**A Web App for Cultural Programmes Information**

1155158054 LUI Chak Sum, 1155159671 LIU Angus Chak Hei, 1155191857 CHEUNG Hop Cheung

1155191419 Liu Xianlong, 1155144508 ZHENG Cun Hao

1. **Abstract**

This project aims to develop a web application using React for the front end and Node.js with MongoDB for the back end. The application allows users to access information on various locations and cultural programs, with features as specified in the requirement. The application follows a single-page architecture and responsive layout, providing a seamless user experience. We have integrated the Leaflet API to enhance the user interface. This report discusses our methodologies, data preprocessing techniques, programming languages, algorithms, and the design of data schemas and models. We also present a comparison of the advantages and disadvantages of our chosen technologies.

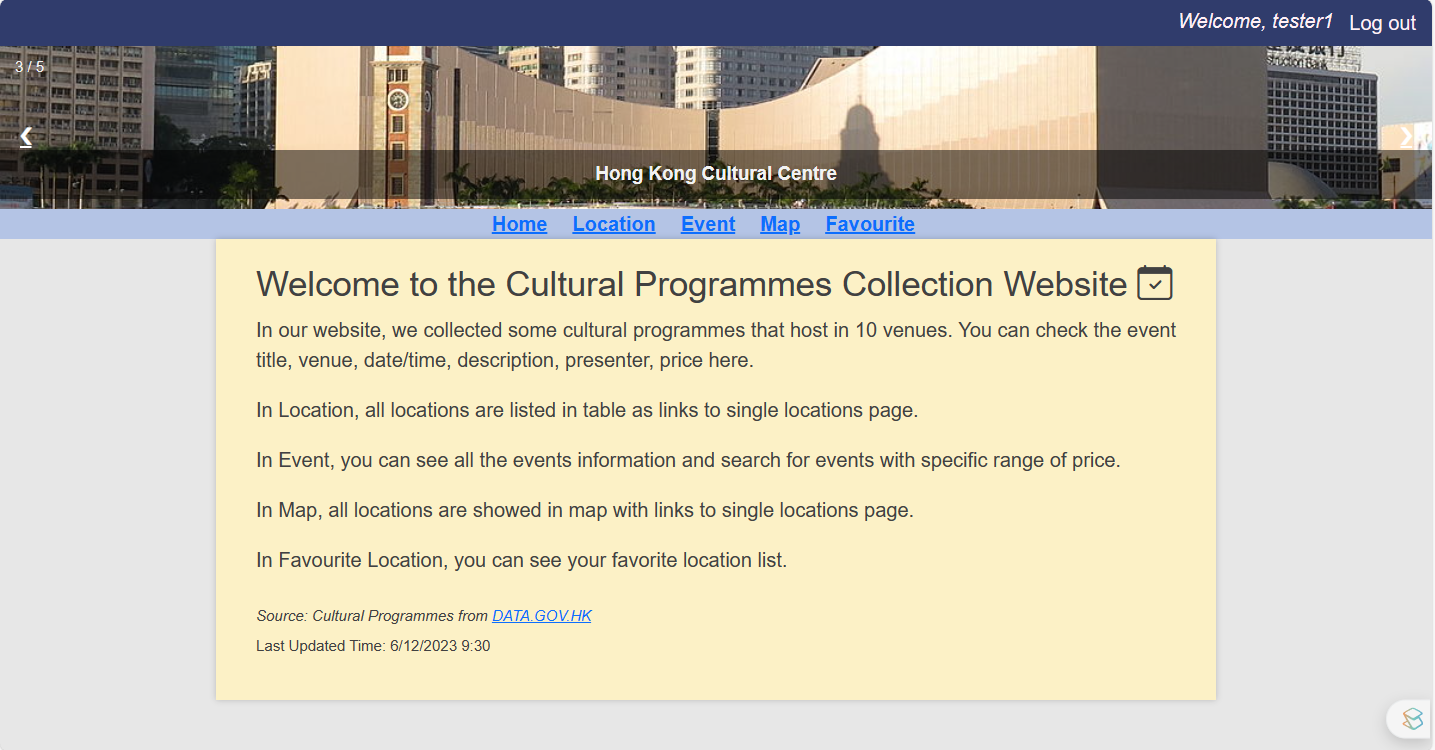


Figure 1: Representative screen of our site

1. **Methodologies**
   1. **Files Submitted**

|  |  |  |
| --- | --- | --- |
| **Path** | **File** | **Description** |
| \ | readme.txt | Commands to start our project server, and the site URL |
| \front-end | package.json | Contains metadata about the project, including dependencies, scripts, and other project-specific configurations for the front-end. |
| \front-end | package-lock.json | Automatically generated by npm to lock down the versions of installed packages and their dependencies. |
| \front-end\public | index.html | A template for index.js to modify the content of web. |
| \front-end\public | favicon.ico | Icon for the browser's tab or bookmark bar for the website. |
| \front-end\public | slides | Pictures for the slideshow in the header. |
| \front-end\src | admin.js | JavaScript file for admin actions export to index.js |
| \front-end\src | icon.png | Icon for the map markers. |
| \front-end\src | index.js | The main JavaScript file for the front end. It includes code for setting up the application, importing other modules, and rendering components. |
| \front-end\src | login.js | JavaScript file for login page export to index.js |
| \front-end\src | style.css | Contains styles and rules to define the visual appearance and layout of the website. |
| \server | server.js | JavaScript file that likely contains code for the server-side implementation, handling API endpoints, and serving the front-end files |
| \server | package.json | Contains metadata about the project, including dependencies, scripts, and other project-specific configurations for the server. |
| \server | package-lock.json | Automatically generated by npm to lock down the versions of installed packages and their dependencies. |

Table 1: Files submitted with description

