Software Project Management Plan

Project Name: MayMap

Date: 10 July 2023

Revision: Version 1.0

Author: Amirul Aiman Bin Rohaimi

Summary: The MayMap is a user-friendly application created with React.js, utilizing the incredible capabilities of the Google Places API. It empowers users to effortlessly explore and visualize various locations on an interactive map. This remarkable application offers convenient features like an intuitive autocomplete search function, instant access to search results from the API, and smooth integration with React.js' server-side rendering.

Revision Page

a. Overview

The Software Project Management Plan (SPMP) provides a comprehensive framework for managing the development and delivery of the MayMap. This document outlines the project scope, objectives, schedule, resources, and deliverables. This plan serves as a roadmap for the project to follow and provides a foundation for monitoring and controlling the project to ensure that it is completed within scope and on time.

b. Target Audience

MayMap targets travelers, local explorers, tourist agencies, event organizers, and developers/tech enthusiasts. It aims to provide a seamless map-based experience powered by the Google Places API, allowing users to search and display places of interest, plan itineraries, discover local attractions, and navigate events.

c. Project Team Members

Name	Role/Title	Email	Sign-Off
Amirul Aiman Rohaimi	Project Manager, Software Development Developer	amirulaimanrohaimi@gmail.com	[illegible]

d. Version Control History

Version #	**Status**	**Date**	**Section, Page(s) and Text Revised**	**Modified by**
1.0	Initial	10-July-2023	SPMP v1	Amirul

Software Project Management Plan

This portion of the paper introduces a proposal to create and construct the "MayMap - World Map" project's software development. It will include the project's overall goals and the objectives that must be met, the project's scope and the deliverables it aims to produce, the project's assumptions and limits, certain commitment, and a quick rundown of the project's timetable.

1. Project Summary

Project Name	MayMap
Project Leader	Amirul Aiman Rohaimi
Project Start Date	July 10, 2023
Project End Date	July 11, 2023
Project Type (Development, Feasibility)	Web Application Development
Platform	Microsoft Visual Studio Code, Netlify
Live Link	https://maymap-app.netlify.app/
Repository	https://github.com/amirulaimanr/MayMap

1.1 Project Objectives and Scope

1.1.1 Purpose

The purpose of the MayMap project is to create a single-page application using React.js that integrates the Google Places API to enable users to search for places and display them on an interactive map.

1.1.2 Objectives

The objective of this project is:

1. to build a highly functional and user-friendly map application that seamlessly integrates the Google Places API.
2. to showcase server-side functionality, through React.js' server-side rendering.
3. to handle API calls on the server and efficiently pass data to client components.

1.1.3 Project Scopes

The project scope focuses on delivering a map application that integrates the Google Places API, provides a user-friendly search experience, displays search results accurately on a map, and leverages the features of React.js with server-side functionality. The emphasis is on creating a robust, scalable, and efficient solution that meets the requirements of the technical assessment.

