comments*

The reviews end point also requires user login.

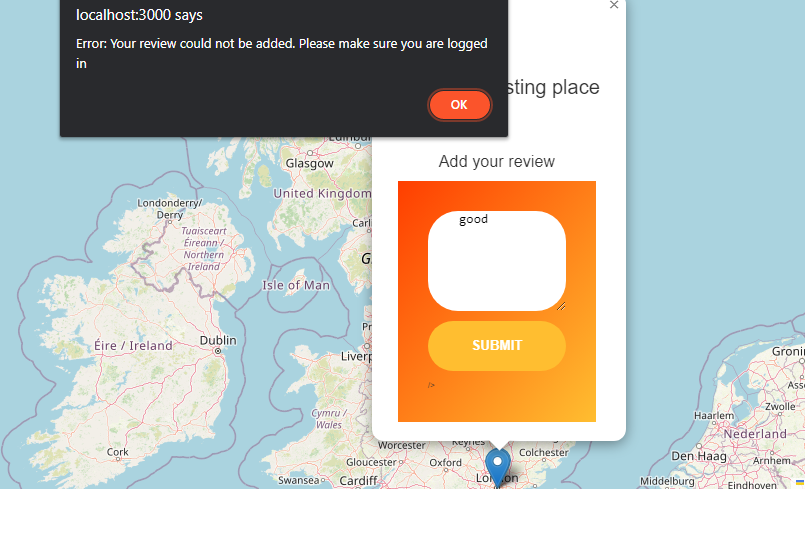
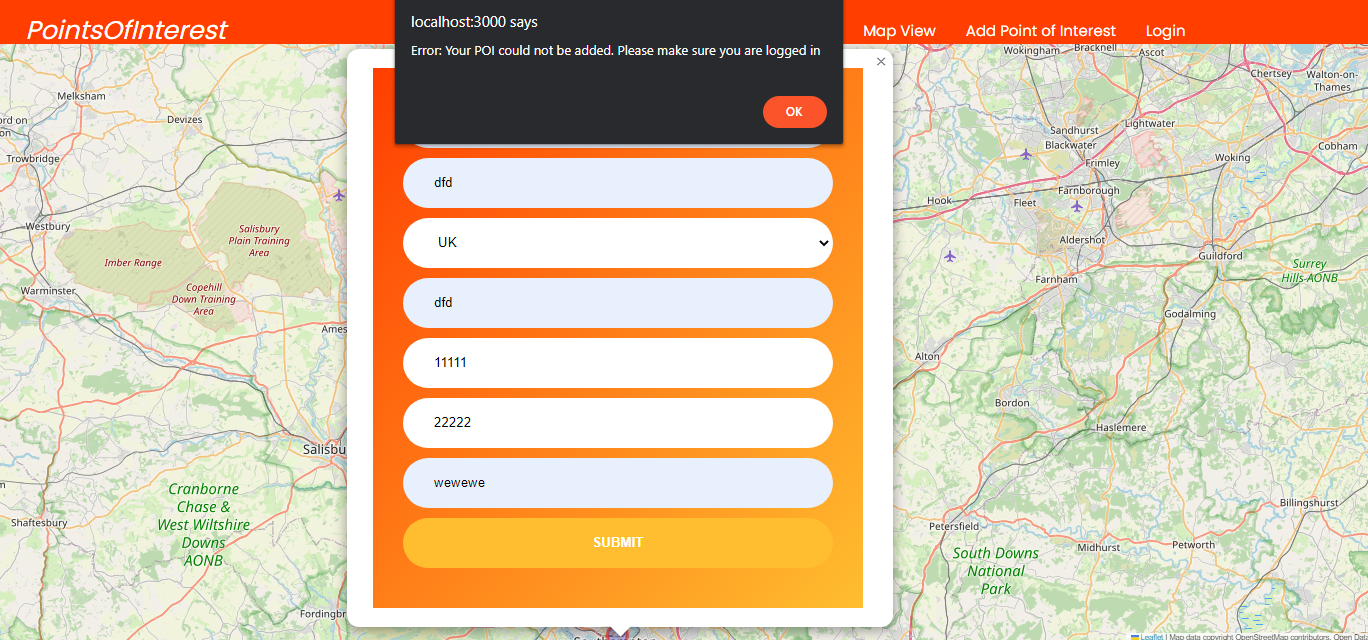


Fig: *Non user review alert*

When the map loads up a red circle is visible on the Solent location. By clicking on this red circle, the user can enter a new point of interest.

The user must be logged in to add appoint of interest. If not, an error message is displayed prompting the user to log in.



there result for searching.

Fig: *Non logged in user alert for add point of interest*

**Add Point of interest**

When the user clicks the Add Points of Interest link. The page will only load if the user is logged in. If not, they are redirected to the login page.

