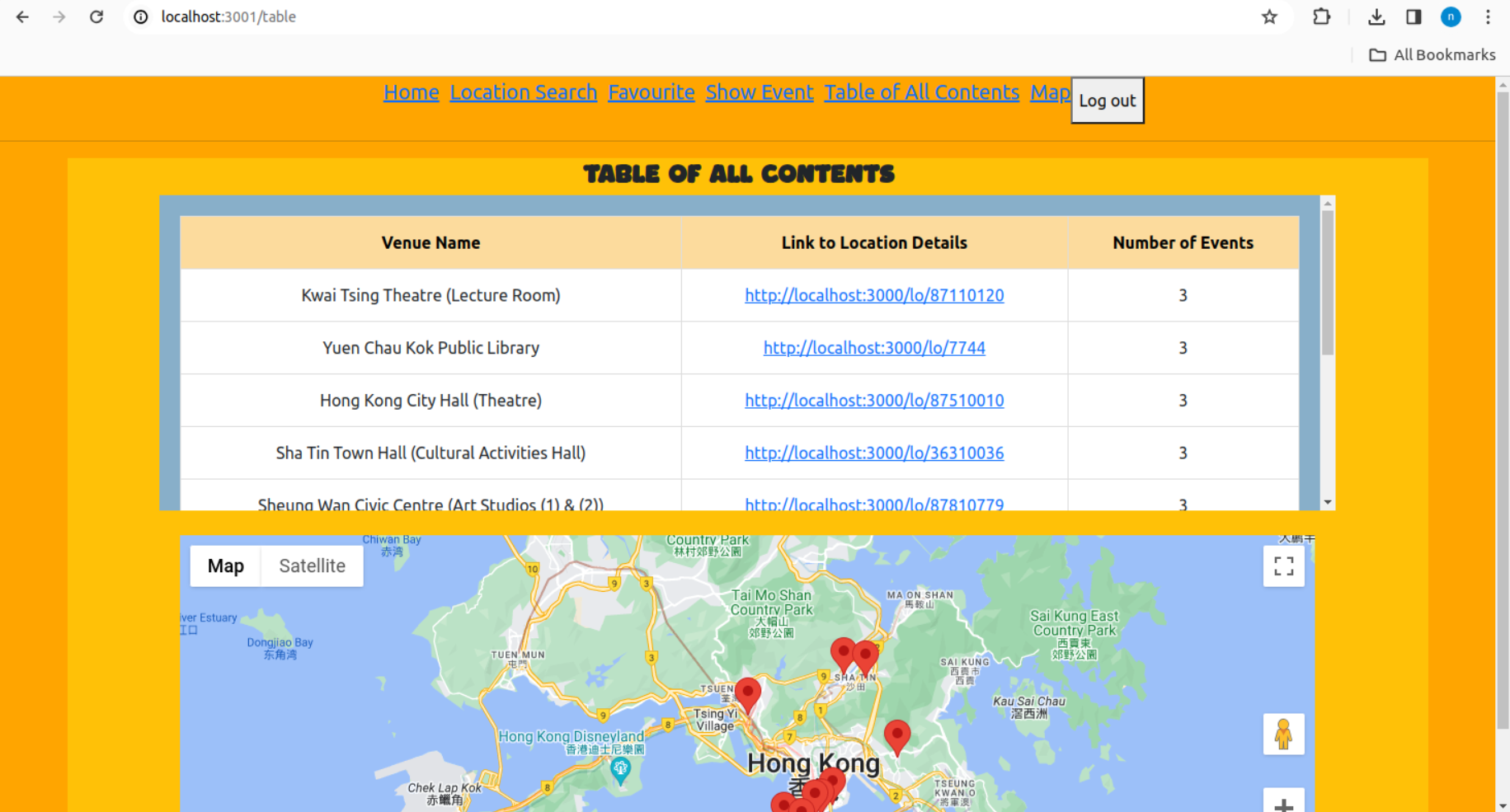
**CSCI2720 Group Project Report**

**Group 11**

**Name & SID:** Cheung Mei Yi 1155159106, Liu Man Yin 1155159567, Ho Yun Kit 1155158328

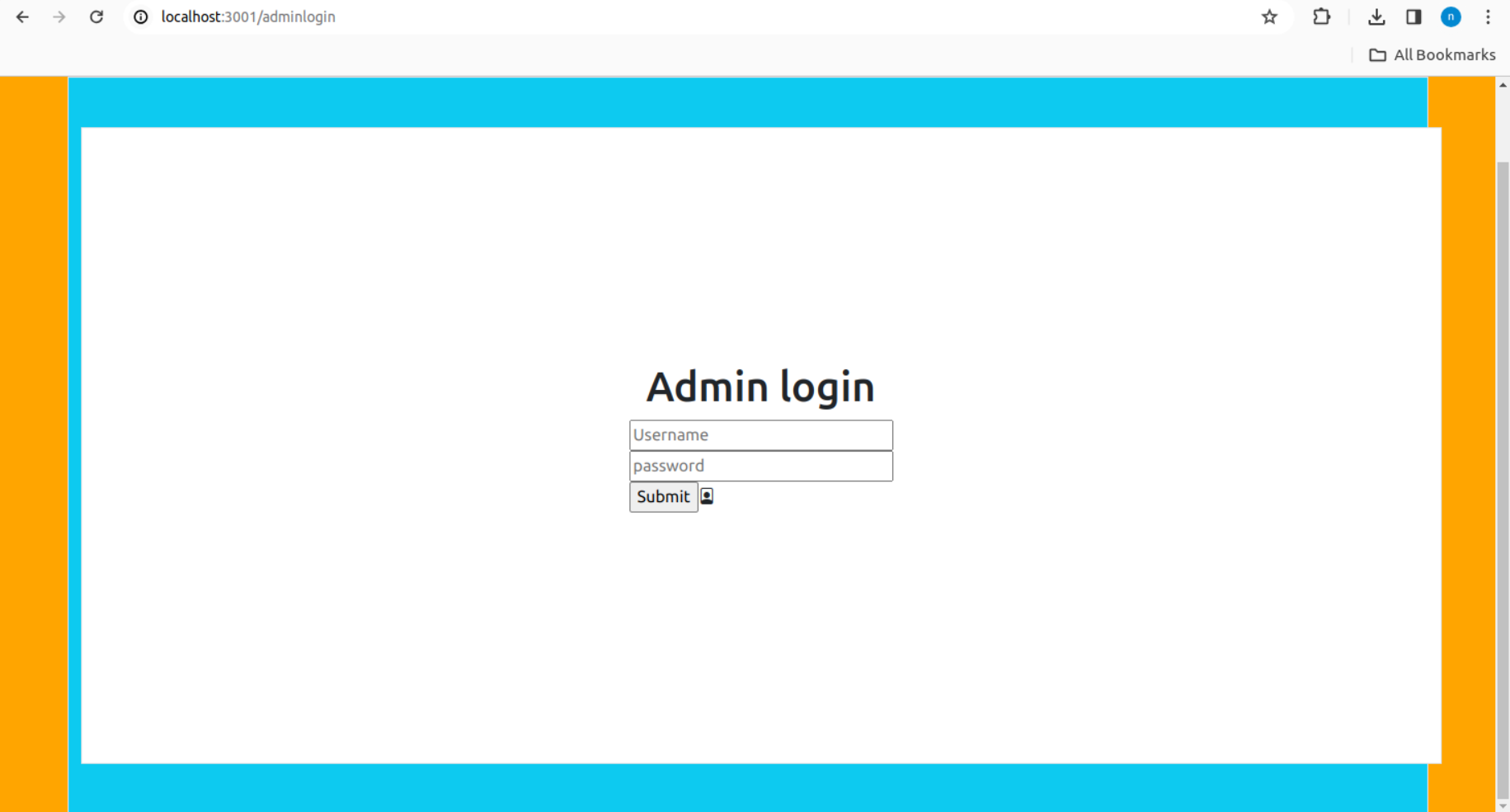
**Abstract**

**The Location App allows people to log in as admin or as user. Once the authentication is done, the user will enter the page with several functions as listed on the top of the navigation bar. The user action would be “Location Search”, “Favourite”, “Show Event”, “Table of All content”, “Map” and “Logout”.**

