CHAPTER 1

INTRODUCTION

Our Event Calendar project addresses the challenges of event organization and management, becoming a critical tool for enhancing productivity and organization in both personal and professional contexts. Introducing a revolutionary approach to event planning and management, the Event Scheduler provides a comprehensive, user-friendly platform that serves individuals, businesses, and organizations.

React Event Calendar, an innovative application designed to streamline the process of managing events with efficiency and ease. Built using the powerful React framework, this scheduler offers a user-friendly interface and robust features that cater to both individual and organizational needs. In this website, we can add, delete, create, and update the events. And also we can share the events to friends,family etc through social media platform such as Instagram, LinkedIn, Facebook, Twitter.

Designed to streamline event scheduling and ensure smooth execution, the Event Calendar enhances productivity and reduces stress by efficiently managing time and events. While traditional methods like paper planners and basic digital calendars fall short, our advanced Event Scheduler meets the dynamic needs of modern users with unparalleled functionality and ease of use .While traditional methods like paper planners and basic digital calendars fall short, our advanced Event Scheduler meets the dynamic needs of modern users with unparalleled functionality and ease of use.

CHAPTER 2

PROBLEM STATEMENT

PROBLEM STATEMENT

- People struggle with fragmented organization, relying on a mix of paper calendars, basic reminder apps, or mental juggling to keep track of their schedules.
- Managing events and reminders is often chaotic, leading to missed appointments and poor time management due to complex and non-integrated calendar solutions like User-Friendly Interface, consolidate events and reminders, Seamless Integration, Secure Access.

SOLUTION

- The React Event Calendar addresses the issue of fragmented organization by providing a unified, feature-rich platform for managing all scheduling needs.
- By consolidating events, reminders, and tasks into one cohesive system, it helps users maintain a well-organized schedule, reduces stress, and increases productivity.
- This holistic approach to event management ensures that users can focus on their tasks and commitments without the worry of missing important appointments or deadlines.
- Unified Scheduling Platform, the React Event Calendar serves as a single, cohesive platform where users can consolidate all their scheduling needs.
- Comprehensive Event Management, users can create, edit, and delete events with ease. The scheduler supports various event types, allowing users to categorize and color-code their schedules.
- This visual distinction helps in quickly identifying different types of events, such as work meetings, personal appointments, or social gatherings.

CHAPTER 3

GOALS AND OBJECTIVES

3.1 Goals:

- To develop and implement an efficient online calendar system for adding, viewing, and managing events which allows users to plan and organize events accordingly.
- Ensure calendar is interactive and responsive across various devices (i.e., computers, tablets and mobile devices).
- It helps users manage personal and professional schedules and ensure they never miss important dates.
- Enhancing user experience by providing an interface that is easy to navigate and use regardless of the user's technical proficiency.
- Improving efficiency in the process of event scheduling by reducing the time and effort required to manage events.

3.2 Objectives to achieve:

- An user-friendly interface that simplifies event creation, updates and modifications and allow users of all technical levels to use the scheduler with ease.
- Use a reliable database to store event details securely and with regular backups to prevent data loss.
- Implement search and filter functionalities for easy event lookup.
- Implement user authentication to allow users to login and manage their personalized event calendars to protect user data and privacy.
- Ensure the scheduler supports various types of events (meetings, appointments, remainders) and enable users to create, edit and delete events with necessary details such as date, time and description.
- Ensuring flexibility by offering customizable features to a wide range of scheduling needs from personal appointments to large-scale events.

CHAPTER 4

REQUIREMENTS

The frontend of the Event Calendar Website will be crafted using a combination of HTML, CSS, JavaScript, and React.js. The backend of the Event Calendar Website will be developed using Node.js with Express.js. The database for the Event Calendar Website will be MongoDB.