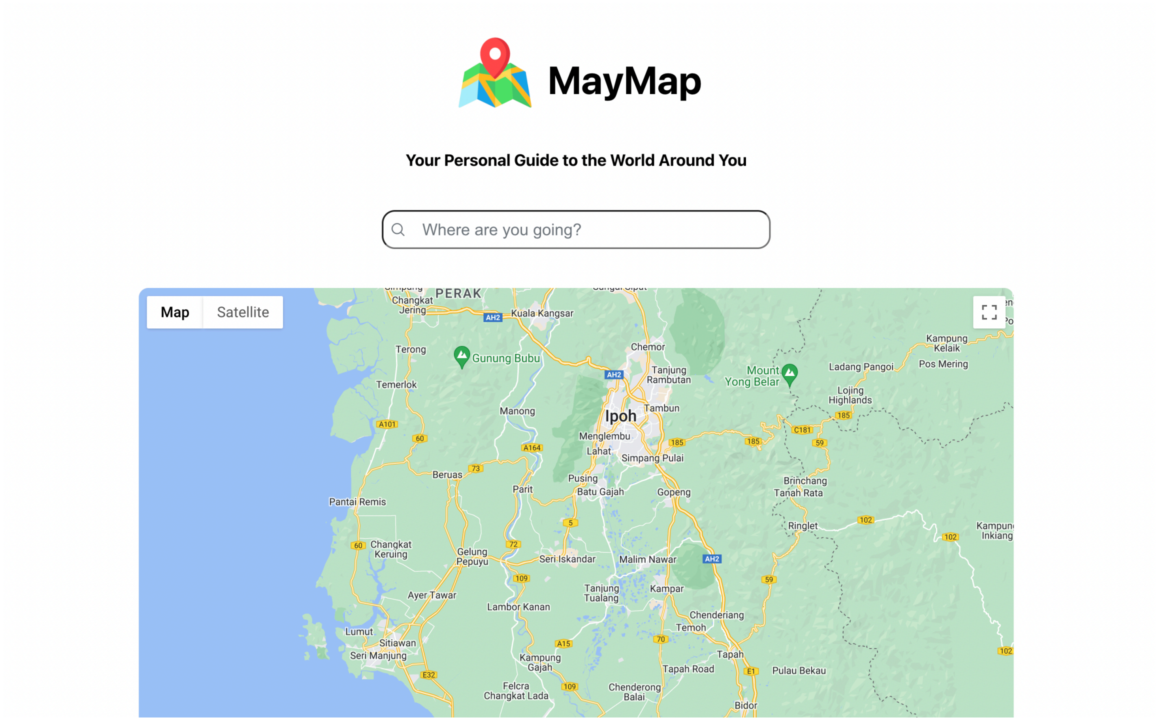
2 Development Environment

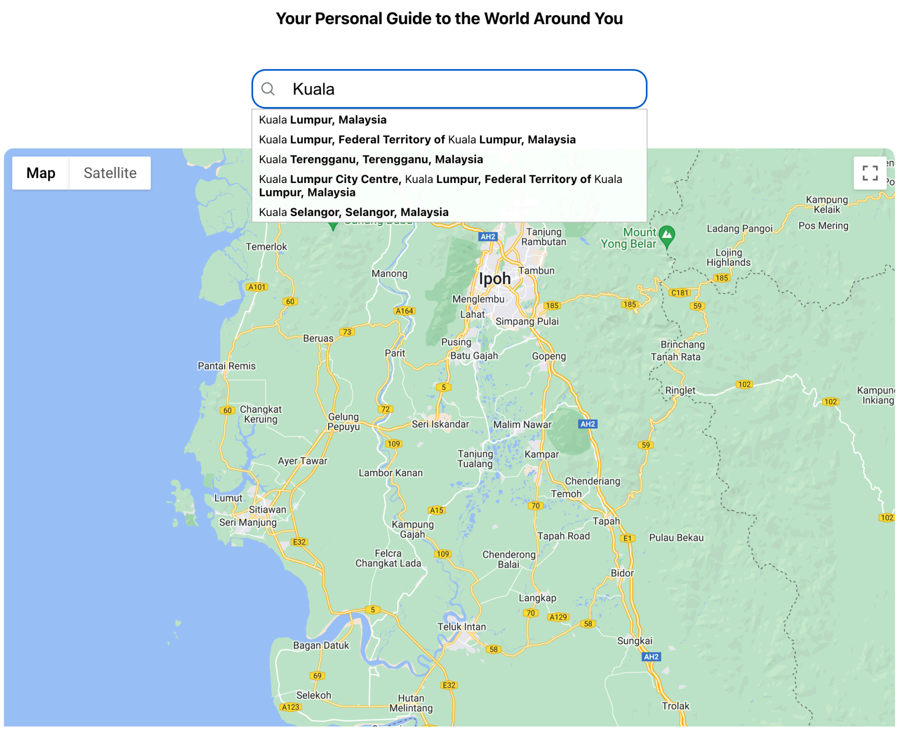
2.1 Software and Tools Requirements

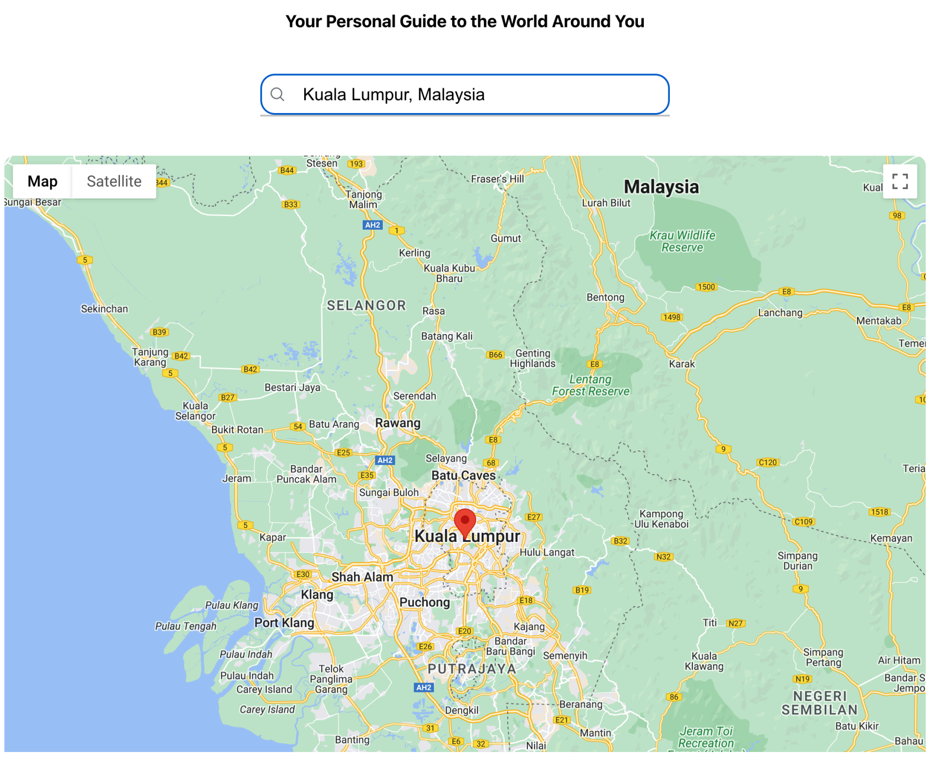
Development	**Frontend**	**Backend**	**Database**
IDE	Vscode	Vscode	-
Code Language	Javascript	Nodejs	-
Framework	ReactJs (Primereact)	ExpressJs	-
Deploy & Hosting	Netlify	Netlify	-
Version control	Git	Git	-
API	-	Geocoding API Google Places API Maps Javascript API	-

