Ngoc Le

WEB APPLICATION FOR SEARCHING LOCAL EVENTS

Technology
2023

ABSTRACT

Author	Ngoc Le
Title	Web Application for Searching Local Events
Year	2023
Language	English
Pages	57
Name of Supervisor	Rayko Toshev

The purpose of this thesis was to develop a website to help users to find local events. The application has a user-friendly interface, which provides users with relevant information about the events, including their location, date, and time.

The technologies used in the web application were HTML, CSS, and JavaScript (React) for the front end, the backend was implemented with Node.js and Express.js, and MongoDB for the database. The project describes in detail the design and development process of web applications, including the user interface, data schema, and backend functionality.

The project results in the web application provides users with a simple and efficient interface to search for local events based on their interests and preferences. In addition, the users can create their events and connect with other users who have similar interests.

Keywords Web applications, HTML, CSS, JavaScript, and MongoDB.

CONTENTS

ABSTRACT 2
LIST OF FIGURES AND TABLES 5
ABBREVIATIONS 7
1 INTRODUCTION 8
1.1 Background 8
1.2 Motivation 9
1.3 Objective 10
1.4 Advantages of the application 10
1.5 Challenges 11
2 LITERATURE REVIEW 12
3 METHODOLOGY 13
3.1 Technologies used 13
3.1.1 MERN 13
3.1.2 HTML & CSS 13
3.1.3 JavaScript 14
3.1.4 ReactJS 14
3.1.5 Axios 15
3.1.6 Redux Toolkit 15
3.1.7 Node.js 16
3.1.8 Express.js 17
3.1.9 MongoDB 18
3.1.10 Mongoose 18
3.1.11 Swagger 19
3.2 Environment Set Up 20
3.2.1 IDE 20
3.2.2 Version Control 20
3.2.3 Dependencies 21
3.3 Requirements 26
3.3.1 Overview of Application 26

3.3.2 Use Case Diagram 26
3.3.3 Project File Structure 28
3.4 Back End 30
3.4.1 Mongoose Models 30
3.4.2 Authentication 34
3.4.3 Routing 38
4 RESULTS 41
4.1.1 Header and Navigation Bar 41
4.1.2 Register and Login Page 42
4.1.3 Home Page 43
4.1.4 Creating a New Event 45
4.1.5 Event Management 47
4.1.6 User Management 47
4.1.7 Admin page 48
5 TESTING 51
6 DISCUSSION 52
6.1 Results 52
6.2 Further improvements 52
7 CONCLUSIONS 54
REFERENCES 55

LIST OF FIGURES AND TABLES

Figure 1. Illustrative diagram of the MERN stack. 13

Figure 2. How node.js process incoming requests using the event loop 17

Figure 3. Object Mapping between Node.js and MongoDB is managed via Mongoose. 19

Figure 4. Git workflow 21

Figure 5. Use case diagram of the user. 27

Figure 6. Use case diagram of the admin. 27

Figure 7. Project structure. 28

Figure 8. Back-end structure 29

Figure 9. Front-end structure. 29

Figure 10. User model. 31

Figure 11. Function in user model. 32

Figure 12. Event model. 33

Figure 13. Token model. 34

Figure 14. Authentication middleware. 35

Figure 15. Generate authentication token function. 37

Figure 16. Generate token function. 37

Figure 17. Save token function. 38

Figure 18. Verify token function. 38

Figure 19. Route folder. 40

Figure 20. Header and Navigation Bar. 41

Figure 21. Header and navigation bar after login. 41

Figure 22. Components file. 41

Figure 23. Login page. 42

Figure 24. Register page. 42

Figure 25. Home page 43

Figure 26. Home page after login. 43

Figure 27. Pop-up page. 44

Figure 28. Home page folder. 44

Figure 29. New event creation. 45
Figure 30. Choose date and time. 45
Figure 31. Filling in all the information. 46
Figure 32. Adding a photo. 46
Figure 33. Notification after clicking Submit. 46
Figure 34. Event management. 47
Figure 35. User management. 47
Figure 36. My profile page. 48
Figure 37. Admin home page. 48
Figure 38. Manage Events page. 49
Figure 39. Manage Users page. 49
Figure 40. Admin ban user. 49
Figure 41. Banned account 50

Table 1. List of features of the application. 26
Table 2. List of authentication routes 39
Table 3. List of user routes 39
Table 4. List of event routes. 39

ABBREVIATIONS

API	Application Programming Interface
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
UI	User Interface
DOM	Document Object Model
I/O	Input/Output
IDE	Integrated Development Environment
UML	Unified Modeling Language
HTTP	The Hypertext Transfer Protocol
NPM	Node Package Manager
JWT	JSON Web Token
URL	Uniform Resources Locators

1 INTRODUCTION

The thesis focuses on the development of a web application for searching local events using the MERN (MongoDB, Express.js, React, Node.js) stack. The MERN stack is a popular technology stack for building robust and scalable web applications. This thesis explores the utilization of this stack to create a comprehensive platform that enables users to discover and explore various local events within their communities.

1.1 Background

The web application for searching local events is developed to address the growing need for a centralized platform that connects individuals with the wide range of events and activities happening in their local area. Staying informed of the numerous activities taking place in one's immediate surroundings might be difficult in today's world of rapid advances in technology and digitalization. Event information is often scattered across various sources, making it difficult to discover and keep track of upcoming events. This web application aims to bridge this gap by providing a single, user-friendly platform where individuals can easily explore, discover, and access local events of their interest.

By collecting event information from various sources and presenting it in a centralized manner, the application simplifies the process of event discovery and enhances the overall user experience. The background of the web application is rooted in the desire to promote community engagement, foster connections between individuals, support local businesses and organizations, and provide a convenient solution for users to explore and participate in the rich tapestry of local events happening in their area.

1.2 Motivation

The development of a web application that helps users find local events is driven by various factors. The primary objective is to promote event exploration by providing a user-friendly platform to discover a wide range of local events. By encouraging users to step out of their comfort zones and explore new experiences, the application fosters curiosity, adventure, and personal growth.

Moreover, the application aims to facilitate social connections by recognizing events as social catalysts that bring together like-minded individuals. By allowing users to discover and attend local events, the application creates opportunities for people to connect, network, and build meaningful relationships. This ultimately enhances social interaction and community engagement.

In addition to promoting social connections, the web application supports local businesses by featuring and promoting their products and services through local events. By driving attendance and increasing visibility, the application contributes to the growth and success of these businesses, fostering vibrant local economies and encouraging community support.

Lastly, the web application seeks to enrich the cultural fabric of a community by curating a diverse range of cultural events. By promoting cultural enrichment among its users, the application allows them to participate in festivals, exhibitions, workshops, and performances that celebrate local heritage, art, music, and cuisine. This cultivates a sense of pride, appreciation, and understanding for the cultural identity of the community, contributing to its promotion and preservation.

In conclusion, the development of a web application for finding local events is motivated by event exploration, social connections, support for local businesses, and cultural enrichment. These factors aim to enhance the quality of life and community dynamics.

1.3 Objective

Through this thesis, the aim is to demonstrate the effectiveness and practicality of the MERN stack in building a web application for searching local events while highlighting its potential to enhance user experiences and promote community involvement.

Users can access approved events in the event list, which includes the date, time, description, photos, and organizer information of the events. However, if users want to create their events, they must first register and log in to the system before they can handle personal events, such as creating and deleting posted events. The admin system manages users and events, including the approval of newly created events. Additionally, the admin may restrict users and delete posted events.

1.4 Advantages of the application

Enhanced Event Discovery: A web application for searching local events provides users with a centralized platform to discover a wide range of events happening in their local area. It improves event visibility and accessibility, allowing users to explore and find events that align with their interests.

Improved User Experience: By utilizing modern web technologies and intuitive design, a web application can offer a user-friendly interface and seamless navigation. Users can easily search, filter, and browse through events, making the process of event discovery and selection more efficient and enjoyable.

Increased Community Engagement: A web application that promotes local events encourages community involvement and participation. It creates a sense of belonging and encourages individuals to actively engage with their local community, fostering a vibrant and connected social fabric.

1.5 Challenges

Data Accuracy and Availability: Ensuring the accuracy and availability of event data can be a challenge. Events may change or get canceled, and keeping the application up-to-date with the latest information requires continuous data management and verification.

User Adoption and Engagement: The success of a web application for searching local events relies on user adoption and engagement. Encouraging users to use the application and regularly check for events may require effective marketing strategies and continuous efforts to provide compelling content and features.

Competing Platforms and Information Fragmentation: There might be other platforms or websites offering similar services, which can lead to information fragmentation. Coordinating with event organizers and aggregating data from multiple sources can be complex, as not all events may be listed in a centralized database.

2 LITERATURE REVIEW

Web development has a rich history that began in the early 1990s with the introduction of the World Wide Web. Initially, websites were simple and text-based, but advancements in technologies such as HTML, CSS, and JavaScript brought about more visually appealing and interactive experiences. The emergence of server-side programming languages enabled the development of dynamic web applications. The Web 2.0 era emphasized user-generated content and collaboration, leading to the rise of interactive websites and social media platforms. The advent of mobile devices prompted the need for responsive design, and frameworks such as React and Angular streamlined development processes. Looking forward, web development continues to evolve with a focus on performance optimization, security, and integration of emerging technologies.

3 METHODOLOGY

3.1 Technologies used

3.1.1 MERN

MERN is a shortened form representing MongoDB, Express.js, React.js, and Node.js. This acronym refers to a JavaScript and JSON-oriented three-tier architecture for web development. React.js, as the top-tier framework, handles the front-end of the application. In the middle layer, Express.js and Node.js work together, with Express.js serving as a server-side framework running within the Node.js server environment. Finally, MongoDB serves as the database tier, responsible for storing and managing data within the application. (What Is The MERN Stack? Introduction & Examples, n.d.)

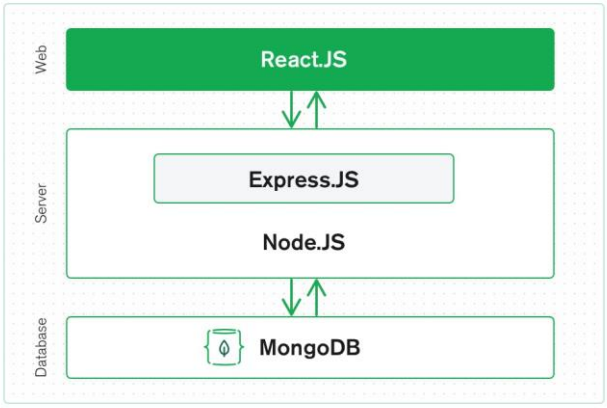


Figure 1. Illustrative diagram of the MERN stack.

3.1.2 HTML & CSS

HTML, known as HyperText Markup Language, is the fundamental cornerstone of the World Wide Web. It plays a crucial role in structuring and signifying web content. While CSS primarily deals with the visual aspects of a web page and JavaScript handles its functionality and behavior, HTML remains the core element responsible for establishing the organization and meaning of content on the web. (HTML: HyperText Markup Language | MDN, 2023)

CSS, short for Cascading Style Sheets, serves as a tool for designing and structuring web pages. It provides the flexibility to customize the appearance of content by modifying attributes such as fonts, colors, sizes, and spacing. Additionally, CSS enables the creation of multi-column layouts and the incorporation of decorative features, animations, and other visual elements, enhancing the overall visual presentation of web pages. (Learn to style HTML using CSS - Learn web development | MDN, 2023)

3.1.3 JavaScript

JavaScript is a versatile scripting and programming language used to incorporate advanced functionality into web pages. Whenever a web page goes beyond simply displaying static information and includes features like real-time content updates, interactive maps, animated graphics, or video players, JavaScript is typically involved. It serves as the third layer in the hierarchy of standard web technologies, with HTML and CSS being the other two layers. (What is JavaScript? - Learn web development | MDN, 2023)

3.1.4 ReactJS

React.js is a JavaScript-based front-end development library that is open-source. It focuses on creating UI components, making it easier for developers to build interactive and intricate user interfaces. With its component-based architecture and expressive nature, React.js simplifies the process of constructing user interfaces with enhanced interactivity and complexity. (Panchal, n.d.)