****

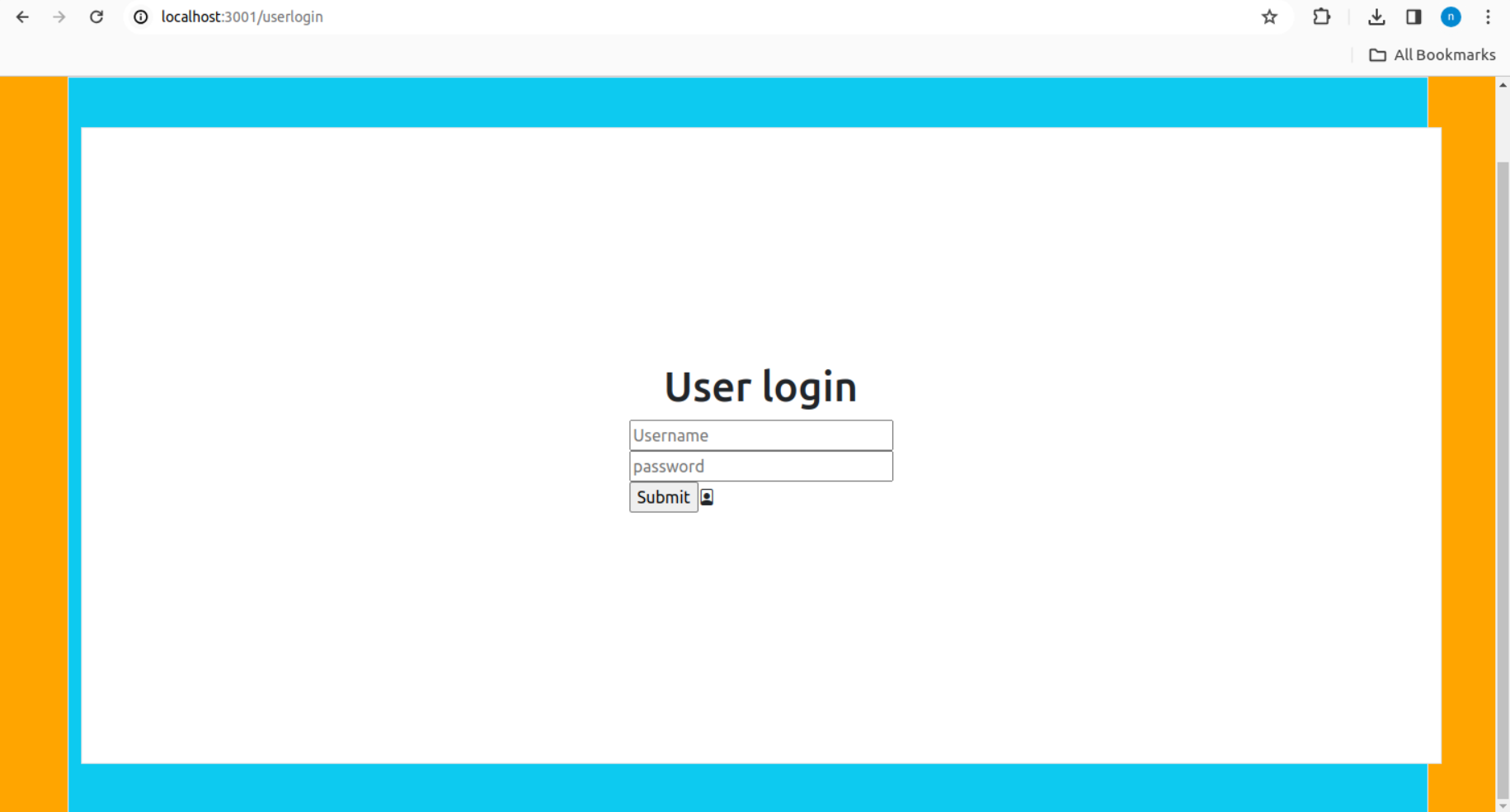
**Admin login**

**Admin may enter username and password, the html form sends a POST request to the backend server and the use of the direction**

****

**User login**

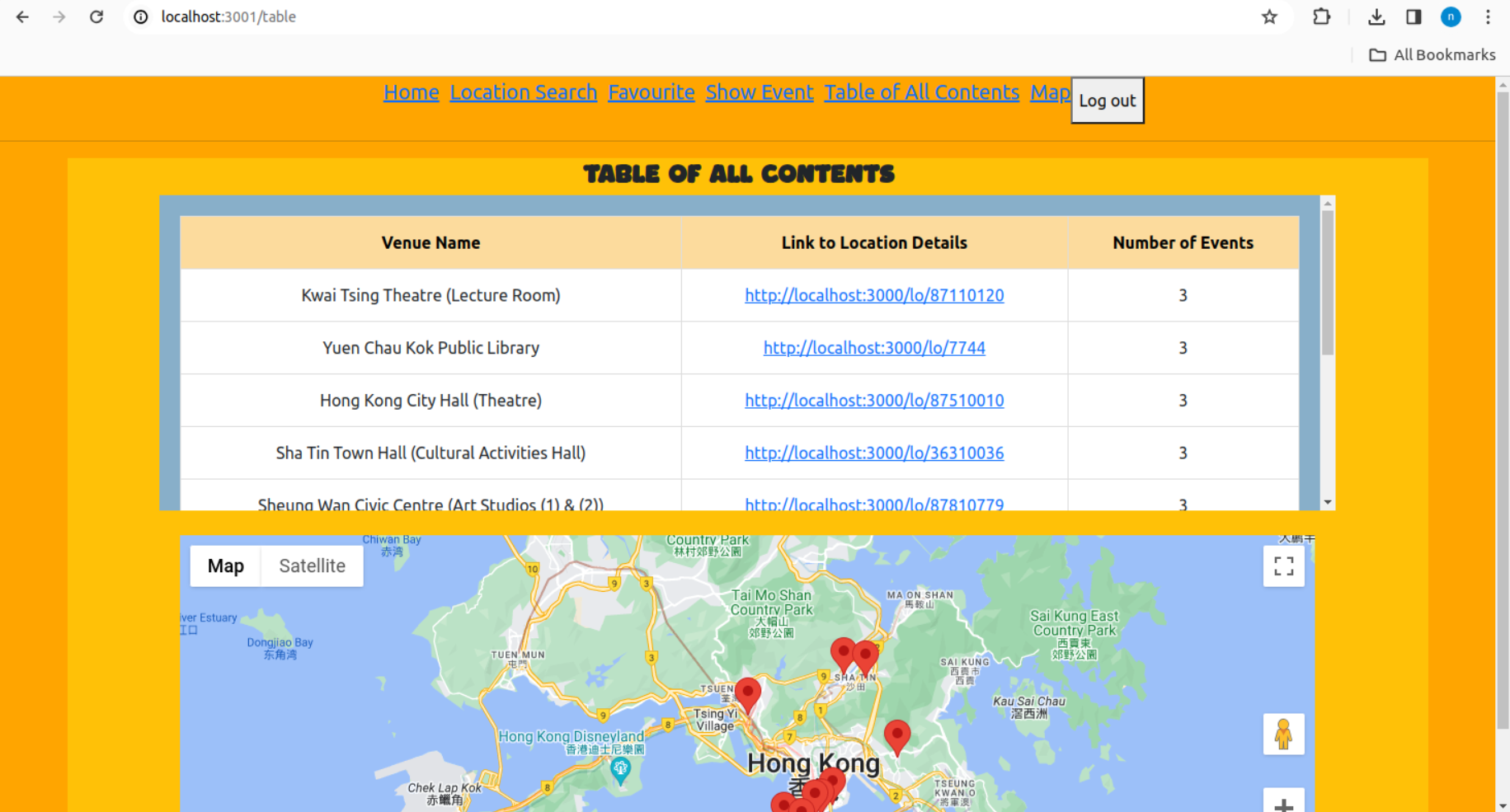
Users may enter their login credential in the webpage. The html form sends the information over the api endpoint in the backend server and the user will be granted a JWT token if successful. If not, error message will be displayed