* 1. **Dataset Pre-processing**

The original attempt of fetching the XML file from the provided source and then parsing it using DOMParser failed. The team instead downloads the latest file and imports it into the system. The XML file is parsed into an array of objects each containing keys: “name”, “price”, “quota”, “locationID”, “dateTime”, “description”, and “presenter”. An ID is also assigned to all of them. The array is then stored in the database for future use.

* 1. **Actions Included**
     1. **Login and Logout**

The website incorporates a login system where users can log in with their credentials. Once the user login with successful authentication, the user is granted access to the website's features.

After logging in as a user, they can utilize the log out button located in the top right corner to log out and navigate back to the login page. Similarly, if logged in as an admin, the user can use the button located in the top left corner to log out.

* + 1. **Admin Dashboard**

Once admin login via login page, they can view all the event detail and user detail in a table. They also edit the detail with the pencil icon or delete with the trash can icon. With the add button, they can add new event or user via the new input role.

一張含有 文字, 螢幕擷取畫面, 軟體, 作業系統 的圖片

自動產生的描述 一張含有 文字, 螢幕擷取畫面, 數字, 平行 的圖片

自動產生的描述

Figure 2: Login page Figure 3: Admin Dashboard

* + 1. **Navigation**

The website employs a navigation menu using the react-router-dom library. The menu allows users to navigate to different sections of the website, such as the home page, location page, event page, map page, and favorite locations page. Each menu item is represented by a link that maps to a specific route defined within the Routes component.

* + 1. **Home Page**

The Home page offers a comprehensive overview of the website's purpose and functionality. Users can easily access information about the website's introduction and the most recent update time of the data.

* + 1. **Location Table**

The location page features a table showcasing various venues where cultural events are held. Each location is presented as a clickable link, enabling users to access a dedicated page with additional details. The component offers sorting options based on event counts at each venue, allowing users to arrange the locations accordingly. Additionally, users have the ability to search for specific locations based on keywords present in their names.

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

Figure 4: Element in Location Page

* + 1. **Map**

The Map page integrates the Leaflet library to display a map with markers representing the locations where cultural events occur. Clicking on a marker can lead users to the associated single location page.