In React, there exists a "virtual DOM object" for each DOM object. This virtual DOM object acts as a lightweight copy or representation of the corresponding real DOM object. While a virtual DOM object possesses similar properties as a real DOM object, it lacks the direct ability to modify the on-screen content. Manipulating the actual DOM can be a slow process. However, manipulating the virtual DOM is considerably faster since no visual rendering takes place.

Conceptually, editing the virtual DOM can be compared to modifying a blueprint, whereas modifying the real DOM is akin to rearranging rooms in an actual house. (React: The Virtual DOM, n.d.)

Components are self-contained and reusable code snippets that have a similar purpose to JavaScript functions. They operate independently and generate HTML output. Components can be classified into two types: Class components and Function components. Both types enable developers to create modular and reusable code blocks within a web application. (React Components, n.d.)

3.1.5 Axios

Axios is a versatile HTTP client that offers seamless integration with both Node.js and web browsers. It is designed with a focus on handling asynchronous operations using promises. A notable advantage of Axios is its isomorphic nature, enabling it to function effectively in both browser and Node.js environments using a single codebase. When used in the server-side context, Axios utilizes the native Node.js http module, while on the client-side (browser), it relies on XMLHttpRequests. This adaptability and compatibility make Axios a popular choice among developers for handling HTTP requests in their applications. (Getting Started | Axios Docs, n.d.)

Axios is employed for communicating with the backend, utilizing the Promise API introduced in JavaScript ES6. By using Axios, we can send requests to an API, retrieve the resulting data, and seamlessly incorporate it into our ReactJS application. Axios is a highly popular HTTP client that enables direct communication with APIs from the browser, streamlining the process of exchanging data efficiently. (How to Use Axios with React: The Ultimate Guide [2023], 2023)

3.1.6 Redux Toolkit

Redux Toolkit is an official package that provides a clear and opinionated approach to building Redux applications that are scalable and efficient. It includes

a range of helpful methods that make common Redux tasks easier, such as setting up the store, defining reducers, handling immutable updates, and creating complete "slices" of state without the need for writing action creators or types manually. (Redux Toolkit: Overview | Redux, 2023)

Moreover, Redux Toolkit also incorporates the RTK Query data retrieval API. This API is specifically designed for Redux and offers powerful features for fetching and caching data. It simplifies the process of handling common scenarios for data loading in web applications, eliminating the need to manually write logic for fetching and caching data. (Redux Toolkit: Overview | Redux, 2023)

3.1.7 Node.js

Node.js is a runtime environment that excels in developing high-performance and scalable server-side and networking applications. It is open-source and compatible with multiple platforms. Operating on a single thread, Node.js leverages the V8 JavaScript runtime engine. Its distinctive feature is its event-driven, non-blocking I/O architecture, which optimizes efficiency and makes it an excellent choice for real-time applications. (What Is Node.js and Why You Should Use It, n.d.) (What Is Express.js? Everything You Should Know, n.d.)

A significant benefit of using Node.js is the absence of locks, eliminating concerns about deadlocks for users. Node.js functions usually do not directly perform I/O operations, preventing the process from becoming blocked, except when synchronous methods from the Node.js standard library are used for I/O tasks. By avoiding blocking operations, Node.js enables the development of scalable systems, making it a highly suitable choice for building such applications. (About | Node.js, n.d.)

Node.js utilizes the "Single Threaded Event Loop" architecture to handle multiple clients simultaneously. In a multi-threaded request-response model, numerous clients send their requests, and the server processes each one before sending

back the response. However, to handle concurrent requests, Node.js employs multiple threads. These threads are organized in a thread pool, and when a request is received, it is assigned to an available thread for processing. (What Is Node.js and Why You Should Use It, n.d.)

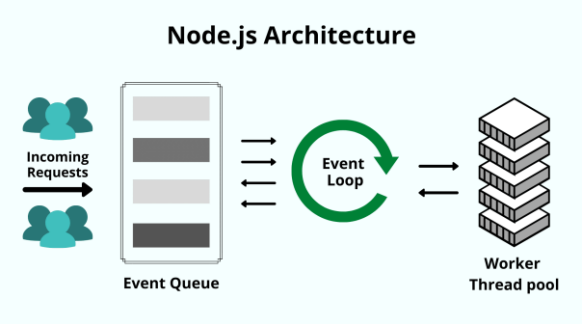


Figure 2. How node.js process incoming requests using the event loop

3.1.8 Express.js

Express.js is an open-source web application framework designed for Node.js. It provides developers with a fast and convenient way to design and construct web applications. Web applications are software programs that can be accessed and run on web browsers. With Express.js, developers can streamline the process of building web applications, allowing for quick and easy development. (What is Express.js? | Why should use Express.js? | Features of Express.js, n.d.)

Express.js is a versatile framework that enables the development of various types of web applications, including single-page, multi-page, and hybrid applications. It has also become the industry standard for constructing backend applications using Node.js. (What Is Express.js? Everything You Should Know, n.d.)

The framework includes a set of tools for building and deploying large-scale, enterprise-ready applications, including web applications, HTTP requests and responses, routing, and middleware. Additionally, it offers a command line interface (CLI) tool called Node Package Manager (NPM) that allows developers to source for created packages. (What Is Express.js? Everything You Should Know, n.d.)

3.1.9 MongoDB

MongoDB, established in 2007, is a flexible and open-source document database designed for storing data with a flexible schema and it operates on a horizontal scale-out architecture. MongoDB stores data as documents using BSON, a binary representation, and applications can easily retrieve this data in JSON format. Scalability is a key focus of MongoDB, enabling collaboration among multiple small machines to create high-performance systems capable of efficiently processing large amounts of data. Unlike traditional SQL databases, MongoDB's document-based approach offers a more adaptable and dynamic solution for data storage. (Why Use MongoDB And When To Use It?, n.d.)

MongoDB is a document database that is highly flexible and open-source. It utilizes a horizontal scale-out architecture, storing data as documents in BSON format while supporting JSON for retrieval. The scalability of MongoDB is its major advantage, as it enables efficient processing of large datasets through the collaboration of small machines. (Patil, 2023)

3.1.10 Mongoose

Mongoose is a library for MongoDB and Node.js that provides an Object Data Modeling (ODM) approach. It allows developers to work with MongoDB databases in a more structured and intuitive manner. The translation between objects created in code and their representation in MongoDB is managed by Mongoose, in addition to schema validation and management of data relationships. (Karnik, 2018)

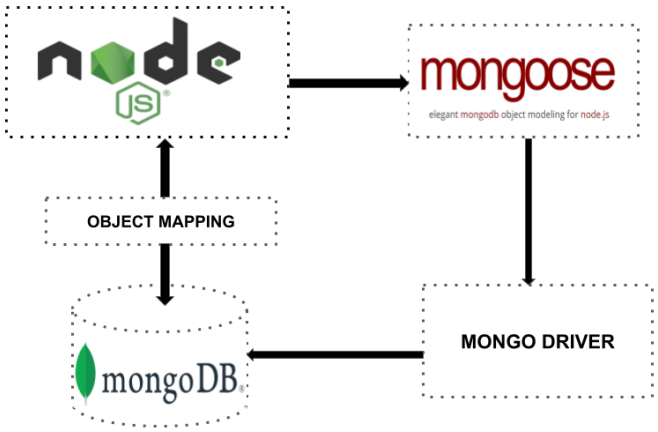


Figure 3. Object Mapping between Node.js and MongoDB is managed via Mongoose.

3.1.11 Swagger

Swagger is a free and open-source set of rules, specifications, and tools for creating and documenting RESTful APIs. Using the Swagger framework, programmers can generate API documentation that is interactive, machine and user readable. (What is Swagger? Definition from WhatIs.com, n.d.)

Indicators including supported operations, parameters, and outputs, as well as information on available endpoints, license requirements, and authorization standards, are frequently included in API specifications. Swagger has the capability to generate documentation automatically by extracting information from the source code. It achieves this by making requests to the API, which responds with a documentation file that is generated based on annotations present in the code. (What is Swagger? Definition from WhatIs.com, n.d.)

Building, describing, testing, and consuming RESTful web services are all made easier with Swagger. It works with both top-down and bottom-up API development methods. Several open-source tools for APIs are provided by Swagger, including Swagger Editor, Swagger Codegen, Swagger User Interface, and Swagger Inspector. (What is Swagger? Definition from WhatIs.com, n.d.)

3.2 Environment Set Up

3.2.1 IDE

An Integrated Development Environment (IDE) is a web application that provides developers with a comprehensive suite of tools and functionalities to streamline software code development.

For this particular project, Visual Studio Code (VS Code) was selected as the preferred code editor. VS Code seamlessly combines the convenience of a source code editor with robust developer tools, prioritizing a smooth and efficient editing experience. It offers a fast and reliable source code editor that is well-suited for everyday use. With extensive support for various programming languages, developers can benefit from features like syntax highlighting, bracket matching, auto-indentation, box selection, and code snippets, ensuring enhanced productivity. Additionally, VS Code allows for intuitive customization of keyboard shortcuts, preferences, and even leverages community-contributed shortcut mappings, further optimizing code navigation for developers. (Why Visual Studio Code?, n.d.)

3.2.2 Version Control

Version control, also referred to as source control, is the process of monitoring and managing alterations made to software code. It involves using specialized software that maintains a record of each code modification in a dedicated database. This enables developers to effectively revert back to previous versions of the code, facilitating error correction and minimizing disruption to the entire team when mistakes occur. (What is version control, n.d.)

In this project, Git was selected as the chosen version control system. Git is a widely used, free, and open-source source code management tool that facilitates effective management of projects, ranging from small to extremely large in scale. It enables multiple developers to collaborate seamlessly on non-linear

development by keeping track of changes made to the source code. (Perveez, 2023) GitHub is an online platform that hosts Git repositories and offers additional collaborative tools. By integrating Git's version control capabilities with its own features, GitHub facilitates smooth and efficient management of code versions and promotes effective teamwork and collaboration among developers.

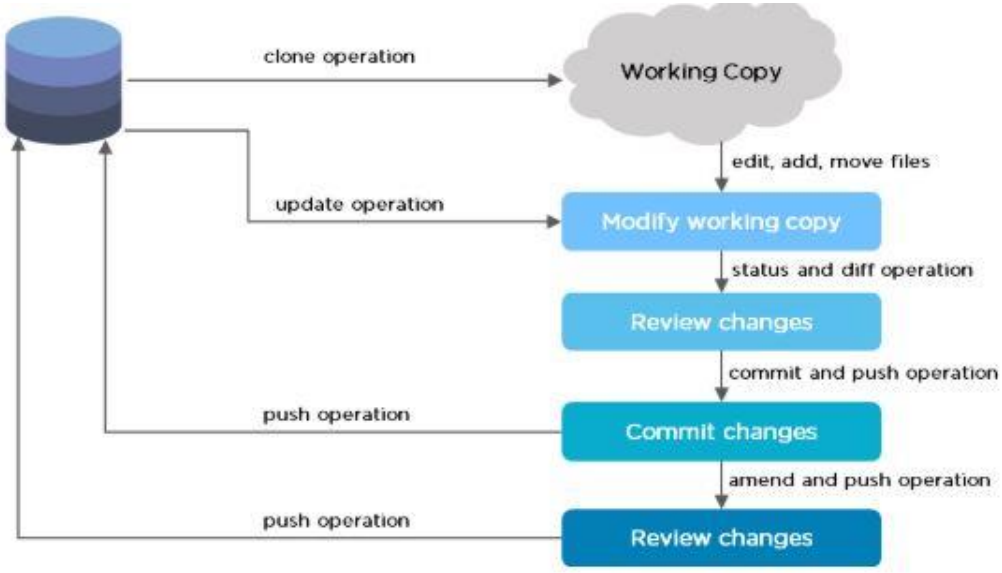


Figure 4. Git workflow

3.2.3 Dependencies

To execute the application efficiently, the back-end libraries listed below must be installed.

- "bcryptjs" is a module that allows developers to store passwords as hashed passwords, instead of the original one. “bcryptjs” is installed by using following command ‘npm install bcryptjs’.
- "cors" stands for Cross-Origin Resource Sharing, is a mechanism that enables web browsers to make cross-origin HTTP requests in a secure way by loosening the same-origin restrictions. “cors” is installed by using following command ‘npm install cors’.