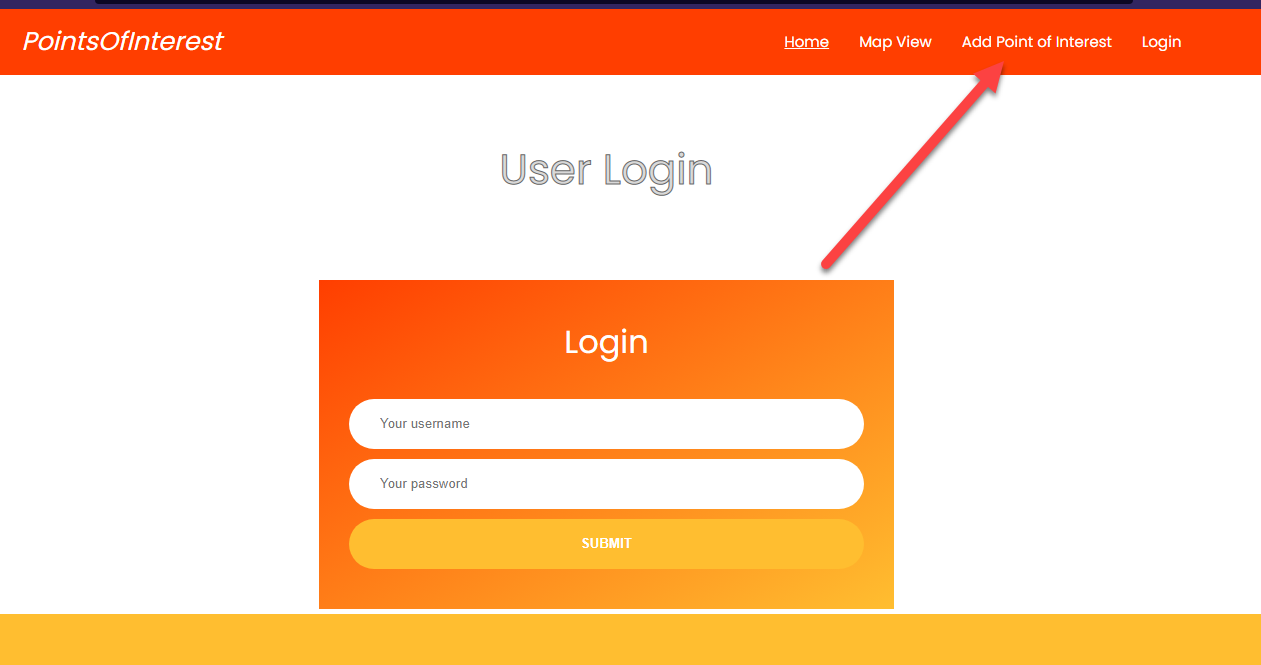


Fig: *User log in Screen to add Point Of Interest(Task Part A2)*

Once the user logs in the “Add Point of interest” screen is loaded. The user then completes all the fields and submit. User friendly feedback received by the user to alert them of any errors and to inform them that they are successful.