****

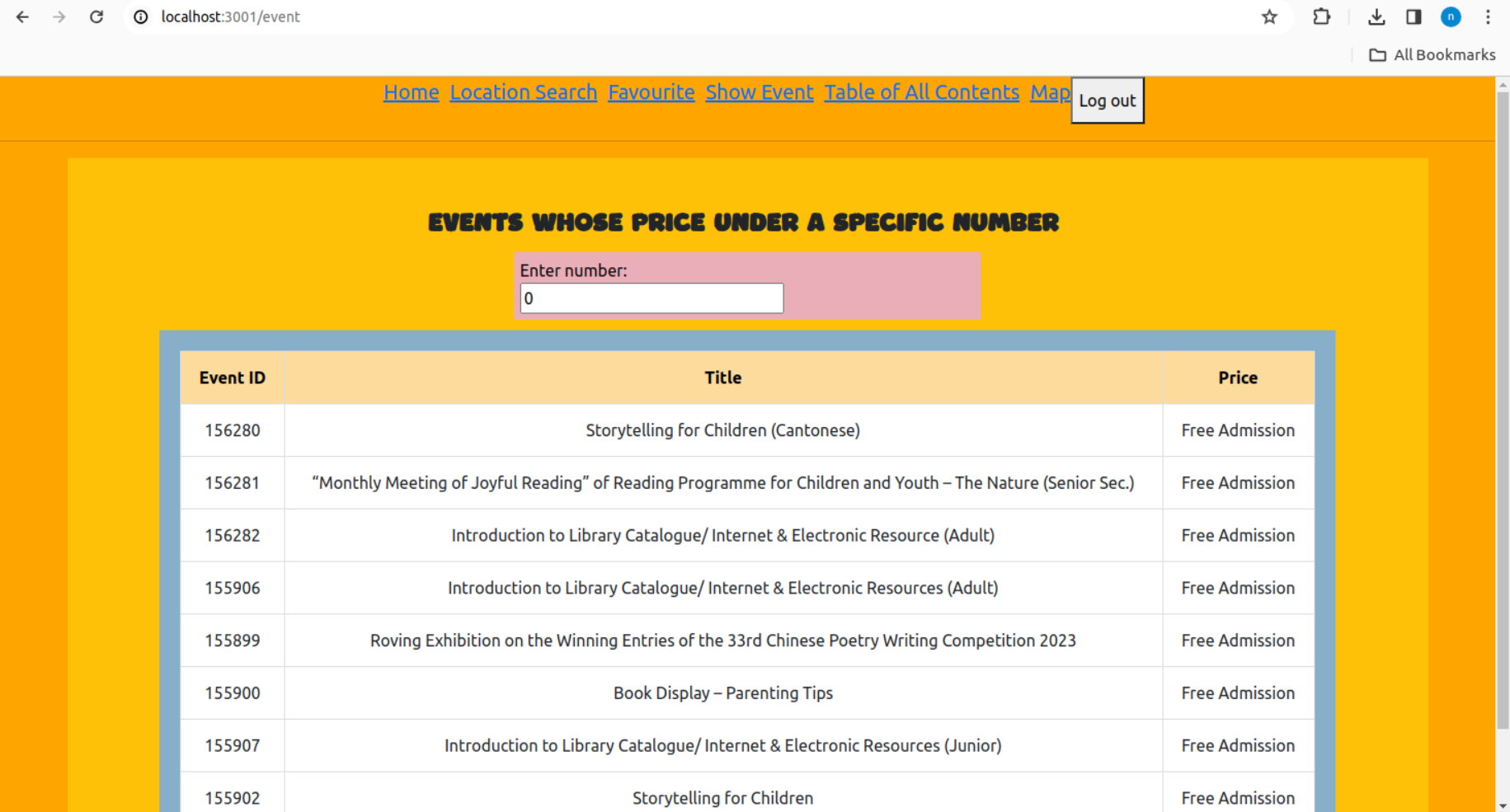
**Table of all locations**

All 10 locations are displayed on the map and the table. User can sort the events by number of events.