- "dotenv" is a package that allows to load environment variables from .env file into process.env. “dotenv” is installed by using following command ‘npm install dotenv’.
- "express" simplifies the creation of routing and controllers for APIs. “express” is installed by using following command ‘npm install express’.
- "express-validation" is an Express middleware that provides request validation capabilities and sends back a response with errors if any of the specified validation rules are unsuccessful. “express-validation” is installed by using following command ‘npm install express-validation’.
- "fs-extra" is a Node.js module that adds functions that are not included in ‘fs’ module for file system operations and provides promise support to ‘fs’ methods. “fs-extra” is installed by using following command ‘npm install fs-extra’.
- "http-status" is a Node.js module that includes an extensive list of constants that illustrate standard HTTP status code, making it easier to create and follow the status of HTTP responses in web applications. “http-status” is installed by using following command ‘npm install http-status’.
- "joi" is a JavaScript validation library that allows developers to define and apply validation rules to objects to ensure data integrity and compliance to specific criteria. “joi” is installed by using following command ‘npm install joi’.
- "jsonwebtoken" enables the generation and verification of JSON Web Tokens (JWTs), which are used in web applications for secure user authentication and data transfer between parties. “jsonwebtoken” is installed by using following command ‘npm install jsonwebtoken’.
- "moment" provides a simple and efficient way to work with dates, times, and durations. “moment” is installed by using the following command ‘npm install moment’.

- "mongoose" is an Object Data Modeling (ODM) library for Node.js and MongoDB. “mongoose” is installed by using following command ‘npm install mongoose’.
- "multer" is a Node.js middleware for handling multipart/form-data that is commonly used for file uploading. “multer” is installed by using following command ‘npm install multer’.
- "nodemailer" is a Node.js module that enables developers to send emails from the application. “nodemailer” is installed by using following command ‘npm install nodemailer’.
- "passport" is a Node.js middleware that simplifies web application authentication by allowing for a flexible and modular approach to implementing various authentication schemes. “passport” is installed by using following command ‘npm install passport’.
- "passport-jwt" is a Passport.js approach that allows JSON Web Token (JWT) authentication in Node.js apps, allowing secure and stateless authentication with JWT tokens. “passport-jwt” is installed by using following command ‘npm install passport-jwt’.
- "swagger-jsdoc" is a package that generates Swagger documentation for Node.js APIs by creating JSDoc comments in your code. “swagger-jsdoc” is installed by using following command ‘npm install swagger-jsdoc’.
- "swagger-ui-express" is a module based on a ‘swagger.json’ file, enables developers to serve auto-generated Swagger UI interface for their API from express. “swagger-ui-express” is installed by using following command ‘npm install swagger-ui-express'.
- "validator" is a Node.js npm package that provides a variety of validation functions for different types of data, including strings, numbers, emails, URLs, and more. “validator” is installed by using following command ‘npm install validator’.

In addition to the essential back-end libraries, the required front-end libraries must be installed. The list of the libraries is given below.

- "@ant-design/icons" is a package that contains a collection of customizable icons for the Ant Design UI library, allowing developers to easily add high-quality icons to their React applications. The command to install this dependency is 'npm install @ant-design/icons'.
- "@reduxjs/toolkit" is a package that contains utilities and conventions for simplifying and streamlining Redux development. The command to install this dependency is ‘npm install @reduxjs/toolkit’.
- "@testing-library/jest-dom" is a library includes a set of custom jest matchers for extending jest. The command to install this dependency is ‘npm install testing-library/jest-dom’.
- "@testing-library/react" is a package that provides a set of utilities and methods for testing React components. The command to install this dependency is ‘npm install @testing-library/react’.
- "@testing-library/user-event” is a package that provides simulation utilities for user events in JavaScript testing. The command to install this dependency is ‘npm install @testing-library/user-event’.
- "antd" is a well-known React UI package that provides a comprehensive set of reusable components as well as guidelines for creating modern and visually appealing web applications. The command to install this dependency is ‘npm install antd’.
- "axios" is installed by using following command ‘npm install axios’.
- "bootstrap" is a popular open-source front-end framework that provides a collection of CSS and JavaScript components to help developers design responsive and visually appealing via the web interfaces. The command to install this dependency is ‘npm install antd’.
- "moment" provides a simple and efficient way to work with dates, times, and durations. “moment” is installed by using the following command ‘npm install moment’.

- "react" is installed by using following command ‘npx create-react-app front-end’.
- "react-bootstrap" is a package that presents pre-built Bootstrap components as reusable React components, which makes it simple to integrate Bootstrap styling and functionality into React applications. The command to install this dependency is ‘npm install react-bootstrap’.
- "react-dom" is a package that provides React applications with rendering capabilities, allowing rendering of React components into the DOM and manage interactions with the browser environment. This package is already included as dependency when creating a React project.
- "react-redux" integrates the Redux state management framework with React, allowing programmers to control the state of an application predictably and effectively while also making it simple for components to connect to and update the state. The command to install this dependency is ‘npm install react-redux’.
- "react-router-dom” is a React application routing framework that allows for dynamic navigation and rendering of multiple components based on URL changes. The command to install this dependency is ‘npm install react-router-dom’.
- "react-scripts" is a set of scripts and configuration used by Create React App to build and run a React application. This package is already included as dependency when creating a React project.
- "react-toastify" is a library that provides a simple and customizable way to display toast notifications in React applications. The command to install this dependency is ‘npm install react-toastify’.
- "web-vitals" is a JavaScript package that aids in the measurement and reporting of important performance indicators of a web page, such as loading time, interaction, and visual stability, providing insights into the user experience, and assisting in performance improvement. The command to install this dependency is ‘npm install web-vitals’.

3.3 Requirements

3.3.1 Overview of Application

Table 1 below shows all the features of the web application.

Table 1. List of features of the application.

	User	Admin
Register	Yes	No
Login	Yes	Yes
View the list of events	Yes	Yes
Create events	Yes	No
View created events	Yes	Yes
Delete posted events	Yes	No
Approve created events	No	Yes
Ban user	No	Yes

3.3.2 Use Case Diagram

Use case diagrams in UML provide a means to represent and analyze the behavior and requirements of a system. They offer a high-level overview of a system's functions and boundaries, showcasing the interactions between the system and its actors. Use case diagrams primarily focus on describing the actions and relationships between actors and use cases, rather than delving into the internal workings of the system. (Use-case diagrams in UML modeling, n.d.)

Use-case diagrams show and clarify the context and requirements of a whole system or key components of a system. A complex system can be modelled with a single use case diagram, or numerous use-case diagrams can be created to illustrate the system's components. (Use-case diagrams in UML modeling, n.d.)

Figure 5 depicts the use case diagram for the users, showcasing the various functions they can perform. To utilize the online application's features, users are required to log in. Once logged in, they can create new events, browse existing published events, delete their posted events, and modify their profile details, including contact information. Prior to logging in, users need to register for an account. However, accessing approved events does not necessitate registration or login.

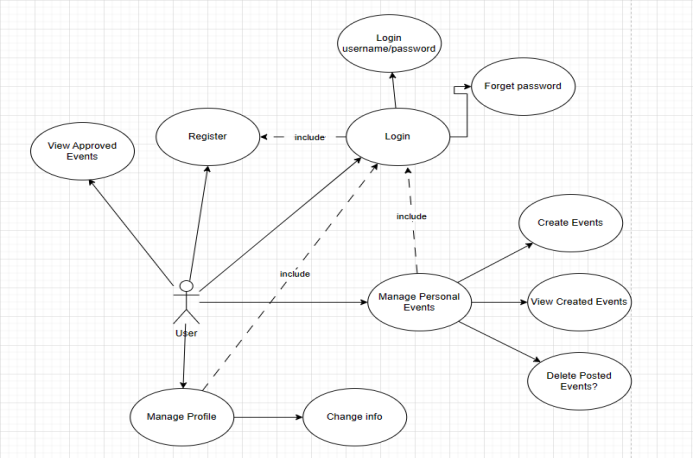


Figure 5. Use case diagram of the user.

Figure 6 shows the tasks that an administrator can perform. The administrator has the authority to view users and ban them if necessary. In terms of event management, the administrator is responsible for approving new events and monitoring existing events.

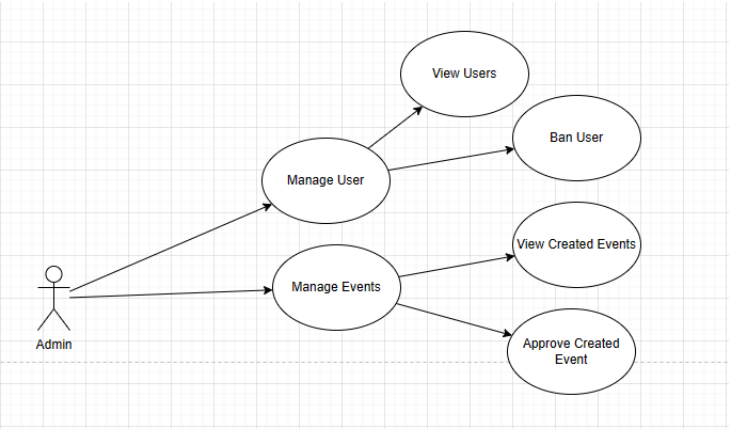


Figure 6. Use case diagram of the admin.

3.3.3 Project File Structure

Figure 7 below shows the structure of the application, the project is divided into two main folders, such as front-end and back-end.

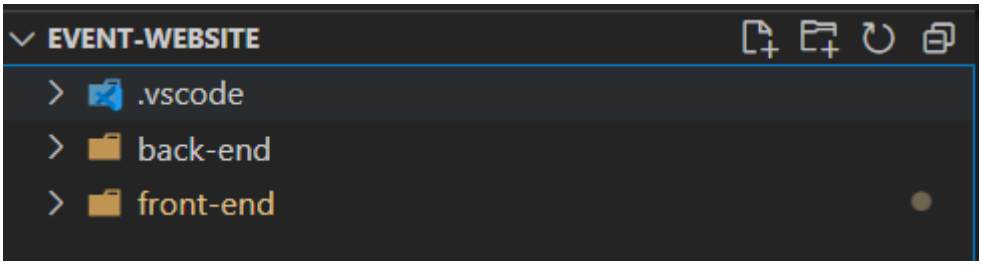


Figure 7. Project structure.

Figure 8 below illustrates the back-end structure. The '*src*' folder splits into two folders, including '*api*' and '*config*'. The '*api*' folder has a variety of small folders, such as:

- '*controllers*' includes all the controllers required for the application.
- '*middlewares*' contains all the middleware for the application.
- '*models*' folder stores the data models required for the application.
- '*routes*' consists of each file for each different route.
- '*services*' is where to store all the business logic.
- '*utils*' contains all the utilities and assistance required for the application.

Besides, the '*config*' folder stores all the configurations required to function with the database.

The structure of the front-end folder is depicted in Figure 9. The folder divides into several sections, which is: 'app', 'common', 'components', 'config', 'features', 'reducers', 'App.css', and 'App.js'

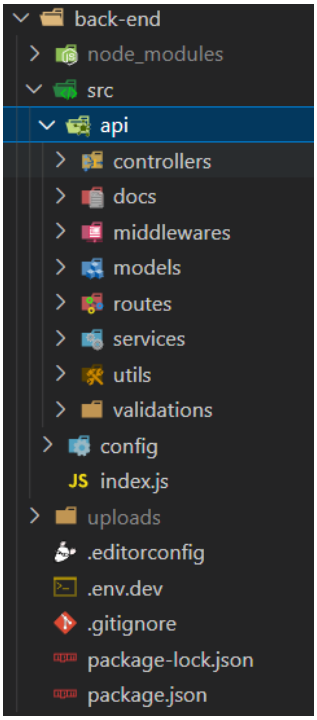


Figure 8. Back-end structure

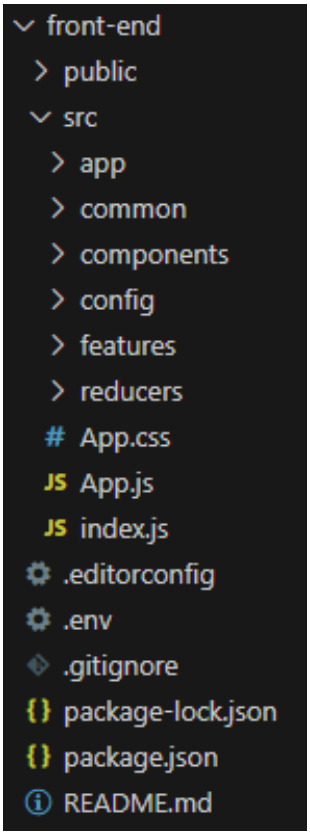


Figure 9. Front-end structure.