* + 1. **Single Location**

On the Single Location page, users have the option to click a heart icon to add the location to their list of favorite locations. If the heart icon is filled with red color, it indicates that the location is already in the user's favorite list. Additionally, users can view the location of the venue on a map and explore the events being held at that specific venue. Furthermore, users can leave comments and share their thoughts about the venue on this page.

一張含有 地圖, 地圖集, 文字 的圖片

自動產生的描述 **一張含有 文字, 地圖, 螢幕擷取畫面, 圖表 的圖片

自動產生的描述**

Figure 5: Element in Map Page Figure 6: Element in Event Page

* + 1. **Event Table**

The Event page presents a list of cultural events. Users can view details such as event titles, venues, dates/times, descriptions, presenters, and prices. The page may also include filter functionality to allow users to filter events based on price range.

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述

Figure 7: Element in Event Page

* + 1. **Favorite Location**

The Favorite Location page displays a list of the user's favorite locations, allowing them to manage their selections.

一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述

Figure 8: Element in Event Page

* + 1. **Slideshow (Special Feature)**

A slideshow featuring pictures of various venues is displayed in the header of the page, allowing users to visualize the appearance of different buildings. Users can navigate through the slideshow by clicking the left and right arrows, which enable them to move to the previous or next picture, respectively. This interactive feature provides users with a visual representation of the venues.

一張含有 建築, 螢幕擷取畫面, 商業建築, 高層建築物 的圖片

自動產生的描述

Figure 9: Slideshow in Header

* 1. **Programming Languages and Important Algorithms**

JavaScript has been chosen as the sole programming language for this project due to its ability to provide dynamic and interactive experiences. As a client-side scripting language, JavaScript runs directly in the user's web browser, enabling real-time manipulation of web page elements and immediate feedback to user actions. With JavaScript, we, as developers, can create features like form validation, dynamic content updates, and interactive user interfaces, which align with the requirements of this project. Additionally, JavaScript benefits from a vast ecosystem of libraries and frameworks, including React, which is implemented in this project, simplifying, and expediting web development. Its wide adoption, browser compatibility, asynchronous capabilities, and flexibility make it an ideal choice for building modern, responsive, and engaging web applications [1].

One of the significant algorithms included in this project is the User Interface (UI) Rendering Algorithm, which involves React components that define the application's UI. With these UI algorithms, we can efficiently update the user interface based on changes in the application's state, providing a seamless and responsive user experience. The use of pagination view in single-page applications (SPAs) helps manage large data sets and improves performance by loading and displaying data in smaller, manageable chunks [2].

* 1. **Data Schemas and Models**

Here is a picture including all the schema and models of this project. Notice that whenever a new account is created, a user will be created with its ID being the maximum of all the users. Each of the schemas has a unique ID, which makes it easier to find and process. In CRUD part, we used a generic programming style to make the code more good-looking and easier to understand.