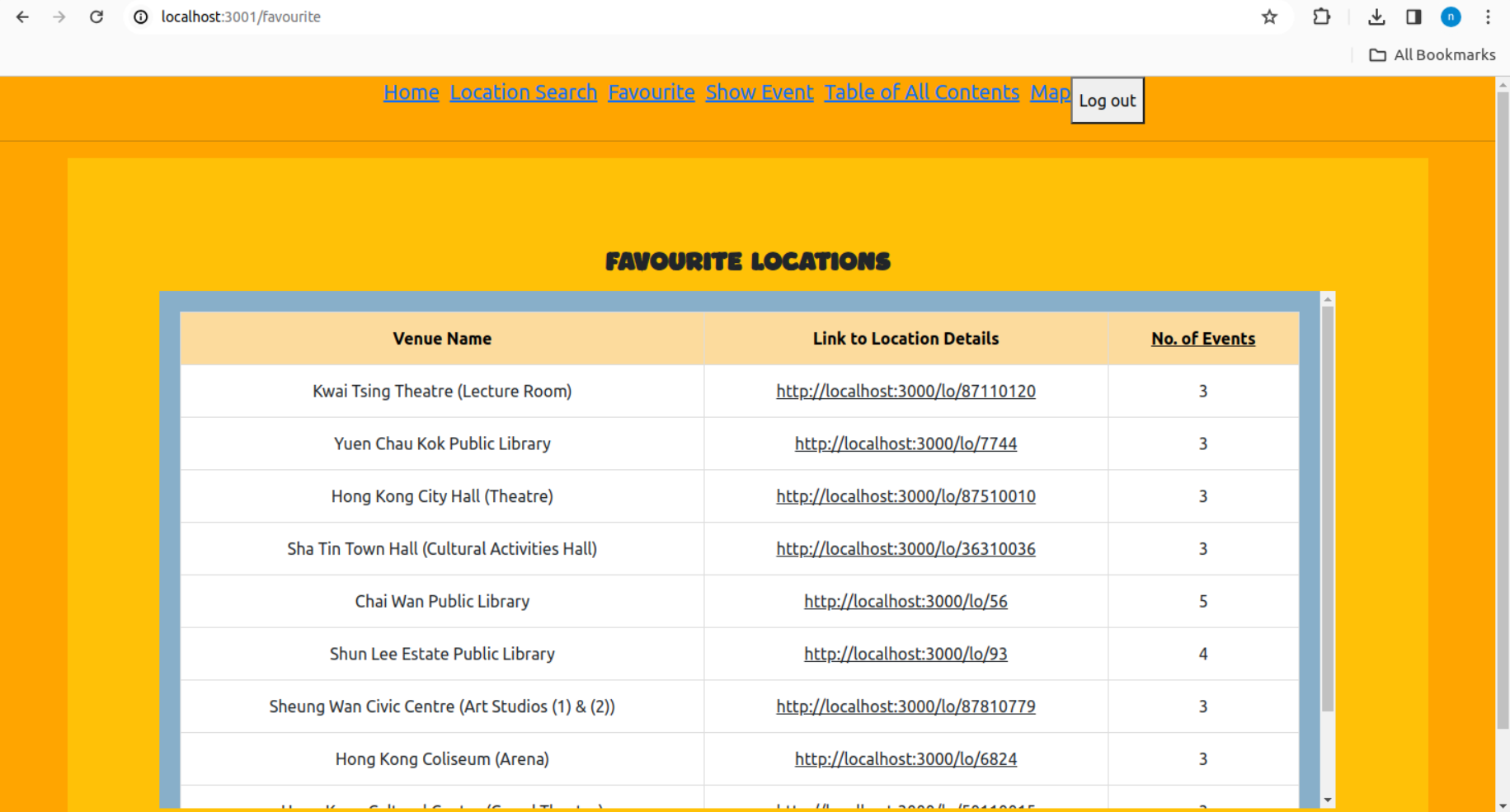
Moreover, users can click on the map and get the link to the location.