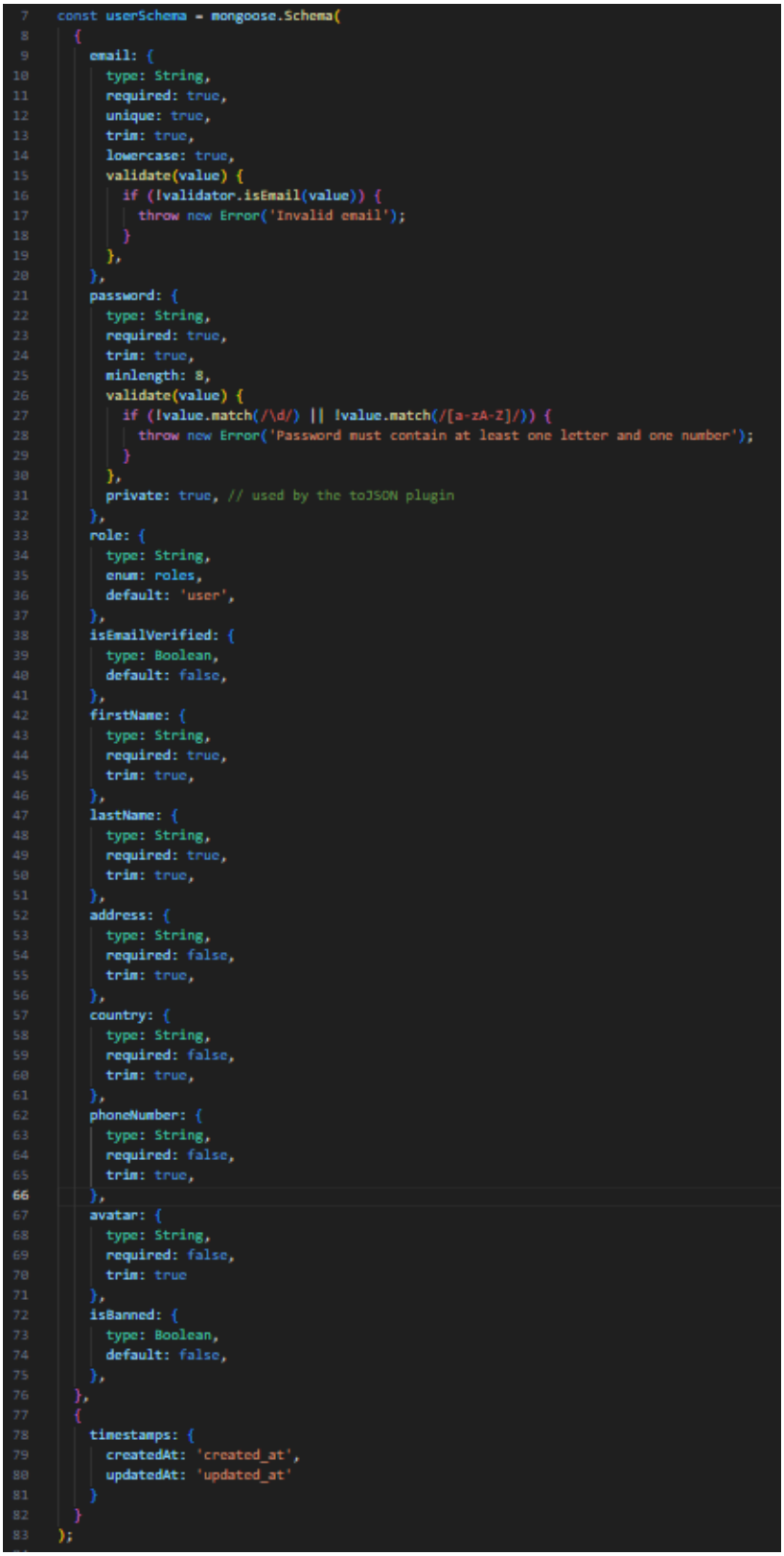
3.4 Back End

3.4.1 Mongoose Models

A Mongoose model is a graphical depiction of the structure of a MongoDB document as defined by a schema. A schema specifies the structure of the documents, including the fields, their types, and any validation procedures. It provides a way to enforce a consistent structure and data integrity for the documents stored in MongoDB. The application contains three models: user, event, and token.

Figure 10 depicts the elements of the user model. The ‘userSchema’ defines the structure and properties of a user object in the application. It includes fields such as email, password, role, isEmailVerified, first name, last name, address, country, phone number, avatar, and isBanned. It also includes timestamps for ‘createdAt’ and ‘updatedAt’ to track when the user was created and last updated. The 'isBanned' field, which has a Boolean value and is set by default to false, indicates whether or not the user has been banned. The schema specifies the data types, required fields, validation rules, and default values for the user object.

As shown in Figure 11, two plugins, ‘toJSON’ and ‘paginate’, are added to the schema. The ‘toJSON’ plugin converts the Mongoose document to a JSON object, and the ‘paginate’ plugin adds pagination capabilities to query results. ‘isEmailTaken’ checks if an email is already taken by another user, and ‘isPasswordMatch’ verifies if a provided password matches the user's stored password. A pre-save middleware is added to hash the password using bcrypt when it is modified. Finally, the schema is used to create the ‘User’ model, which is exported for use in other parts of the application.



```
const userSchema = mongoose.Schema(
  {
    email: {
      type: String,
      required: true,
      unique: true,
      trim: true,
      lowercase: true,
      validate(value) {
        if (!validator.isEmail(value)) {
          throw new Error('Invalid email');
        }
      },
    },
    password: {
      type: String,
      required: true,
      trim: true,
      minlength: 8,
      validate(value) {
        if (!value.match(/\d/) || !value.match(/[a-zA-Z]/)) {
          throw new Error('Password must contain at least one letter and one number');
        }
      },
      private: true, // used by the toJSON plugin
    },
    role: {
      type: String,
      enum: roles,
      default: 'user',
    },
    isEmailVerified: {
      type: Boolean,
      default: false,
    },
    firstName: {
      type: String,
      required: true,
      trim: true,
    },
    lastName: {
      type: String,
      required: true,
      trim: true,
    },
    address: {
      type: String,
      required: false,
      trim: true,
    },
    country: {
      type: String,
      required: false,
      trim: true,
    },
    phoneNumber: {
      type: String,
      required: false,
      trim: true,
    },
    avatar: {
      type: String,
      required: false,
      trim: true
    },
    isBanned: {
      type: Boolean,
      default: false,
    },
  },
  {
    timestamps: {
      createdAt: 'created_at',
      updatedAt: 'updated_at'
    }
  }
);
```

Figure 10. User model.

```
// add plugin that converts mongoose to json
userSchema.plugin(toJSON);
userSchema.plugin(paginate);

/**
 * Check if email is taken
 * @param {string} email - The user's email
 * @param {ObjectId} [excludeUserId] - The id of the user to be excluded
 * @returns {Promise<boolean>}
 */
userSchema.statics.isEmailTaken = async function (email, excludeUserId) {
  const user = await this.findOne({ email, _id: { $ne: excludeUserId } });
  return !!user;
};

/**
 * Check if password matches the user's password
 * @param {string} password
 * @returns {Promise<boolean>}
 */
userSchema.methods.isPasswordMatch = async function (password) {
  const user = this;
  return bcrypt.compare(password, user.password);
};

userSchema.pre('save', async function (next) {
  const user = this;
  if (user.isModified('password')) {
    user.password = await bcrypt.hash(user.password, 8);
  }
  next();
});

/**
 * @typedef User
 */
const User = mongoose.model('User', userSchema);

module.exports = User;
```

Figure 11. Function in user model.

Figure 12 below depicts the event schema structure in detail. The event model contains all of the fields, such as title, datetime, description, address, country, website, status, image, and author. The schema configuration includes timestamps for ‘createdAt’ and ‘updatedAt’, which automatically track the creation and modification dates of each event.

Figure 13 depicts the structure of a token schema, which defines how tokens associated with user activities are managed. It includes fields such as token, user, type, expires, blacklisted. The schema configuration includes timestamps for ‘createdAt’ and ‘updatedAt’, which automatically track the creation and modification dates of each token. The 'token' field is of the String type, and it stores the actual token value. The ‘user’ field represents the user associated with the token. It is defined as a reference to the User model. It ensures that the value stored in this field corresponds to a valid user in the User collection. The ‘type’ field indicates the type of token. The ‘expires’ field stores the expiration date and

time of token. And the ‘blacklisted’ field indicates whether or not the token has been blacklisted. It is of the Boolean type and has a default value of false.

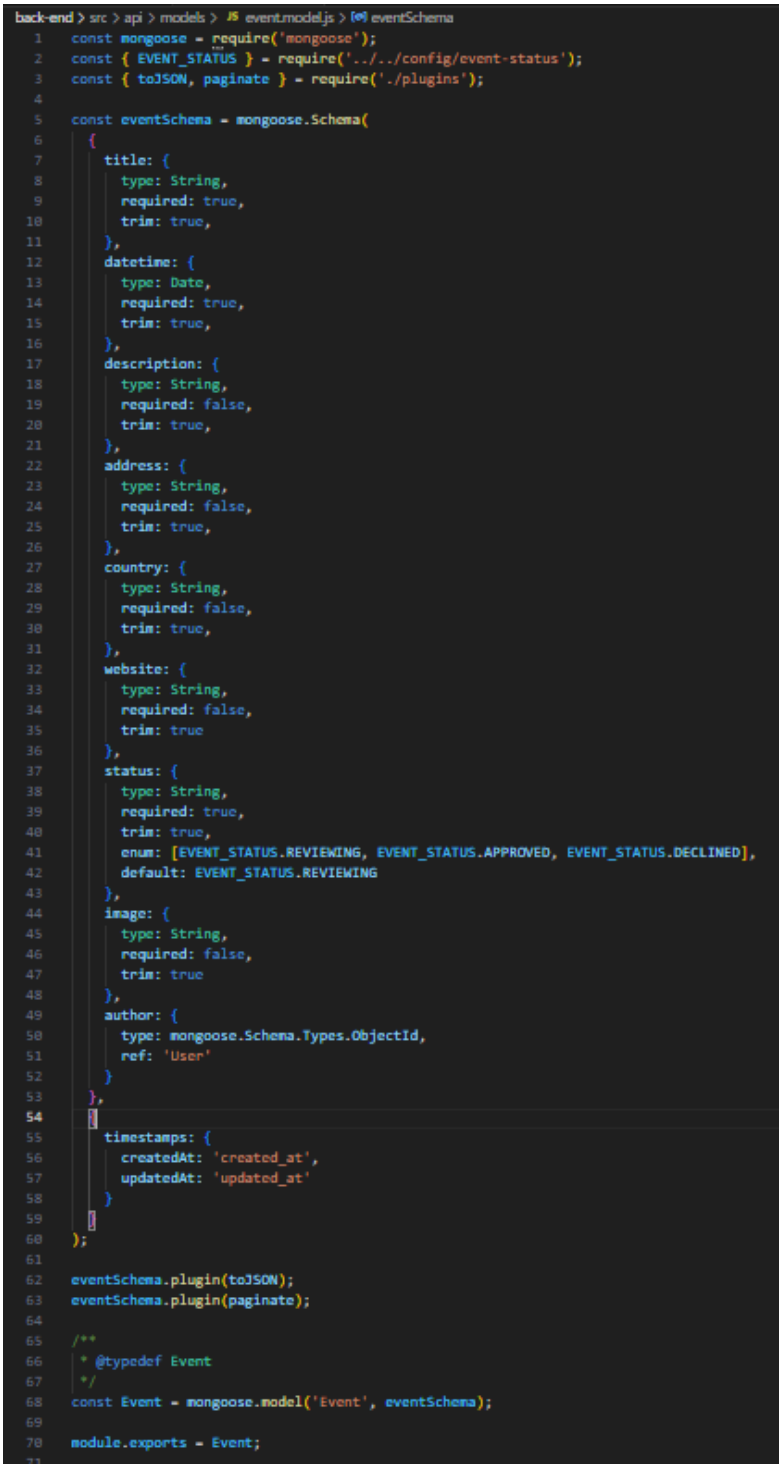


Figure 12. Event model.