4.1 Frontend: HTML, CSS, JavaScript, React.js

The frontend of the Event Calendar Website will be crafted using a combination of HTML, CSS, JavaScript, and React.js to deliver a dynamic and interactive user interface. HTML will form the structural backbone of the website, defining the layout and essential elements. CSS will be employed to style the application, ensuring it is visually appealing and user-friendly across different devices and screen sizes. JavaScript will add interactivity, enabling features such as real-time data updates and user interactions. React.js, a powerful JavaScript library, will be used to build the user interface with a component-based architecture. This approach not only promotes reusability and maintainability of code but also enhances the user experience by providing fast and responsive UI updates through its virtual DOM. React state management capabilities will allow seamless data flow and synchronization within the application, ensuring users have a smooth and engaging experience when browsing, searching, and managing events.

4.2 Backend: Node.js with Express.js

The backend of the Event Calendar Website will be developed using Node.js with Express.js, forming a robust and efficient environment for server-side logic and data management. Node.js provides a non-blocking, event-driven architecture, which is ideal for building scalable network applications. Express.js, a minimalist web framework for Node.js, will be used to streamline the development process by providing a set of robust features for web and mobile applications. It will handle routing, middleware, and HTTP requests, ensuring smooth communication between the frontend and backend. This combination allows for efficient handling of user authentication, authorization, event management, and real-time updates.

Node.js with Express.js will facilitate the development of RESTful APIs, ensuring that the client-side can seamlessly interact with the server, perform CRUD operations, and maintain data integrity. This backend setup ensures the application can handle high traffic volumes and provide quick, reliable responses to user actions.

4.3 Database: MongoDB

The database for the Event Calendar Website will be MongoDB, a leading NoSQL database known for its scalability and flexibility in handling large volumes of unstructured data. MongoDB stores data in JSON-like documents, which makes it a perfect fit for applications that require dynamic data schemas and need to handle a variety of data types. This database will store user data, event details, and reminders, providing high performance and quick access to information. MongoDB's ability to horizontally scale across many servers ensures that the application can grow with an increasing number of users and events without compromising on performance. Additionally, MongoDB's robust querying capabilities and indexing features will allow for efficient data retrieval and manipulation, which is crucial for real-time applications. The use of MongoDB ensures that the Event Scheduler Website can maintain high availability, reliability, and consistency of data, enhancing the overall user experience and operational efficiency.

CHAPTER 5

DESIGN

5.1 Use Case Diagram

Use case diagram below represents the functionalities of an Event Calendar Website, delineating actions that can be performed by Users and Admins. This detailed diagram includes both user-facing and administrative functions to ensure clarity on the roles and responsibilities within the application.