****

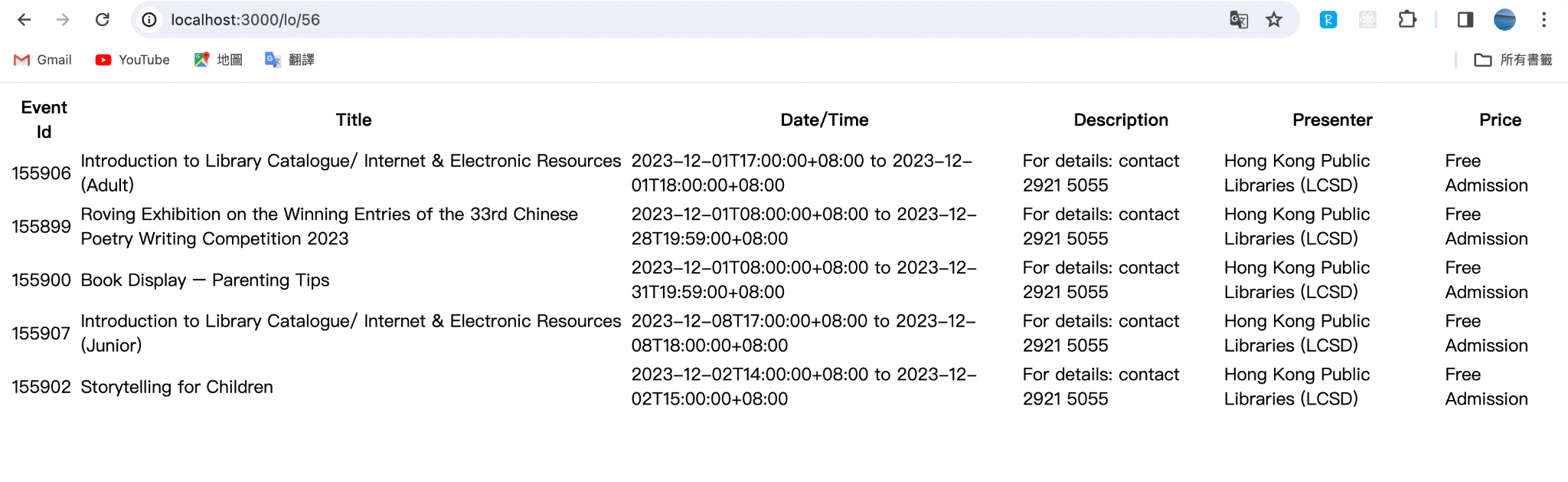
**Event table**

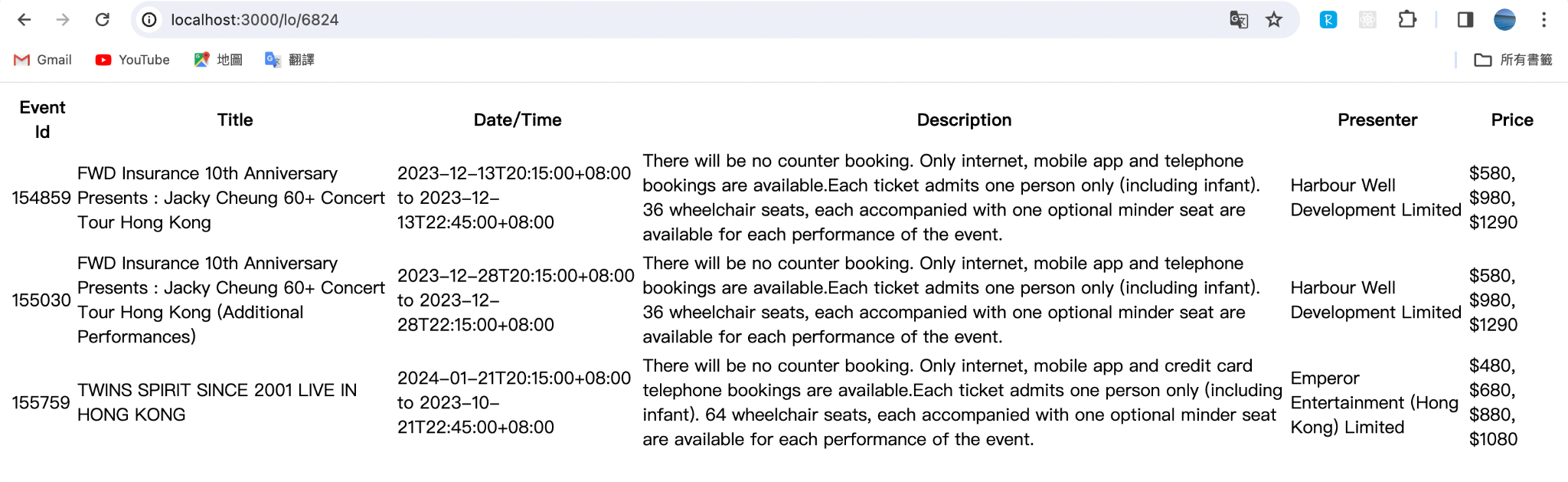
Users can input a number and get the list of events whose price under a specific number.****