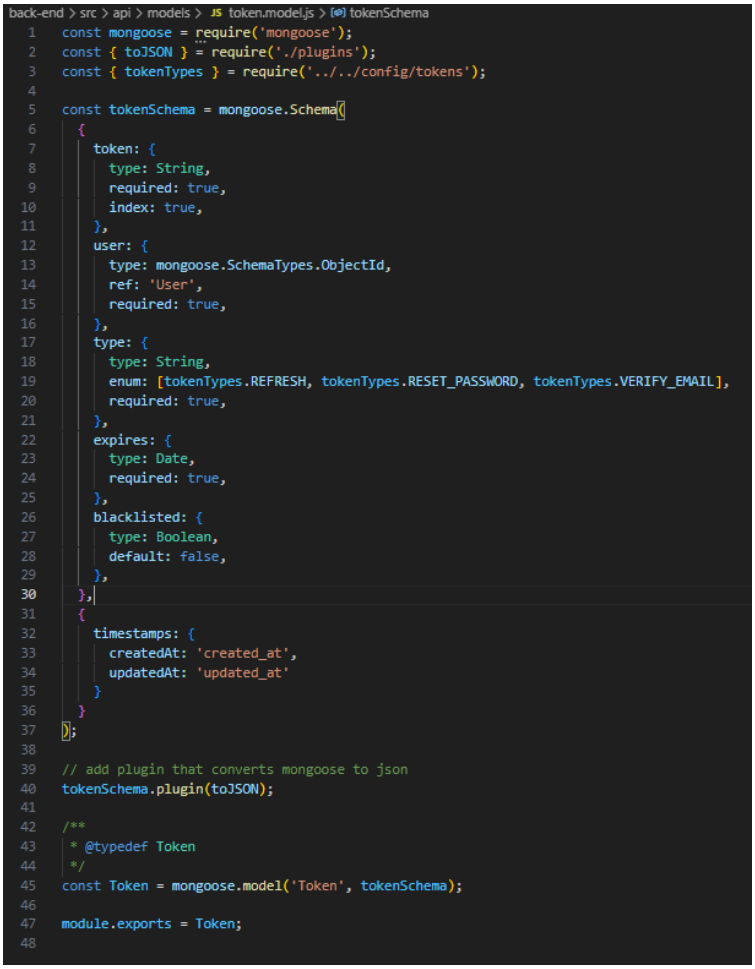


Figure 13. Token model.

3.4.2 Authentication

Authentication is the method used to confirm a user's identity. An incoming request is matched with a set of identifying credentials in this process. The submitted credentials are compared to those in a database on a local operating system or within an authentication server that has information about the authorized user. (What is Authentication? Definition of Authentication, Authentication , n.d.)

Authentication middleware is used to validate user identities and ensure they have the appropriate permissions to access specified assets or execute specific actions. When a user logs into the system, the middleware is activated.

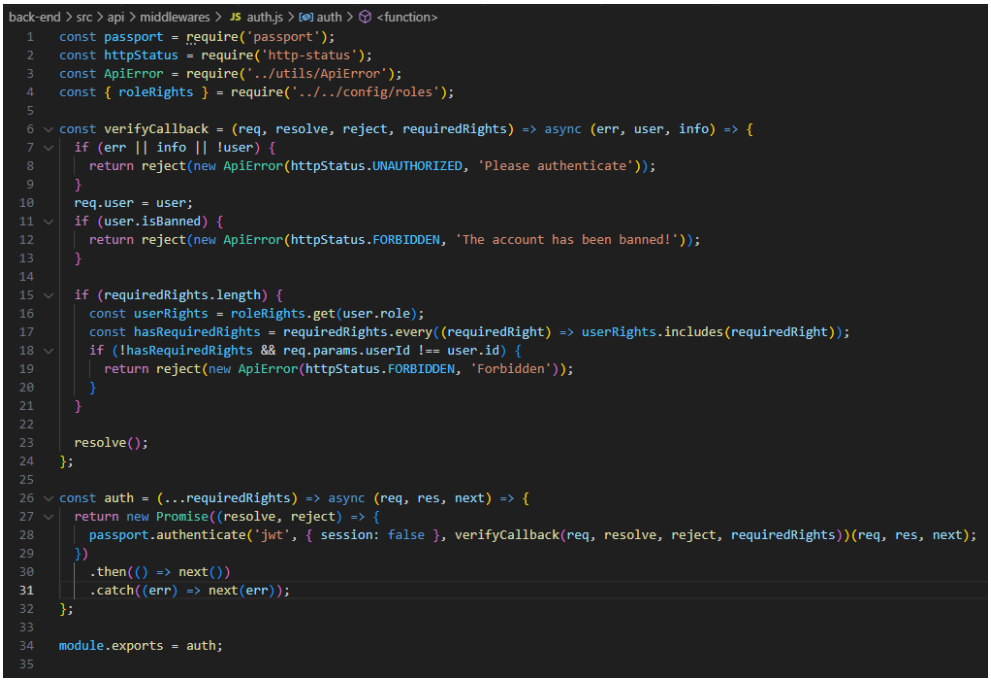


Figure 14. Authentication middleware.

On line 26 of Figure 14 above, the ‘auth’ function is a middleware generator that takes in an array of ‘requiredRights’ as parameters. It returns an async middleware function that will be used in route handlers. Within the ‘auth’ middleware, a new Promise is created to handle the authentication process. It is called ‘passport.authenticate’ with the strategy name *‘jwt’* and additional options ‘*{ session: false }*’. The ‘passport.authenticate’ function is passed the ‘verifyCallback’ function, along with the ‘req’, ‘resolve’, ‘reject’, and ‘requiredRights’ arguments. This sets up the verification callback for the authentication process.

The function ‘verifyCallback’ defined on line 6 serves as an asynchronous function that is invoked during the authentication process. It takes in the arguments ‘err’, ‘user’, and ‘info’ from Passport.js. Within the ‘verifyCallback’ function, various checks are performed to handle errors, missing user information, or authentication failures. If any of these conditions are met, a corresponding ‘ApiError’ instance is created, and the Promise is rejected. On successful authentication, the ‘user’ object is assigned to ‘req.user’. Additionally, the

function verifies if the user is banned and rejects the Promise if necessary. If ‘requiredRights’ are specified, it checks whether the user's role includes all the required rights. If not, and if the ‘userId’ request parameter does not match the authenticated user's ID, the Promise is rejected with a "Forbidden" error. If all checks pass, the Promise is resolved, allowing the control to be passed to the next middleware or route handler. Ultimately, the ‘auth’ middleware is exported to be utilized in the application's routes.

A JSON Web Token (JWT) is an established open standard (RFC 7519) that outlines a concise and autonomous approach to securely exchanging information as a JSON object between various entities. (JSON Web Token Introduction - jwt.io, n.d.) JWTs find widespread usage in web applications and APIs as a prevalent mechanism for handling authentication and authorization tasks.

The code provided below in Figure 15 is a module that handles token generation, verification, and saving in a web application. They are commonly used in web applications to handle user authentication, password reset, email verification, and token-based authentication mechanisms.

As shown in Figure 15, the function ‘generateAuthTokens’ generates both an access token and a refresh token for a user. It takes a ‘user’ object as a parameter and uses the ‘generateToken’ and ‘saveToken’ functions to generate and save the refresh token. It also generates an access token with a shorter expiration time. The function returns an object containing the access token and its expiration date, as well as the refresh token and its expiration date.

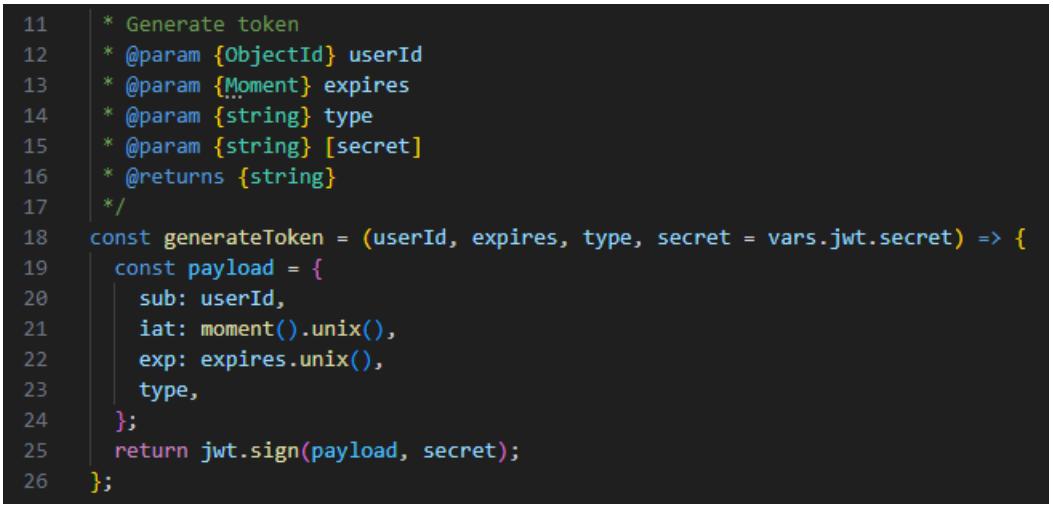
The function ‘generateToken’ as shown in Figure 16, which generates a JWT (JSON Web Token) with the provided ‘userId’, ‘expires’, ‘type’, and an optional ‘secret’. It uses the ‘jwt.sign’ method from the ‘jsonwebtoken’ library to create the token and returns it.

```
/**
 * Generate auth tokens
 * @param {User} user
 * @returns {Promise<Object>}
 */
const generateAuthTokens = async (user) => {
  const accessTokenExpires = moment().add(vars.jwt.accessExpirationMinutes, 'minutes');
  const accessToken = generateToken(user.id, accessTokenExpires, tokenTypes.ACCESS);

  const refreshTokenExpires = moment().add(vars.jwt.refreshExpirationDays, 'days');
  const refreshToken = generateToken(user.id, refreshTokenExpires, tokenTypes.REFRESH);
  await saveToken(refreshToken, user.id, refreshTokenExpires, tokenTypes.REFRESH);

  return {
    access: {
      token: accessToken,
      expires: accessTokenExpires.toDate(),
    },
    refresh: {
      token: refreshToken,
      expires: refreshTokenExpires.toDate(),
    },
  };
};
```

Figure 15. Generate authentication token function.



```
 * Generate token
 * @param {ObjectId} userId
 * @param {Moment} expires
 * @param {string} type
 * @param {string} [secret]
 * @returns {string}
 */
const generateToken = (userId, expires, type, secret = vars.jwt.secret) => {
  const payload = {
    sub: userId,
    iat: moment().unix(),
    exp: expires.unix(),
    type,
  };
  return jwt.sign(payload, secret);
};
```

Figure 16. Generate token function.

On line 37 in Figure 17, the function ‘saveToken’ saves the token in the database by creating a new ‘Token’ document. It takes the ‘token’, ‘userId’, ‘expires’, ‘type’, and an optional ‘blacklisted’ parameter. It uses the ‘Token.create’ method to create and store the token document, and then returns the created token document.

```
/**
 * Save a token
 * @param {string} token
 * @param {ObjectId} userId
 * @param {Moment} expires
 * @param {string} type
 * @param {boolean} [blacklisted]
 * @returns {Promise<Token>}
 */
const saveToken = async (token, userId, expires, type, blacklisted = false) => {
  const tokenDoc = await Token.create({
    token,
    user: userId,
    expires: expires.toDate(),
    type,
    blacklisted,
  });
  return tokenDoc;
};
```

Figure 17. Save token function.