2.2 System Interfaces

- **Main page**


MayMap
Your Personal Guide to the World Around You
Where are you going?
Map
Satellite
PERAK
Ipoh


Your Personal Guide to the World Around You
Kuala
Kuala Lumpur, Malaysia
Kuala Lumpur, Federal Territory of Kuala Lumpur, Malaysia
Kuala Terengganu, Terengganu, Malaysia
Kuala Lumpur City Centre, Kuala Lumpur, Federal Territory of Kuala Lumpur, Malaysia
Kuala Selangor, Selangor, Malaysia
Map
Satellite
Ipoh



With the MayMap web application, users can embark on a range of activities to explore and navigate various places effectively. Users have the ability to search for specific locations, such as restaurants, attractions, or hotels, using the autocomplete search feature powered by the Google Places API. The application then displays the search results on an interactive map, allowing users to visually explore the surrounding area and get a clear overview of nearby points of interest. This empowers users to plan trips, discover new places, find local favourites, and create personalized itineraries, making MayMap a versatile tool for both travellers seeking new adventures and local explorers looking to uncover hidden gems in their vicinity.

3 Functional and non-functional Requirements

3.1 Functional specifications:

- Place Search and Autocomplete

Users can search for places by entering keywords in the search box. The search box should provide autocomplete suggestions based on the user's input. The application should retrieve search results from the Google Places API and display them on the map.

- Map Display and Interaction

The application should have an interactive map interface to display search results. Places retrieved from the API should be marked with pins or markers on the map. Users should be able to zoom in/out and pan the map.

3.2 Non-functional specifications:

- Performance:

The application should load quickly and provide a smooth user experience, even with a large number of markers or search results. The map and search functionality should be responsive and perform well across different devices and network conditions

- Security:

API keys and other sensitive information should be securely stored and protected from unauthorized access.

- Scalability:

The application should be designed to handle a growing number of users, search queries, and saved places. The system should be scalable, capable of handling increased traffic and load without compromising performance.

- Code Quality and Maintainability:

The application code should follow best practices, be well-structured, modular, and maintainable. Proper documentation and comments should be provided to aid future development and maintenance.

4 Specific Requirements

4.1 Purpose of using ReactJS

1. **Server-Side Rendering (SSR):** React.js has a feature called server-side rendering (SSR), which helps speed up how quickly web pages load initially and improves their visibility on search engines.

2. **React Ecosystem:** React.js is backed by a vibrant and active community, offering a wide range of third-party libraries and tools. These libraries enhance the development process by adding extra features and utilities. For instance, there are libraries specifically created for seamlessly integrating mapping APIs, making it easier to integrate the Google Places API and display maps within the MayMap application.

3. **Component-Based Architecture:** React.js follows a component-based architecture, allowing developers to break down the user interface into reusable and independent building blocks. This approach promotes code reusability, easier maintenance, and simplifies the development process. As a result, it becomes more efficient to construct and manage complex UI elements in the application.

4.2 Different of using Server Side Render

1. **Faster Initial Page Loading:** When using traditional client-side rendering, the browser must download and execute the JavaScript bundle before displaying the

initial content. This can introduce delays, especially with large bundle sizes or slower internet connections. In contrast, SSR serves pre-rendered HTML from the server, allowing the browser to start rendering immediately. This means users can see the content more quickly without waiting for JavaScript to load and execute.

2. **Improved Search Engine Visibility:** Search engines rely on HTML content to index and rank websites. Client-side rendered applications often have limited initial HTML content, making it challenging for search engines to understand and rank the application's content. SSR resolves this issue by providing fully rendered HTML during the initial request. This enables search engines to easily crawl and index the MayMap application, boosting its visibility in search results.