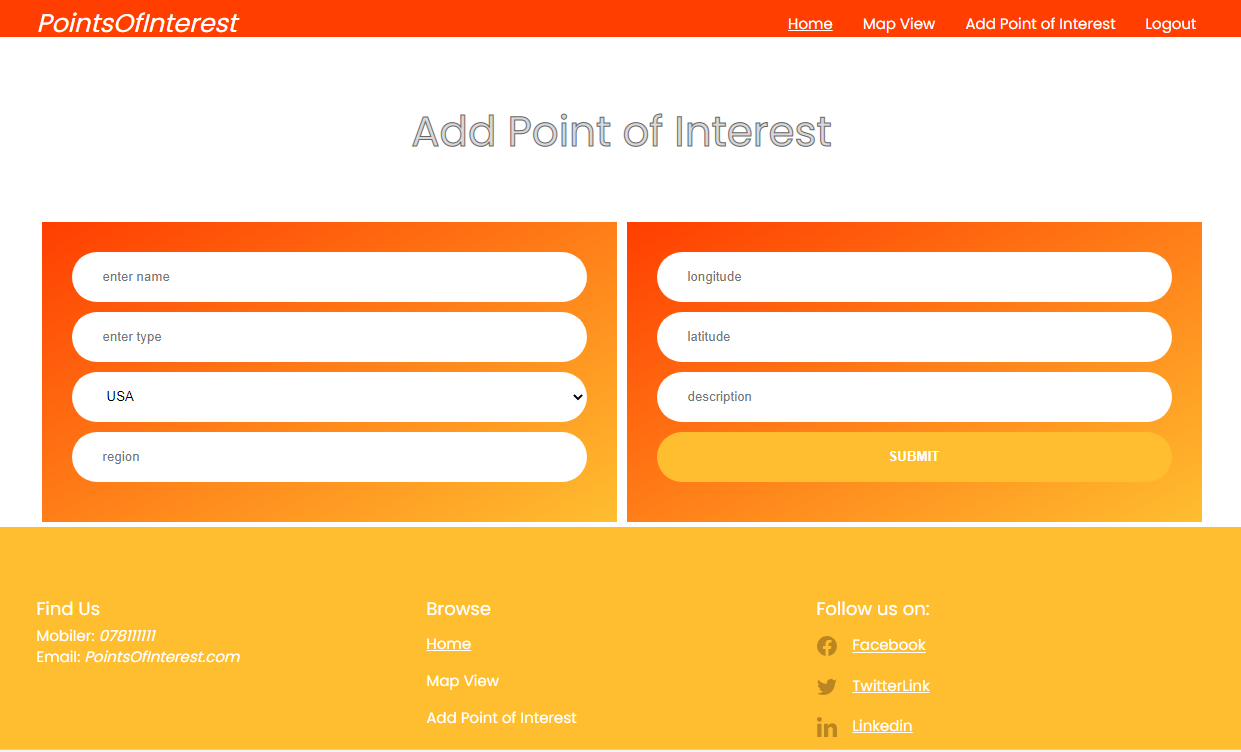


Fig: *Adding points of interest form*

**Tech Stack**

Here are the tech used in this assignment/project below:

**Database Design**

MySQL database is used for data storage.



**API Endpoints**

The API endpoints are built with Nodejs and express js. For client-side rendering, we use the ejs Templating engine.

For DOM manipulation JavaScript is used and for styling I use CSS.

**MVC**

To keep the code clean and easy to manage, the MVC design pattern is used. Because the project is small, the routers act as controllers rather than creating dedicated controller files.

Here is an example of how MVC is used.

When the router/api endpoint receives a request, it passes the submitted data to the Dao layer. The Dao then queries the database and pass the result to the router/controller. The controller then pass the data to the browser (view layer).

**Route/Controller**

Take the poi route/controller. You can see in the diagram below there are two api endpoints , one to load the **add points of interest** page and the other to search by region.

Both routes use the get method.

In the first ”/add” endpoint you will notice a middleware function ***isLoggedIn*** added. This middleware check to see if the user is logged in before they are allowed to access the page to add points of interest.



Fig: *Api to get poi by region*

**DAO**



Fig: *Add poi to mysql db on DAO Layer*



**Fig:** *search poi by region on DAO Layer*

**View Layer**



Fig: *Home page to render search box and results*

**Client side JavaScript code to add poi**

const *addPOI* = async (poiData) => {  
  
 *fetch*('http://localhost:3000/poi/', {  
 method: 'POST',  
 headers: {  
 "Content-type": "application/json"  
 },  
 body: JSON.stringify(poiData)  
 })  
 .then(function(response) {  
 return response.json();  
 })  
 .then(function(poiResult) {  
  
 const messageBox = document.getElementById('message-box');  
 if(poiResult.status == 'success'){  
  
 messageBox.classList.add('success');  
 messageBox.classList.remove('error');  
 messageBox.innerHTML = `<strong>${poiResult.status}: ${poiResult.message}`;  
 }else{  
 if(poiResult.status == 'error'){  
 messageBox.classList.add('error');  
 messageBox.classList.remove('success');  
 messageBox.innerHTML = `<strong>${poiResult.status}:</strong> ${poiResult.message}`;  
 }  
 }  
  
 })  
 .catch(function(error) {  
 const messageBox = document.getElementById('message-box');  
 messageBox.classList.add('error');  
 messageBox.classList.remove('success');  
 messageBox.innerHTML = `<strong>${poiResult.status}:</strong> ${poiResult.message}`;  
 });  
  
}

**Security and Scalability**

The system uses passport middleware for authentication with local Strategy.

**Module system**

I use the ES6 module system throughout the project rather than the older Common JS. To implement es6 modules, node requires certain changes. One way of achieving this is to include a “type” entry in your package. Json file as shown below.

Text

Description automatically generated

**Conclusion and Reflection**

In conclusion, the web application has implemented all key requirements (**Task: Part A to G**) of the assignment. I have applied my understanding of the course material to develop the site.

I have used latest technologies such as ECMAScript 6 classes module system, promises and es6 arrow functions and more. I have also applied good design and development principles in the form of MVC as well as better security provided by passport.

It was not easy implementing es6 module system in NodeJS. One of the issues is that “\_\_filename” is not supported in the NodeJS implementation of es6 modules. To overcome this, I had to install a third-party node module and used it as shown below:

import { fileURLToPath } from 'url';

const \_\_filename = fileURLToPath(import.meta.url);

const \_dirname = path.dirname(\_filename);

I also had to add the .js extension to the “bin/www” file to get things working.