Function ‘verifyToken’ verifies the authenticity and validity of a token. It takes the ‘token’ and ‘type’ as parameters. It uses the ‘jwt.verify’ method to decode and verify the token, and then queries the ‘Token’ collection to find a matching token document based on the provided parameters. If a matching token is found and is not blacklisted, the function returns the token document; otherwise, it throws an error.

```
/**
 * Verify token and return token doc (or throw an error if it is not valid)
 * @param {string} token
 * @param {string} type
 * @returns {Promise<Token>}
 */
const verifyToken = async (token, type) => {
  const payload = jwt.verify(token, vars.jwt.secret);
  const tokenDoc = await Token.findOne({ token, type, user: payload.sub, blacklisted: false });
  if (!tokenDoc) {
    throw new Error('Token not found');
  }
  return tokenDoc;
};
```

Figure 18. Verify token function.

3.4.3 Routing

Routing is the mechanism that routes requests indicated by a URL and HTTP method to the code responsible for handling them. (14. Routing - Web Development with Node and Express [Book], n.d.)

Table 2. List of authentication routes.

Route	Purpose
POST /auth/register	Register as user
POST /auth/login	Login
POST /auth/logout	Logout
PATCH /auth/update-profile	Update profile
GET /auth/profile	Get user's profile
POST /auth/refresh-tokens	Refresh authentication tokens
POST /auth/forgot-password	Forgot password
POST /auth/reset-password	Reset password
POST /auth/send-verification-email	Send verification email
POST / auth/verify-email	Verify email

Table 3. List of user routes

Route	Purpose
POST /users	Create a user
GET /users	Get all users
GET /users/:userId	Get a user by ID
PATCH /user/:userId	Update a user by ID
DELETE /user/:userId	Delete a user

Table 4. List of event routes.

Route	Purpose
GET /events	Get all events

POST /events	Create an event
GET /events/:eventId	Get an event by ID
DELETE /events/:eventId	Delete an event
PATCH /events/:eventId	Update an event by ID

Table 2 pertains to authentication, Table 3 relates to users, and Table 4 is associated with events. These tables outline the routes specific to this application, specifying the HTTP method and corresponding URL pattern that triggers the execution of a particular function or controller. Here is a basic explanation of the HTTP methods employed in this project:

- The GET method retrieves a representation of a resource from the backend.
- The POST method submits or sends data to the backend to create a new resource.
- The PATCH method updates a resource on the backend.
- The DELETE method removes a specified resource from the backend.

As illustrated in Figure 19, each route is stored in its own file.

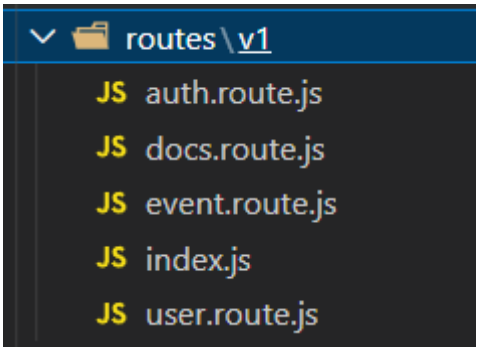


Figure 19. Route folder.

4 RESULTS

4.1.1 Header and Navigation Bar

The header and navigation bar are both common layout elements on a website. The header typically displays at the top of a webpage and contains essential elements such as the site title, navigation, and user account information in this web application. The navigation bar is an area of the header that contains a collection of links or buttons that allow users to navigate to other sections or pages of the website. It gives users with a clear and organized manner for accessing different parts of the site. Figure 20 shows the header, which consists of the site title and navigation bar. The navigation bar leads to the home and login pages.

Figure 20. Header and Navigation Bar.

Figure 21 demonstrates how the header and navigation bar change after logging in. The navigation bar allows to the profile modification profile, creation of new events, and returning to the main page.

Figure 21. Header and navigation bar after login.

The components folder, which contains the styles and functions used to create the header and navigation bar, is shown in Figure 22 below.

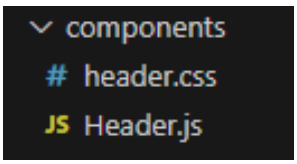


Figure 22. Components file.

4.1.2 Register and Login Page

A login page is an important part of any website or web application since it allows users to authenticate and obtain access to their personalized accounts or restricted content.

Figure 23 shows the Login screen after clicking on the Sign in button in the navigation bar. If the user already has an account, they sign in with the email and password they created previously. If they do not have any accounts, clicking 'register now' will take them to the registration screen, as seen in Figure 24. Users must enter their email address, password, and confirm their password. Then they upload their avatar, select their country of residence, and click the Register button. They will be taken directly to the home page.

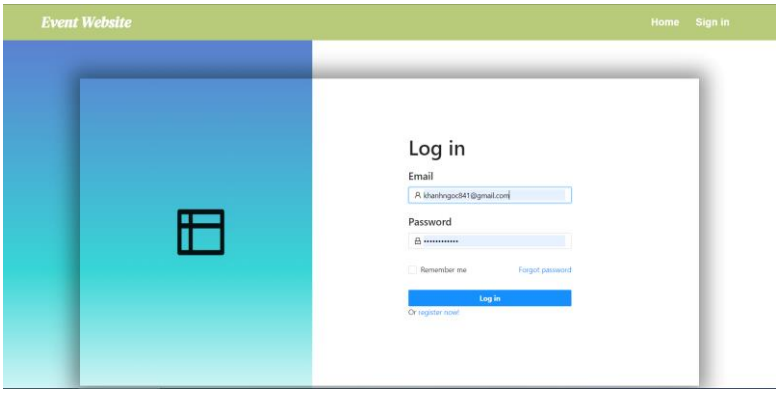


Figure 23. Login page.

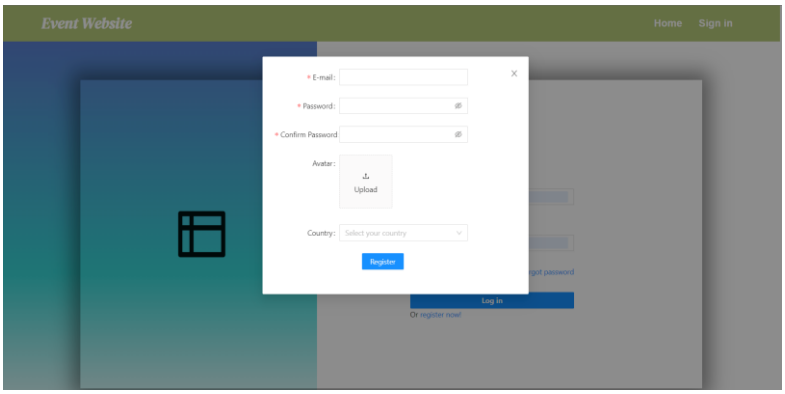


Figure 24. Register page.

4.1.3 Home Page

The home page is the main or default page that users see when they visit a website or launch a web application. It acts as an entry point to the site and often includes an overview of the content, significant features, or important information of the site. Figures 25 and 26 show the home page before and after login, with no changes other than a banner. The home page of this web application contains crucial information, especially a listing of events. Every event includes the event title, a brief description of the event, the date and time, a photo, and the person who posted the event.

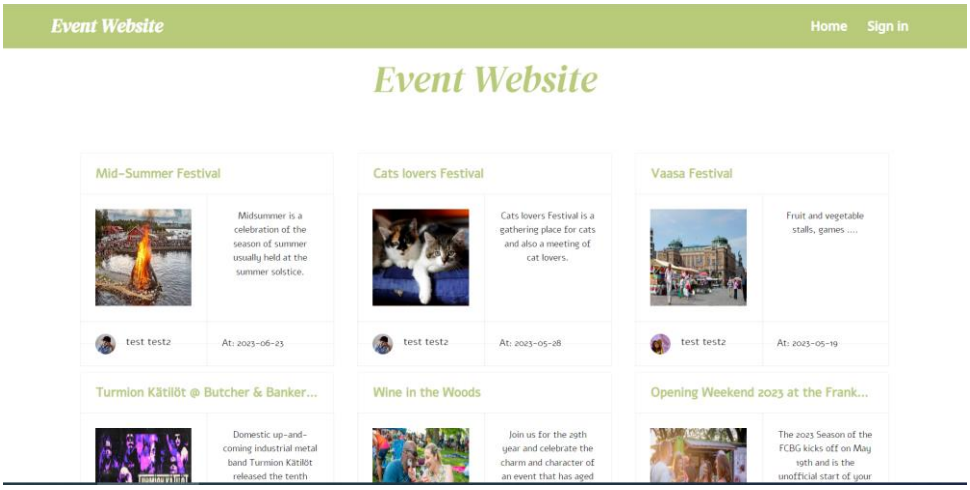


Figure 25. Home page

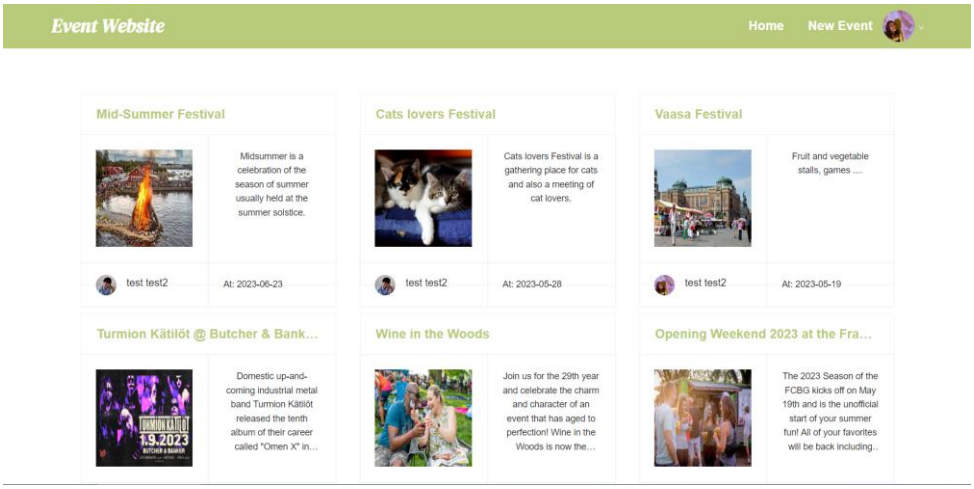


Figure 26. Home page after login.

When users click into an event to access all its details, a pop-up page appears. It will be displayed in its entirety, including all the event's descriptions, its website, and the country where it is held.

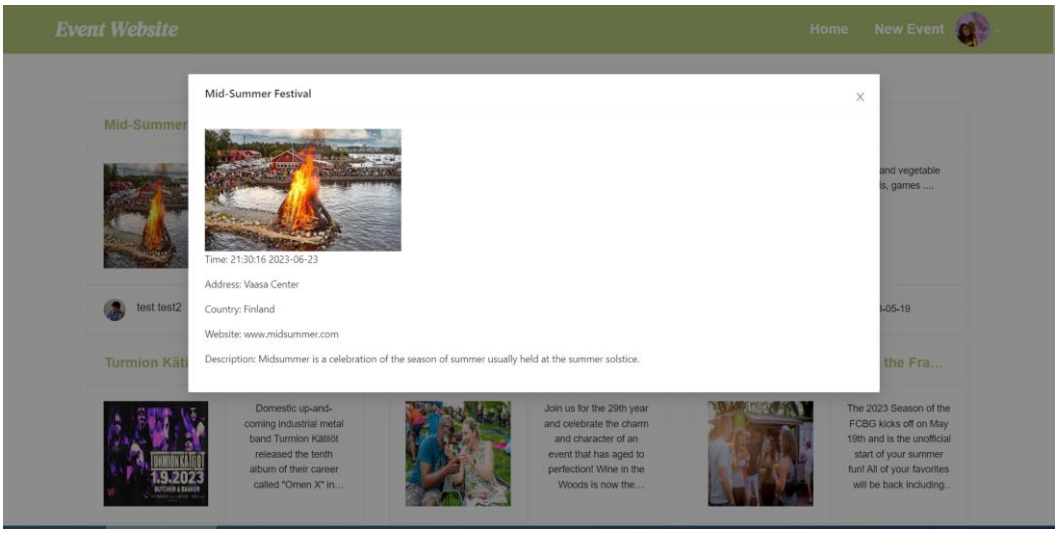


Figure 27. Pop-up page.

Figure 28 depicts the home directory, which contains all of the functions that allow the home page to function properly, as well as the page styling.