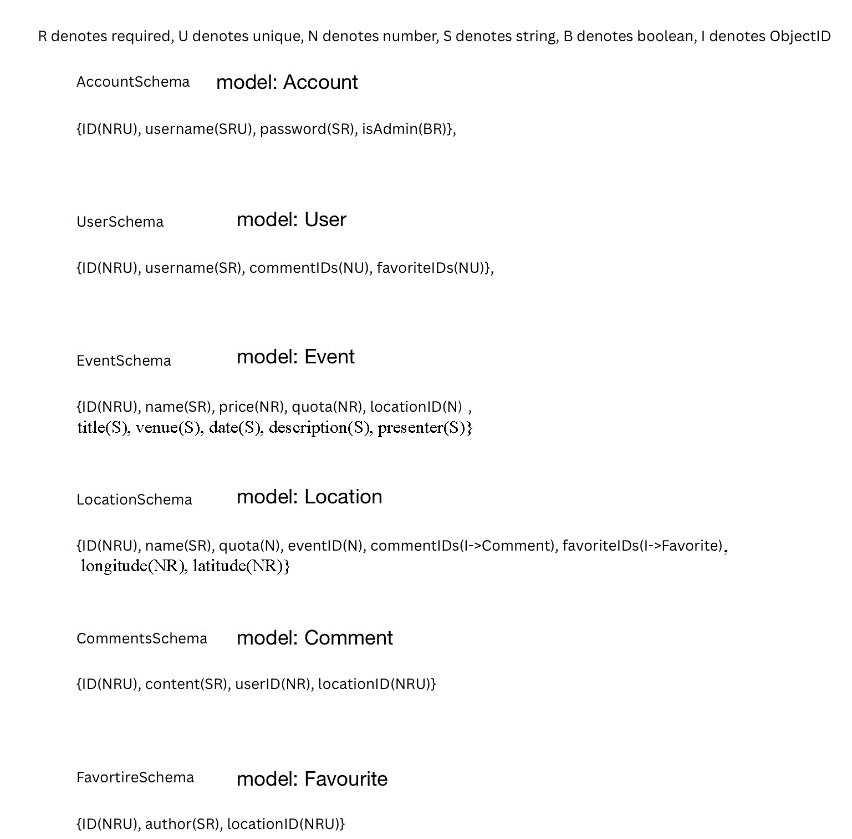


Figure 10: Data Schema and Model Structure

* 1. **Comparison**

|  |  |
| --- | --- |
| **React** | **Angular** |
| Higher performance since Virtual DOM render much faster than Real DOM. | Lower performance since it uses Real DOM. |
| There is flexibility in adding and integrating various tools and libraries. | While it has some tools and libraries available, the options are comparatively narrower. |
| One-way data binding updates the view automatically based on changes in the model, but it lacks automatic synchronization from the view back to the model, requiring additional manual effort for synchronization. | Two-way data binding simplifies code by automatically updating the model and view whenever a change occurs, keeping them in sync. |
| React lack of built-in TypeScript support may require additional configuration and setup for projects that prefer static typing. | Angular reduces coding time and enables the creation of fast apps, while TypeScript helps identify and eliminate errors early in the development cycle, making it more efficient. |

Table 2: Advantages and Disadvantages about React over Angular [3]

|  |  |
| --- | --- |
| **MongoDB** | **MySQL** |
| Non-relational database, more dynamic, lesser restriction | Relational database, enforce schema with strict rules |
| MongoDB has more scalability by using replica sets and sharding. | Only provide vertical scalability by adding more resources to the current database server. |
| keep most of the data in a single document, Joins are not optimized for performance. | MySQL is designed to enact high-performance joins across multiple tables that are appropriately indexed. |
| MongoDB excels at inserting or updating a large number of records. | MySQL is faster when selecting a large number of records. |

Table 3: Advantages and Disadvantages about MongoDB over MySQL [4]

1. **References**

[1] C. Tsang, *CSCI2720 Lecture 05 JavaScript*. [Online]. Available: <http://tinyurl.com/yu3m9szj>

[2] C. Tsang, *CSCI2720 Lecture 02 SPA & Rounting*. [Online]. Available: <http://tinyurl.com/ytvut7pa>

[3] Kinsta, *Angular vs React: A Detailed Side-by-Side Comparison.* [Online]. Available:

<https://kinsta.com/blog/angular-vs-react/>

[4] AWS, *What’s the difference between MongoDB and MySQL.* [Online]. Available:

<https://aws.amazon.com/compare/the-difference-between-mongodb-vs-mysql/>

1. **Appendix**
2. **Workload Distribution**

|  |  |  |
| --- | --- | --- |
| **Person-In-Charge** | **Workload** | |
| Lui Chak Sum, Liu Angus Chak Hei | User actions with CSS |
| Cheung Hop Cheung | Action actions with CSS |
| Liu Xianlong, ZHENG Cun Hao | Backend and data handling |