**Favourite Locations**

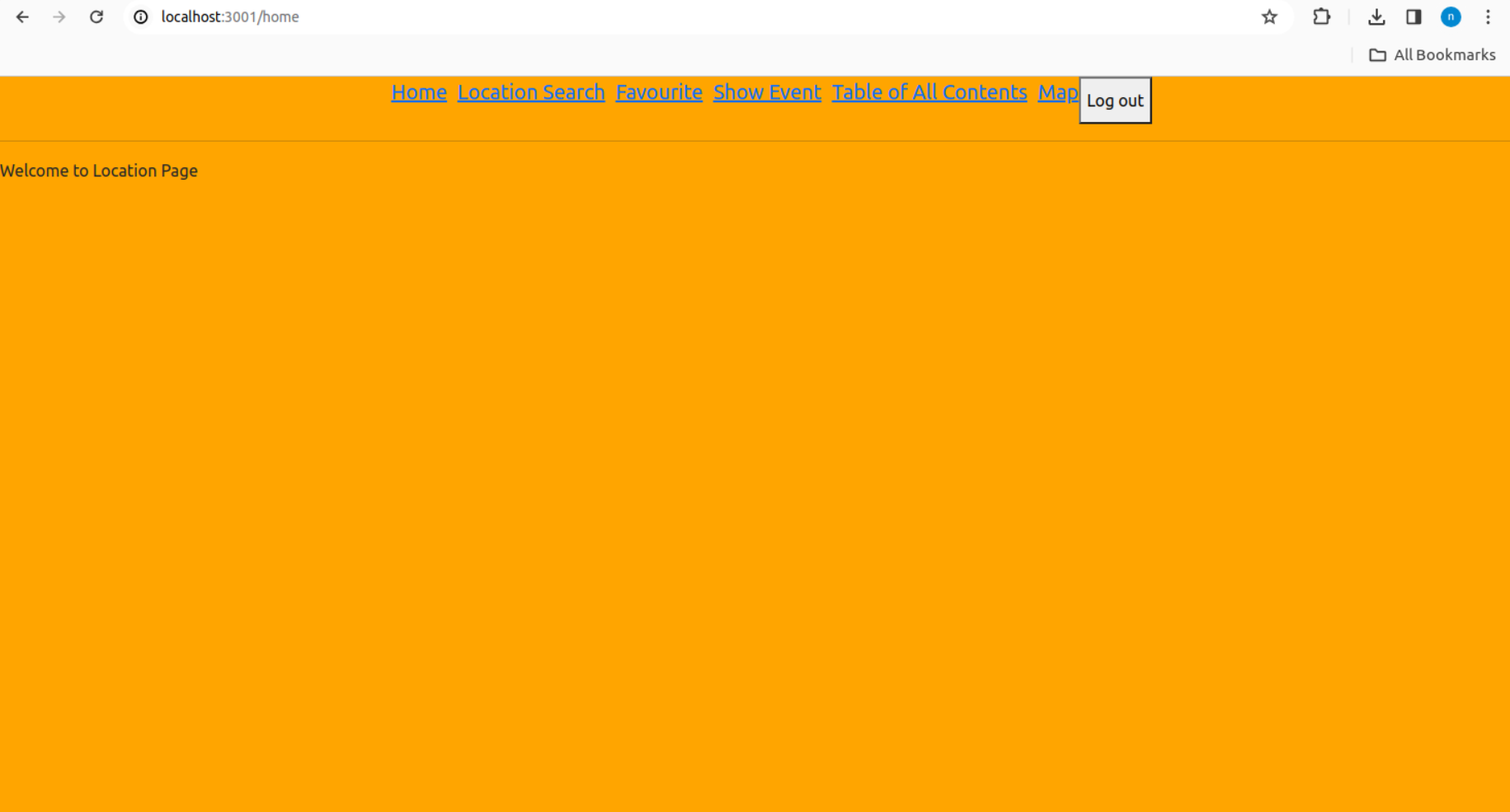
Users can save locations into a favorite locations. So that it will be more convenient to find the location again and skip the tedious searching process.****

**Location Details**

****

****

**Home page**

Home page displaying the navigation bar as well as a welcome message.****

**Methodologies**

2.1 Files submitted

Backend:

* server.js: handles client requests and manipulates location and user data in MongoDB database
* Project.locations.json, Project.events.json, Project.comments.json, Project.users.json, Project.useraccounts.json: database setup
* package.json, package-lock.json

Frontend:

* Index.jsx: provides navigation to different component
* /table/Table.jsx: fetches and displays all location data in a table and on a Google Map
* /table/about.css: defines the styles for Table.jsx component
* /event/Event.jsx: fetches and displays event data based on a specific number
* /event/about.css: defines the styles for Event.jsx component
* /search/Search.jsx: having a search box for keywords, display the result once the input is submitted
* /search/about.css: defines the styles for Search.jsx component
* /favourite/Favourite.jsx: list the favourite location of the user
* /favourite/about.css: defines the styles for Favourite.jsx component
* /SingleMap/SingleMap.jsx: define the Map page
* /authentication/UserLogin.jsx: define the User Login page
* /SingleMap/Comments.jsx: define the area of Comments
* /SingleMap/ListOfEvents.jsx: define the area of Events in a Location

2.2 Dataset preprocessing

* Datasets: Venues of programmes, Programme information from <https://data.gov.hk/en-data/dataset/hk-lcsd-event-event-cultural>
* For preprocessing of the event.xml file, we write python code for the filtering of venues with more than 3 events
* We have chosen 10 venues with ID: 8262, 87110120, 7744, 87510010, 56, 36310036, 93, 87810779, 6824, 50110015
* As MongoDB stores time in UTC by default, we convert the timezone from HongKongTime to UTC when creating new events using moment-timezone, storing the startDateTime and endDateTime with -8 hours
* For some venues in venues.xml, latitude and longitude are not provided, we use Google map data instead in such cases

2.3 Actions on app

User Actions:

* Table of All Contents:

The locations are sorted by the number of events. The sorted locations are displayed in a table. Each row in the table displays the venue name, a link to location details and the number of events. The sorting order can be toggled by clicking the “Number of Events” table header.