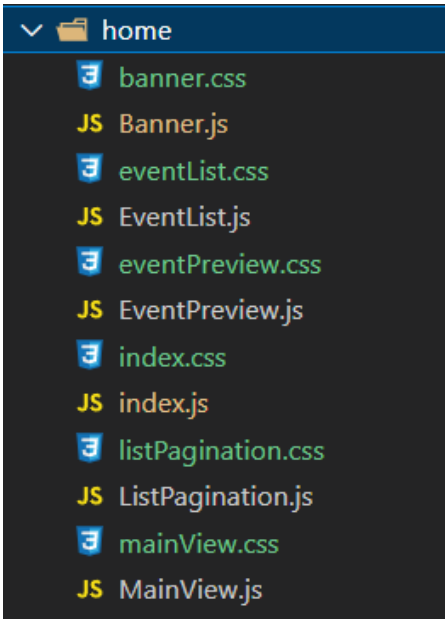


Figure 28. Home page folder.

4.1.4 Creating a New Event

Figure 29 shows the information that users must provide after clicking the 'New Event' button.

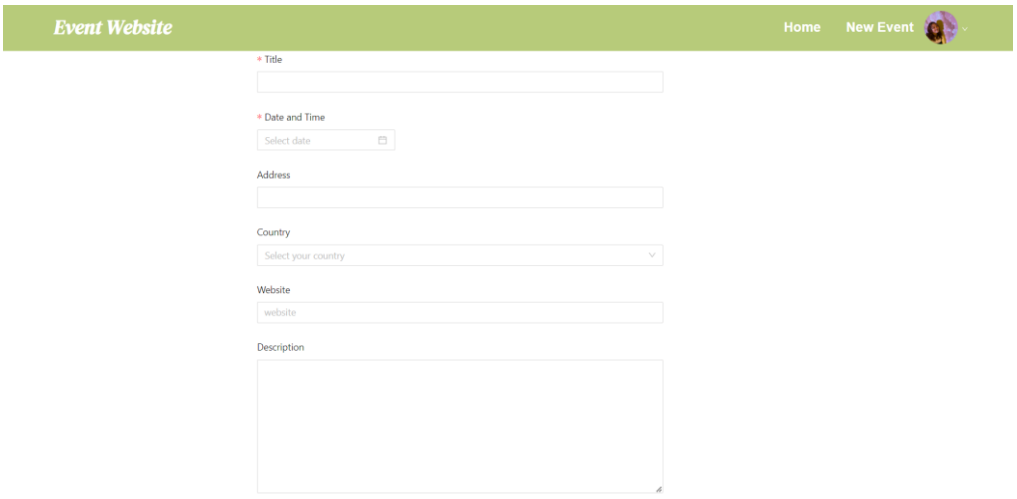


Figure 29. New event creation.

Then, a title and can be entered and a date and time sequence selected. Figure 30 depicts how to choose a date and time.

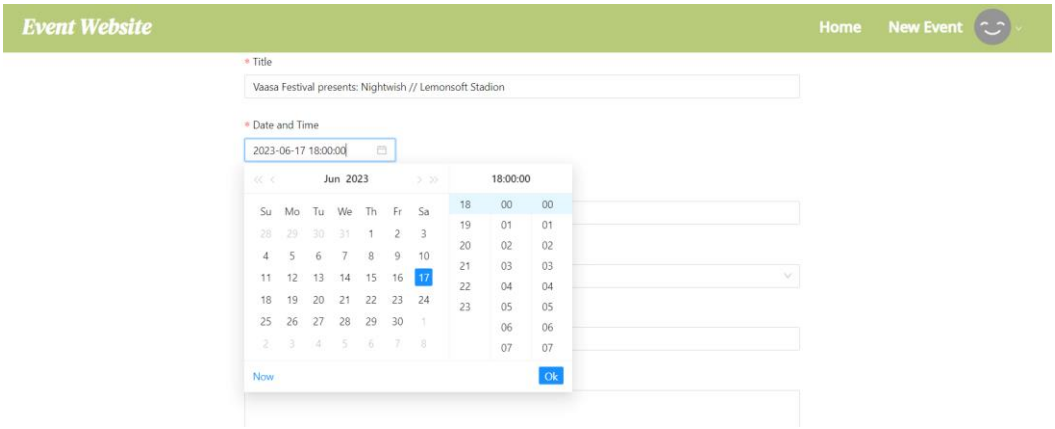


Figure 30. Choose date and time.

All the information is filled in as shown in Figure 31. After that, a photo can be added and the 'Submit' button pressed, as shown in Figure 32.

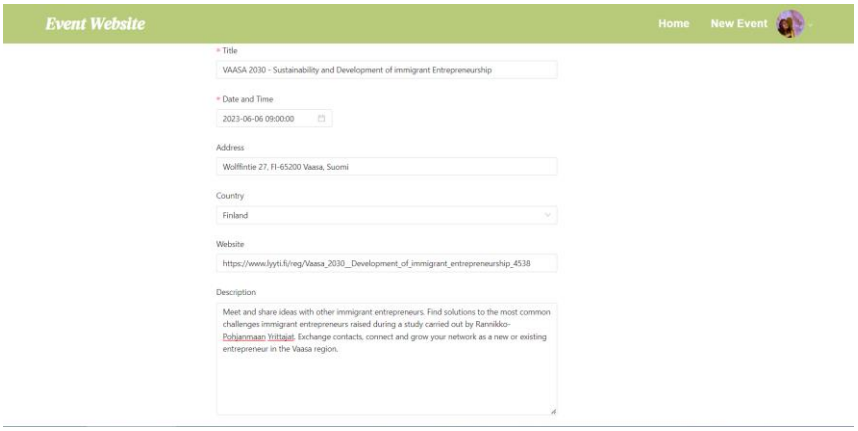


Figure 31. Filling in all the information.

Figure 32. Adding a photo.

After submitting the form, a notification will appear in the upper right corner of the page, as shown in Figure 33, alerting users that the event has been filed and is awaiting approval, which they can view on the event management page.

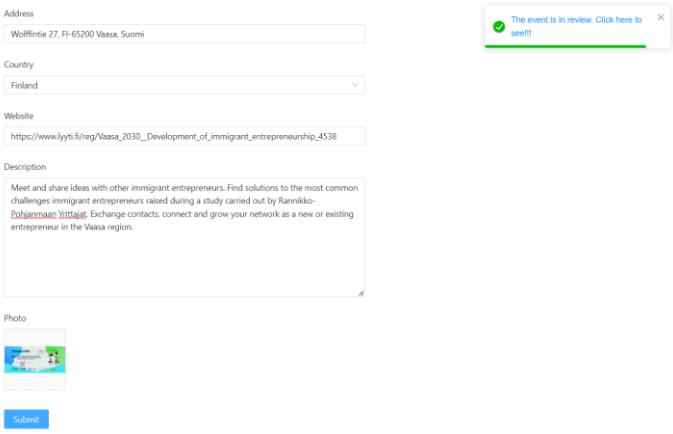


Figure 33. Notification after clicking Submit.

4.1.5 Event Management

The events that are awaiting clearance are displayed on the event management web page. Users can view and even remove events from this page. This page can be accessed in two ways: by clicking the notice as shown in Figure 30, or by clicking 'My events' as shown in Figure 32.

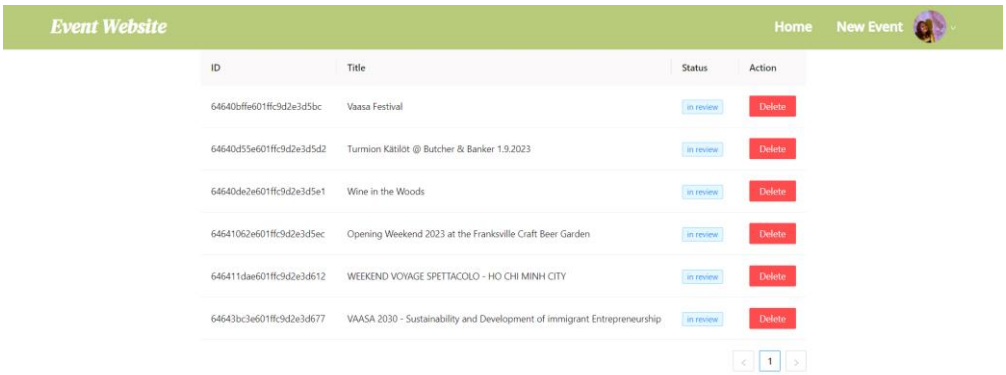


Figure 34. Event management.

4.1.6 User Management

After checking in, click into the avatar in the navigation bar to see elements that the user can manage, such as changing their information, managing their events, or even logging out, as seen in Figure 35.

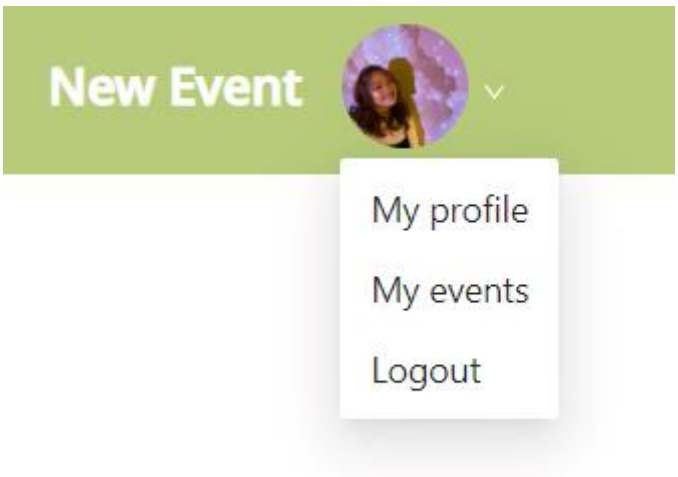


Figure 35. User management.

After selecting ‘My profile’, users are taken to the ‘My profile’ page, as seen in Figure 36. The avatar, password, and country of residence can be modified and updateed on this page.

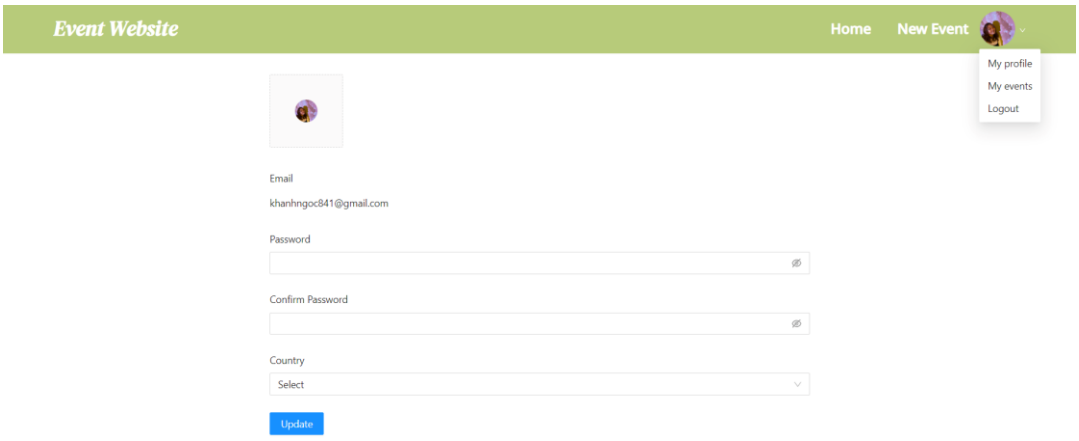


Figure 36. My profile page.

4.1.7 Admin page

Administrators must use the email address and password they previously gave to access the admin page. Following successful login, the homepage will show up as depicted in Figure 37. There are some differences between the user page and the event page in the navigation bar, such as 'Manage Users' and 'Manage Events'.