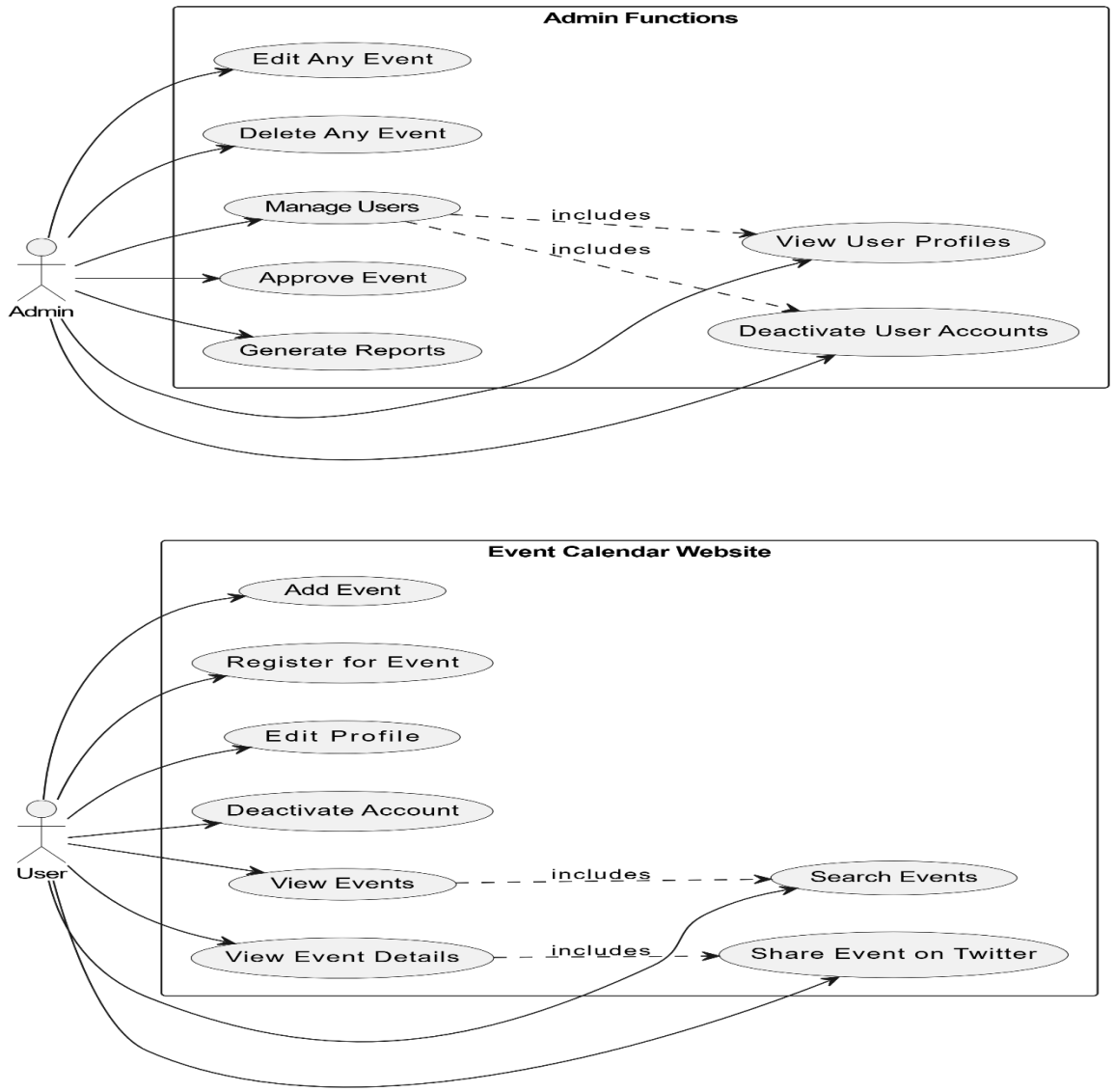

FIGURE 5.1: Use Case Diagram for Event Calendar Website

Actors

User: Represents a typical user of the Event Calendar Website, performing various actions like viewing, adding, and registering for events.

Admin: Represents an administrative user with elevated permissions to manage events and users, ensuring the smooth operation and moderation of the platform.

5.2 Use Cases for User

- View Events: Allows the user to browse a list of events available on the platform.
- Search Events: Enables the user to search for specific events based on various criteria.
- View Event Details: Provides detailed information about a specific event, such as date, time, location, and description.
- Add Event: Allows the user to create and add a new event to the platform.
- Register for Event: Enables the user to register their attendance for an event.
- Share Event on Twitter: Provides the option to share event details on Twitter for broader visibility.
- Edit Profile: Allows the user to modify their profile information.
- Deactivate Account: Enables the user to deactivate their account if they choose to leave the platform.

5.3 Use Cases for Admin

- Edit Any Event: Grants the admin the ability to edit any event details on the platform, regardless of who created it.
- Delete Any Event: Allows the admin to remove any event from the platform.
- Manage Users: Provides the admin with tools to manage user accounts and their activities on the platform.
- Approve Event: Grants the admin the authority to approve events before they become publicly visible.
- Generate Reports: Enables the admin to generate various reports related to events and user activities for analysis and decision-making.
- View User Profiles: Allows the admin to view detailed profiles of users on the platform.
- Deactivate User Accounts: Enables the admin to deactivate user accounts if necessary, for instance, in cases of misconduct or policy violations.