The Google Map has markers for each location. When a marker is hovered over, an InfoWindow is displayed with a link to the location details. The InfoWindow is hidden when the mouse is moved away from the marker.

* Show Event:

There is an input field for the user to enter a number for the price. When the number in the input field changes, it updates the event table accordingly. The table displays the event ID, title and price of the events whose price is under the input number.

* View and Write Comment:

Users can view comments after they login to the page. They input their comment on the input box, and then click on the submit button which causes

* View Single Location Map

User can pick a location, then they will be able to view it on map with no other locations. They can also view a table of events in that location, view and post comments.

* View

There is an input field for the user to enter a number for the price. When the number in the input field changes, it updates the event table accordingly. The table displays the event ID, title and price of the events whose price is under the input number.

* Search Location

User can type a keyword in the search box, and the corresponding location details will be shown once the keyword is sent

* View

User can view their favourite location

Admin Actions:

* Create and Delete user

Admin can visit the admin panel, which contains a list of users. The admin can create and delete users.

2.4 Programming languages and algorithms

* The programming languages are mainly React, JavaScript, CSS and HTML

Server side (localhost:3000):

* API endpoints for login/out, CRUD event details, CRUD user data, location search, showing table of all locations, showing table of events details (of the 10 locations), comments, showing user’s favourites locations are as follows:

| **Endpoints** | **Usage/ Functions** |  |
| --- | --- | --- |
| /lo | Fetch all locations, respond with json that contains  1. location name, 2. link to location details page (localhost:3000:/lo/:locationID), 3. no. of events held in that location | GET |
| /lo/:locationID | Fetch events for a specific location, respond with a HTML table with columns: 1. Event Id, 2. Title, 3. Date/Time, 4. Description, 5. Presenter, 6. Price | GET |
| /lo/:locationID/comments | Handle submission of user comments | POST |
| /keywords | Fetch locations with specific keywords in name field, for searching purposes | POST |
| /favourite | Fetch user's favourite locations, respond with json that contains  1. location name, 2. link to location details page (localhost:3000:/lo/:locationID), 3. no. of events held in that location | GET |
| /coordinates | Fetch coordinates and location links | GET |
| /events | Fetch event data with a price lower than or equal to the query price, response with an array of objects containing 1. event ID, 2. title, 3. price | GET |
| /login | Handle login requests,  both user and admin would login through this endpoint, if the user is an admin, the return json field: "isAdmin" would be set to True | POST |
| /newEvent | Handle Admin Event Creation, part of the CRUD requirement  (For Input, note that while some field in the database on Event is optional, it should still be present in the Json body of this post request,  empty string "" can be used for them) | POST |
| /updateEvent | Handle Admin Event Update, part of the CRUD requirement | POST |
| /deleteEvent | Handle Admin Event Deletion, part of the CRUD requirement | POST |
| /newUser | Handle Admin User Creation, part of the CRUD requirement | POST |
| /updateUser | Handle Admin User update, part of the CRUD requirement  (For Input, if there is no change to username and password, empty string "" or falsy object can be passed. However, "oldUsername" and "isAdmin" must be passed correctly) | POST |
| /deleteUser | Handle Admin User delete, part of the CRUD requirement | POST |
| /renewJWT | Handle login token | POST |

2.5 Data schemas and models

| **Models** | **Schemas** |
| --- | --- |
| Location |  |
| Event |  |
| User |  |
| UserAccounts |  |
| Comment |  |

2.6 Libraries and framework

* In server.js, we use express, cors, mongoose, moment-timezone, and jsrsasign libraries
* The frontend design is mostly related to React to create different interfaces. The <App> components consists of a list routing from React routing library

2.7 Comparison of chosen library and other technologies

There are a myriad of benefits of using our React over other javascript libraries such as Vue.js

| React | Vue |
| --- | --- |
| Longer history and more mature | Shorter history and less stable |
| More scalable, larger project can be written easily | Less scalable, development of large scale project is more difficult |

Despite benefits, there are some drawbacks

| React | Vue |
| --- | --- |
| More complex | Simpler, easier to code |
| Larger file size, slower load time | Smaller file size, faster load time |

**Appendix**

Work Distribution

| Testing and coordination between back-end and front-end | Cheung Mei Yi 1155159106 |
| --- | --- |
| Front-end Development: Search, Favourite |
| Front-end Development: User Login, Admin Login, Single Location (Map, Events and Comments) | Rickie |
| Front-end Development: Table of All Contents (table and map) and Show Event | Ho Yun Kit 1155158328 |
| Back-end Development: Data preprocessing, Database manipulation, server side code for user actions (favourite, table of all locations, location details, search, comments) | Liu Man Yin 1155159567 |
| UX/UI Design |
| Back-end Development: server side code for login (JWT token) and CRUD actions of events and users | Alex |