3. **Enhanced User Experience:** SSR contributes to a better user experience by delivering content faster and creating a more responsive application. Users no longer need to wait for the JavaScript bundle to download and execute before accessing meaningful content. Additionally, SSR benefits users with slower internet connections or less powerful devices, as they may encounter performance issues with client-side rendering.

4.3 Handle error and failures when making API Calls

1. **Implement Error Boundaries:** Wrap components responsible for making API calls to the Google Places API with Error Boundary components. These Error Boundaries efficiently capture errors within their child components and provide the

ability to display a fallback user interface instead of experiencing a complete application crash.

2. **Manage API Call Errors:** Employ try-catch blocks or promises with catch handlers to effectively handle errors encountered during API calls. Enclose the API call within a try block and capture any errors that may occur. Within the catch block, update the error state variable with the specific error information.

3. **Present Error Messages:** Within the component's render method, examine the error state variable. If an error has occurred, present an appropriate error message or user interface component to notify the user about the encountered issue.

4.4 Process of setting up environment variables securely

1. **Create a .env file**: In the root directory of React.js project, this file will be used to store environment variables.

2. **Define the environment variable**: In the .env file, define the sensitive API key as an environment variable. For example, REACT_APP_GOOGLE_MAPS_API_KEY=api_key.

3. **Add .env to .gitignore:** To ensure that the .env file is not committed to version control system (e.g., Git), add .env to project's .gitignore file. This prevents exposing sensitive API key in repository.

4. **Accessing environment variables in the React component**: In React component, access the environment variables by using process.env followed by the variable name. For example, process.env.REACT_APP_GOOGLE_MAPS_API_KEY.

5. **Build and Deployment:** When building or deploying this application, environment variables will be configure depending on deployment platform.

4.5 Handle user authentication and authorization if were required

1. **User Authentication:** Implement a user authentication system using a secure authentication mechanism, such as JWT (JSON Web Tokens) or OAuth.

2. **Validate the user's credentials**: on the server-side and generate an authentication token upon successful authentication. Store the token securely, such as in browser cookies or local storage, for subsequent API requests.

3. **Protected Routes:** Create a higher-order component (HOC) or a custom Route component to protect specific routes in the application that require authentication. When a user tries to access a protected route, check if they have a valid authentication token stored. If not, redirect them to the login page. If the user has

a valid token, allow access to the protected route and render the corresponding component.

4. **User Authorization:** Define user roles or permissions that determine what actions and features each user can access. Store the user's role or permissions alongside their authentication token. Within each component or route, check the user's role or permissions before allowing certain actions or displaying specific content.

4.6 Step to ensure the deployment process is smooth and efficient

1. **Production Optimization:** Optimize React.js application for production by minimizing bundle sizes, enabling code splitting, and implementing techniques like tree shaking and compression.

2. **Build Process:** Set up a build process to generate a production-ready build of React.js application. This typically involves using tools like webpack or create-react-app's build script. Ensure that all necessary dependencies are properly installed and included in the build process.

3. **Environment Configuration:** Set up environment variables for the production environment, including sensitive data such as API keys, database credentials, or server URLs. Ensure that the environment variables are securely stored and accessible during the deployment process.

4. **Deployment Platform:** Choose a deployment platform that suits the requirements, such as AWS Amplify, Netlify, Vercel, or Firebase. Set up the necessary accounts and configurations for the chosen platform. Configure the deployment platform to pull the code from version control system (e.g., GitHub, Bitbucket).

4.7 Optimize performance

1. **Code Splitting:** Utilize code splitting to divide React.js application into smaller chunks, allowing for lazy loading of components and reducing the initial bundle size.

2. **Bundle Size Optimization:** Minimize the bundle size by eliminating unnecessary dependencies, using smaller alternative libraries or components, and leveraging tree shaking to remove unused code.

3. **Performance Testing and Optimization:** Test the performance of this application using tools like Lighthouse to identify bottlenecks and areas for improvement. Optimize critical rendering path elements like reducing render-blocking resources (CSS and JavaScript), optimizing images, and leveraging browser caching.

4. **Memoization and Pure Components:** Implement memoization techniques such as memo or React.memo to prevent unnecessary re-renders of components. Use

pure components that do not depend on external state or props changes to avoid unnecessary rendering cycles.

4.8 Testing methodologies

1. **Unit Testing:** Unit tests focus on testing individual components in isolation. For React.js, I would utilize testing frameworks like Jest, along with libraries like React Testing Library or Enzyme, to write unit tests. Unit tests ensure that each component functions correctly and handles various scenarios and props as expected.

5. References

Appendix A: Gantt Chart

	10/07	11/07
Planning	■	
Analysis Software Requirement	■	
Design Process	■	
Development	■	■
Testing	■	■
Deployment		■
Review		■
Documentation		■