5.4 Relationships and Inclusions

- View Events includes Search Events: Viewing events inherently involves the ability to search for specific events based on user-defined criteria.
- View Event Details includes Share Event on Twitter: When viewing event details, the user can choose to share the event on Twitter.
- Manage Users includes View User Profiles and Deactivate User Accounts: Managing users involves viewing their profiles and, if necessary, deactivating their accounts.

Chapter 6

IMPLEMENTATION

The application integrates a frontend built with React and Apollo Client for managing GraphQL365 queries. It ensures user session management with an idle timer component and handles routing via React Router. The backend, powered by Express

6.1: Frontend code

```
import { ApolloProvider } from '@apollo/client'
import client from './apollo'
import UserIdleTimer from './components/UserIdleTimer/UserIdleTimer'
import { useContext } from 'react'
import AuthContext from './store/auth-context'
import AppRoutes from './Routes'
import { Container } from 'react-bootstrap'
import Footer from './components/Footer/Footer'

import './App.css'
import { Toaster } from 'react-hot-toast'
import styled from 'styled-components'
import { useMatch } from 'react-router-dom'

const RoutesContainer = styled(Container)`
  min-height: calc(100vh - 85px);
`

function App() {
  const { auth, removeAuth } = useContext(AuthContext)
  const welcomePagePath = useMatch('/')

  return (
```

```
    <>
      <RoutesContainer>
        {auth && <UserIdleTimer onLogout={removeAuth} />}

        <ApolloProvider client={client}>
          <AppRoutes />
        </ApolloProvider>
      </RoutesContainer>
      {!welcomePagePath && <Footer />}
      <Toaster />
    </>
  )
}

export default App
```

6.2: Backend code

```
import { ApolloServer } from '@apollo/server'
import { expressMiddleware } from '@apollo/server/express4'
import { ApolloServerPluginDrainHttpServer } from '@apollo/server/plugin/drainHttpServer'
import compression from 'compression'
import express from 'express'
import enforce from 'express-sslify'
import path from 'path'
import cors from 'cors'
import dotenv from 'dotenv'
import cookieParser from 'cookie-parser'
import http from 'http'

import { connect, set } from 'mongoose'
import { rootValue } from './graphql/resolvers'
import { typeDefs } from './graphql/schema'
import { json, urlencoded } from 'body-parser'
```

```
import { constants } from './config/constants'
import { context } from './middleware/auth'
import { IContext } from './interfaces/types'

dotenv.config()

const { ENV, PORT, URI, MONGODB_URI } = constants

const corsOptions = {
  origin: URI,
  credentials: true,
}

const app = express()

// enforce https for production
if (ENV === 'production') {
  app.use(enforce.HTTPS({ trustProtoHeader: true }))
}

app.use(cors(corsOptions))
app.use(cookieParser())
app.use(compression())

app.use(json())
app.use(urlencoded({ extended: true }))

app.use('/', express.static(`${__dirname}/../build`))

app.get('*', (req, res) => {
  res.sendFile(path.join(__dirname, '../build', 'index.html'))
})
```

```
const startServer = async () => {
  const httpServer = http.createServer(app)
  const apolloServer = new ApolloServer<IContext>({
    typeDefs,
    rootValue,
    plugins: [ApolloServerPluginDrainHttpServer({ httpServer })],
  })

  await apolloServer.start()
  app.use(
    '/graphql',
    cors<cors.CorsRequest>(),
    json(),
    expressMiddleware(apolloServer, {
      context,
    }),
  )

  try {
    set('strictQuery', false)
    await connect(MONGODB_URI)

    await new Promise<void>((resolve) =>
      httpServer.listen({ port: PORT }, resolve),
    )
    console.log(`🚀 Server ready at http://localhost:${PORT}/graphql`)
  } catch (err) {
    console.error('Error occured while connecting to MongoDB: ', err)
  }
}

startServer()
```

CHAPTER 7

TESTING

Testing is a vitally indispensable phase in the React-Event Calendar project's development to ensure that it really does do it all establishing that the code meets the requirements prescribed and giving the user a seamless experience. It encompasses all the testing types executed, the whole testing strategy, and the end results that are a guarantee that the application is of good quality, works fast, and is easy to use.