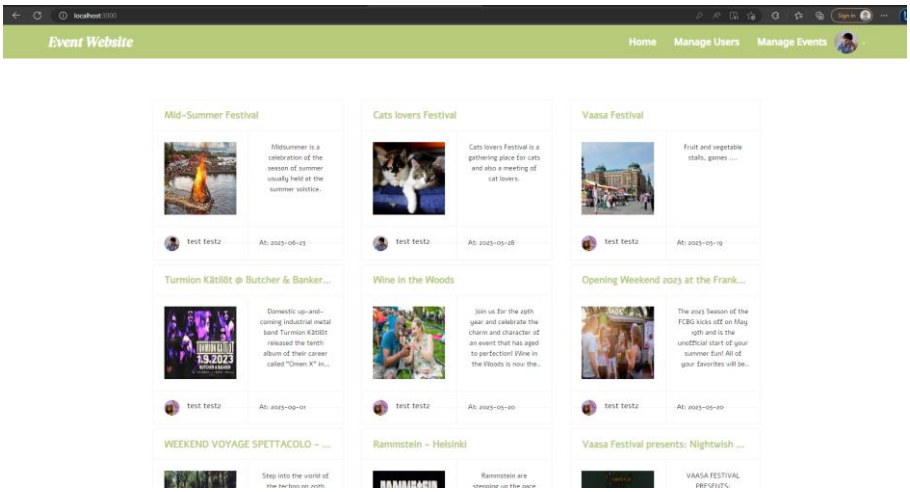


Figure 37. Admin home page.

A new event is ready to be approved in the 'Manage Events' section illustrated in Figure 38. The admin has the option to approve or delete this event there.

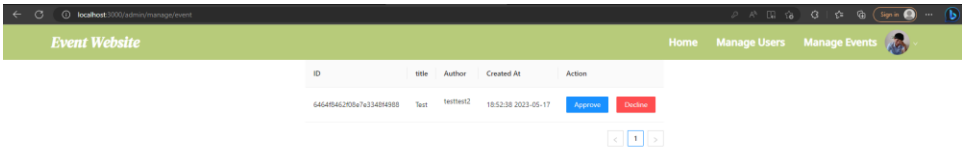


Figure 38. Manage Events page.

When an administrator clicks the 'Manage Users' button in the navigation bar, all previously created users are displayed on this page, as shown in Figure 39. The administration has the authority to ban users, as shown in Figure 40. After pressing the 'Ban' button, a notification will display in the upper right corner, indicating that the user has been banned.

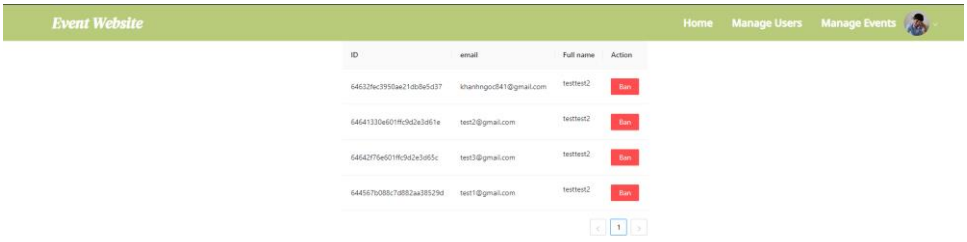


Figure 39. Manage Users page.

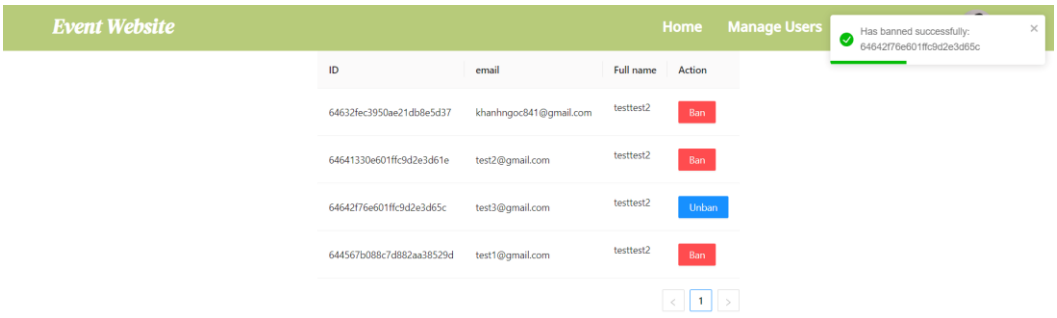


Figure 40. Admin ban user.

The banned user is unable to access the system. As seen in Figure 41, the notification will indicate that the account has been blocked.

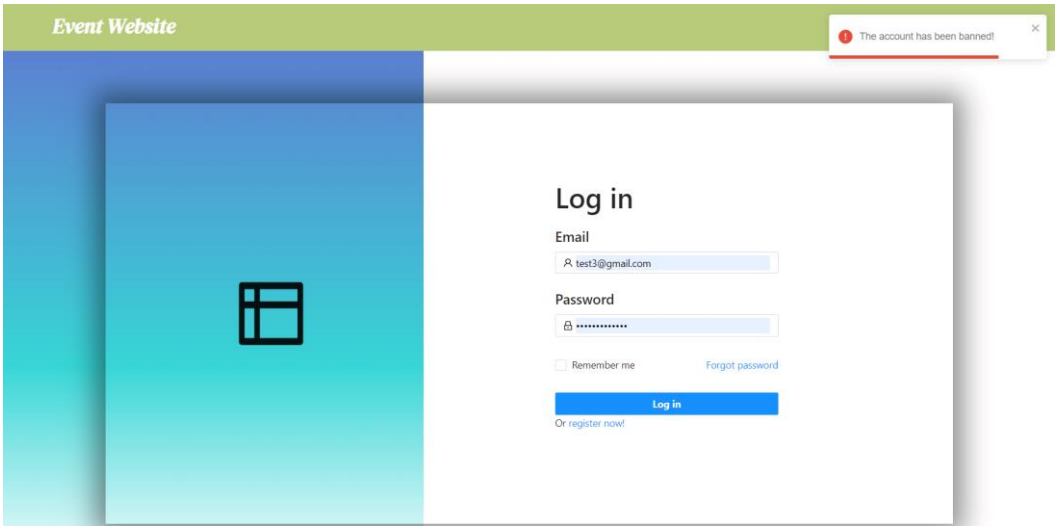


Figure 41. Banned account.

5 TESTING

There are many different frameworks and tools available to help with testing a MERN (MongoDB, Express, React, Node.js) stack application. Jest was utilized for the back end, a popular JavaScript testing framework that provides an easy and intuitive approach to write unit tests, integration tests, and API tests for the Node.js code.

React Testing Library is an option for front-end testing. It is a testing package created exclusively for testing React components. It contains utilities for rendering components, simulating user interactions, and making claims about the generated output. Jest is another good option. Jest may also be used to test React components by taking advantage of its snapshot testing capability, which records a component's expected output and compares it to the actual output during subsequent test runs.

Jest is a database testing tool that allows creating test cases that communicate with the MongoDB database to test database-related functionality. Mock database connections or test databases can be used to isolate and verify database operations.

6 DISCUSSION

6.1 Results

Demonstrating the effectiveness and capabilities of the MERN stack, a web application for searching local events has been successfully developed using this technology stack. The thesis provides detailed explanations on the utilization of MongoDB as the database, Express.js as the web framework, React as the front-end library, and Node.js as the server-side runtime environment. To ensure the delivery of a robust and feature-rich application, various tools and packages were employed in conjunction with these technologies. The seamless integration between the different components of the application facilitated efficient data management, smooth user interactions, and rapid development.

The MERN stack's flexibility and scalability enabled the implementation of several features, including event search, filtering, user authentication, and admin functionalities. The thesis comprehensively delved into the technical aspects of each technology and provided detailed explanations on how they were utilized during the development process.

6.2 Further Improvements

There are areas where further improvements can be made to enhance its functionality and user experience of the application.

Firstly, implementing a comprehensive testing strategy, including unit tests and end-to-end tests, will ensure the application's reliability and identify any potential issues or bugs.

Secondly, streamlining the deployment process through automation tools such as Docker and CI/CD will simplify updates and maintenance, allowing for easier scalability.

Thirdly, adding a search engine feature will enable users to perform advanced searches and filter events based on location, date, category, or keyword, improving event discovery.

Additionally, incorporating sorting options by categories will enhance usability and make it easier for users to browse events based on their interests.

Lastly, integrating user feedback and reviews will provide valuable insights for ongoing improvement and help users make informed decisions.

By addressing these areas of improvement, the web application can deliver a more robust and user-friendly experience for individuals searching for local events.

7 CONCLUSIONS

The initial purpose of this project is to develop a web application that assists users in finding events near them when they travel and to promote tourism. This web application uses the MERN stack to provide an efficient and scalable foundation for managing event data and delivering a responsive user interface. The use of MongoDB as the database ensures that event information is stored and retrieved efficiently. The usage of Express.js and Node.js allows for the creation of a dependable back-end infrastructure capable of handling user requests, authentication, and data processing. React.js as the front-end framework provides the application with a dynamic and interactive user experience.

In conclusion, the thesis introduces a web application designed to help people discover and engage in local events. With its seamless user experience, the application empowers individuals to explore and participate in a diverse array of activities in their community. It is a valuable tool for anyone seeking to get involved and stay connected with their local area.

REFERENCES

14. Routing - Web Development with Node and Express [Book]. (n.d.). Retrieved from O'Reilly: https://www.oreilly.com/library/view/web-development-with/9781491902288/ch14.html

About | Node.js. (n.d.). Retrieved from Node.js: https://nodejs.org/en/about

Getting Started | Axios Docs. (n.d.). Retrieved from Axios: https://axios-http.com/docs/intro

How to Use Axios with React: The Ultimate Guide [2023]. (2023, January 5). Retrieved from KnowledgeHut: https://www.knowledgehut.com/blog/web-development/axios-in-react

HTML: HyperText Markup Language | MDN. (2023, May 9). Retrieved from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/HTML

JSON Web Token Introduction - jwt.io. (n.d.). Retrieved from JWT.io: https://jwt.io/introduction

Karnik, N. (2018, February 11). *Introduction to Mongoose for MongoDB*. Retrieved from freeCodeCamp: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/

Learn to style HTML using CSS - Learn web development | MDN. (2023, February 23). Retrieved from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Learn/CSS

Panchal, K. (n.d.). *Angular vs React Detailed Comparison 2022*. Retrieved from Groovy Web: https://www.groovyweb.co/blog/angular-vs-react-detail-comparison/

Patil, P. (2023, March 19). *What is MongoDB? A Complete Guide » Coding* . Retrieved from Coding Torque: https://codingtorque.com/what-is-mongodb/

Perveez, S. H. (2023, January 30). *What is Git: Features, Command and* . Retrieved from Simplilearn: https://www.simplilearn.com/tutorials/git-tutorial/what-is-git

React Components. (n.d.). Retrieved from W3Schools: https://www.w3schools.com/react/react_components.asp

React: The Virtual DOM. (n.d.). Retrieved from Codecademy: https://www.codecademy.com/article/react-virtual-dom

Redux Toolkit: Overview | Redux. (2023, March 6). Retrieved from Redux: https://redux.js.org/redux-toolkit/overview

Use-case diagrams in UML modeling. (n.d.). Retrieved from IBM: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case

What is Authentication? Definition of Authentication, Authentication . (n.d.). Retrieved from The Economic Times: https://economictimes.indiatimes.com/defaultinterstitial.cms

What is Express.js? | Why should use Express.js? | Features of Express.js. (n.d.). Retrieved from Besant Technologies: https://www.besanttechnologies.com/what-is-expressjs

What Is Express.js? Everything You Should Know. (n.d.). Retrieved from Kinsta: https://kinsta.com/knowledgebase/what-is-express-js/

What is JavaScript? - Learn web development | MDN. (2023, May 10). Retrieved from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

What Is Node.js and Why You Should Use It. (n.d.). Retrieved from Kinsta: https://kinsta.com/knowledgebase/what-is-node-js/

What is Swagger? Definition from WhatIs.com. (n.d.). Retrieved from TechTarget: https://www.techtarget.com/searchapparchitecture/definition/Swagger

What Is The MERN Stack? Introduction & Examples. (n.d.). Retrieved from MongoDB: https://www.mongodb.com/mern-stack

What is version control. (n.d.). Retrieved from Atlassian: https://www.atlassian.com/git/tutorials/what-is-version-control

Why Use MongoDB And When To Use It? (n.d.). Retrieved from MongoDB: https://www.mongodb.com/why-use-mongodb

Why Visual Studio Code? (n.d.). Retrieved from Visual Studio Code: https://code.visualstudio.com/docs/editor/whyvscode