Architecture Design Document for

Student Clubs Event Management Platform

This is the proposed architecture for the student's event management portal, the requirements of which have been given earlier. For this application there are three key stakeholders: students, club coordinators, and an administrator. Each of them have their use cases. As it is meant for students of the institution, there should be proper authentication using the institution's email id. The application is to be available on the web, and should have a reasonable interface and response time for each of the use cases, as specified in the requirements.

1. Architecture Design

The application will follow the variation of the MVC architecture style in which the front-end code is a set of html files with javascript code embedded in them which can make calls to some back-end APIs. The back-end API just returns a JSON object, and the front-end code updates part of the currently rendered page with the information in the JSON objects. Hence, in this architecture, we can say that the front-end component contains some files which are used for rendering a page to the user, and the backend component has the code for executing the APIs for providing information for parts of the front-end pages.

A request from the browser of the user comes to the webserver. If it is a front-end file, the webserver fetches the file and sends it back to the browser for rendering. If the request is for an API call, then the webserver forwards it to the back-end. The back-end has multiple components. The overall architecture is shown in the figure xx.

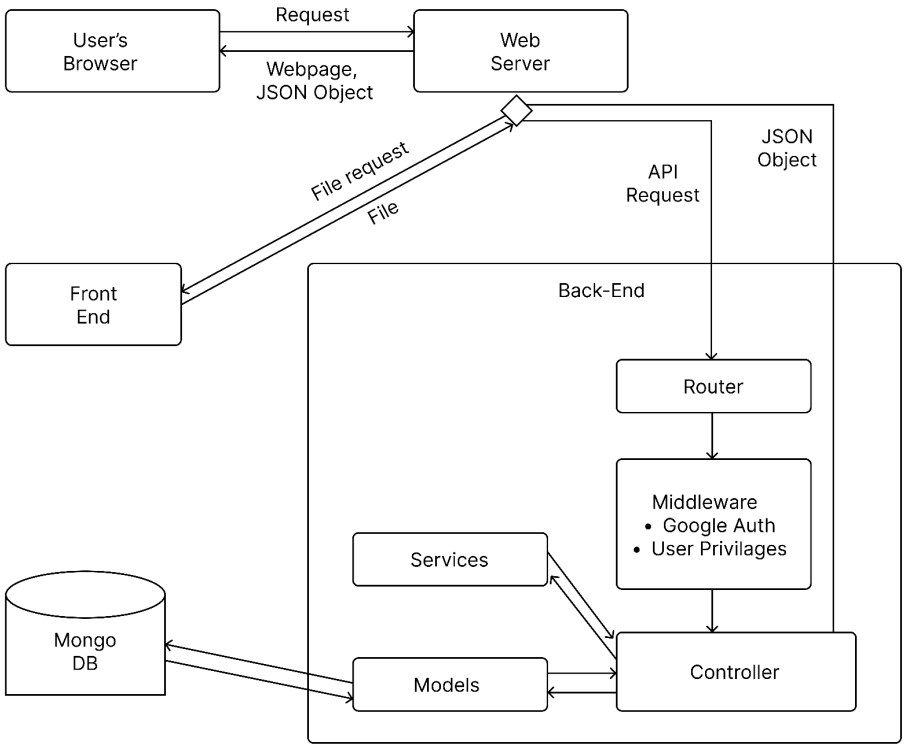

Figure: Architecture for the application

Component Description
Front-end: It essentially contains a set of html files with javascript code embedded in them. These are the screens the user sees, each with various components, some of which may be updated due to an API call made by the javascript code in the page.

Back-end: Back end has many components.

- **Router**: The router is responsible for mapping incoming API requests to the appropriate controller actions and handles the routing logic, ensuring that requests are directed to the correct parts of the application.

- **Middleware**: A request passes through the middleware, where the user is authenticated (using google auth) and the user's privileges checked. Suitable flags are set.

- **Controller**: Contains the code for all the APIs. On an API request, code for that API is executed (by the application server). Controller modules interact with Models for

database access, and use some services. Upon completion of an API request, it sends a JSON object back to the browser (which will update the webpage with the information.)

- **Models**: Contains the functions to create the databases and establishes the connection. It passes the database tables to the controller as objects on which the controller can make method calls for getting the data or executing the operations.

- **Services**: These are various services that the controller uses. Email and notifications service utilized to send the updates to users (Optional).

2. Technology Stack and Other Choices

The following technologies have been chosen. As required by the SRS, the main criteria for selection was that the technology should be free and open source. In addition, the choices should be able to support the functional as well as non-functional requirements of the application. Another factor in selection was familiarity with the technology of the team.

Frontend

- HTML(Hyper Text Markup Language) and CSS(Cascading Style Sheets). Using HTML and CSS as core technologies ensures compatibility across various web browsers and devices. They provide full control over the look and feel of the application's user interface.
- React.js. It is free and open source and its component-based structure and wide range of libraries will facilitate building the application quickly with minimal code development, which will also promote maintainability.
- Redux. Redux is an open source and rich library that provides for managing application state, facilitating predictable data flow and updates. This will be particularly useful for handling the state of event requests, user registrations, and notifications.

Backend

- TypeScript. TypeScript will be used as the backend programming language. It is open source and enhances code maintainability and catches type-related errors during development, reducing runtime issues.
- Node.js. Node.js is one of the most popular open source JavaScript backend for developing web-based applications. It provides rich libraries, eases connections with database, has event-driven, non-blocking I/O architecture which allows good response time and allows high concurrency.
- Express.js. It is an open source web application framework which provides functionality for routing which simplifies the process of defining API routes and handling HTTP requests using MVC (Model View Controller) architecture. It also provides support for development.

Database

- MongoDB. Its NoSQL, document-oriented nature is well-suited for handling diverse data types, such as event details, user profiles, and club information. Its flexibility allows for

efficient data retrieval and storage, which is essential for managing event requests and club information. It is open source and allows limited use for free.

Security

- It was decided to use the google auth service for user authentication. This is a proven and reliable service and is available for free.

Hosting

- The application will be hosted on a virtual server to be provided by the organization. A suitable configuration will be sought to ensure fast execution of the application. The option of cloud was not taken, as the organization has sufficient servers and expertise to manage them.

3. Evaluation

- The architecture supports modularity and allows for new features to be added to existing screens, by suitably adding new APIs and changing the front-end code to call the new APIs. Completely new screens in the front-end can also be easily added. Hence, the architecture supports ease of modifiability.
- Security and access control is ensured by google auth. It also ensures the design constraint that institute email be used for login and authentication. (Some other security requirements like input validation will be supported during implementation.)
- Performance challenges have been minimized by selecting the modified MVC style and the chosen technology stack, which ensures that each request does not need to result in the full page being loaded.
- To ensure performance constraints are met, a suitable virtual host is to be allocated to host the application.
- As standard technologies (e.g. html, css, javascript, etc.) that are supported in all major browsers are used, compatibility requirements will be met.

—-

Prepared by: Bhagesh Gaur, Madhava Krishna, Ramanjeet Singh, Pankaj Jalote