7.1 KINDS OF EXAMINATION

7.1.1 Unit Testing:

- A means to determine that each component and function operates independently.
- Test the scheduler's ability to handle different types of events (one-time, recurring, all-day events, etc.).

7.1.2 Integration Testing:

- It confirms that all components like event creation, calendar view, and reminder notifications work perfectly when they are combined.
- If the event scheduler integrates with other systems (like calendars, email notifications), test these integrations thoroughly.

7.1.3 End-to-End (E2E) Testing:

- It explains the most basic user tasks such as event creation, editing, removal of the event.
- Test with unusual scenarios (e.g., overlapping events, events spanning across different time zones).

7.1.4 Performance Testing:

- It is used to check how the application is performing against different conditions. It aims at the load time, responsiveness, and scalability.
- Check how the scheduler handles errors or unexpected user inputs.

7.1.5 Usability Testing:

- Conduct usability tests with potential users to gather feedback on overall user experience and identify any usability issues.

7.2 Strategy of Testing

7.2.1 Test Planning

- Define the scope, objectives, and resources needed for testing. Identify and prioritize test cases for each type of testing based on the application's functionalities.

7.2.2 Test Design

- Develop detailed test cases and scenarios. Create automated tests for unit, integration, and E2E testing.

7.2.3 Test Execution

- Tests to make sure that the app can bear expected loads. Usability test is an excellent way to get feedback from many kinds of users and to make necessary changes.

7.2.4 Test Review and Closure

- Check the outcome of the tests with the development team.

By the strict practice of testing the React-Event Scheduler initiative has been officially pronounced as efficient, functional, and user-friendly. A systematic approach that included unit, integration, end-to-end, performance, and usability testing has been exercised to perfect the application to meet the highest quality standards.

The verification phase succeeded in guaranteeing that the application is solid and ready to be placed but also managed to provide key observations for future improvements ensuring that the highly reliable Tool for React-Event Scheduler remains a capability to manage events efficiently. This is done to analyse the user-friendliness and the overall user experience. Arrange feedback from the real users so that they can know the defects in usability and where the improvement is required.

CHAPTER 8

RESULTS

The implementation of the Event Calendar website yielded significant improvements in managing personal and professional schedules. The results were evaluated based on user feedback, performance metrics, and the overall usability of the system.

8.1 User Feedback

- **Ease of Use**: Users reported that the interface was intuitive and easy to navigate. The clear design and layout allowed users to quickly understand how to create, edit, and manage their events.
- **Efficiency**: Users appreciated the ability to set reminders and notifications, which helped them stay on top of their schedules without missing important events.
- **Accessibility**: The responsive design ensured that the website worked seamlessly across various devices, including desktops, tablets, and smartphones. This feature was particularly praised by users who needed to access their calendars on the go.

8.2 Performance Metrics

- **Load Times**: The website demonstrated fast load times, which improved the overall user experience. Optimization techniques used in the development process contributed to the website's quick responsiveness.
- **Reliability**: The backend infrastructure, built using Node.js and MongoDB, proved to be robust and reliable. There were no significant downtime or data integrity issues reported during the testing phase.

8.3 Usability

- **Event Creation and Management**: The ability to create detailed events with titles, descriptions, dates, times, locations, and tags was well-received. Users found the recurring events feature particularly useful for regular appointments and meetings.

- **Reminders and Notifications**: The customizable notification system was effective in reminding users of upcoming events. Both email and push notifications were successfully delivered, ensuring users were always informed.
- **Search and Filter**: The search and filter functionalities allowed users to quickly find specific events, enhancing the overall usability of the calendar. Color-coded events further helped in distinguishing between different types of events at a glance.

8.4 Key Metrics

- **User Retention**: High user retention rates were observed, indicating that users found the calendar useful and continued to use it regularly.
- **Engagement**: There was significant user engagement with the various features of the calendar, including event creation, reminders, and calendar views.
- **Error Rates**: Minimal error rates were reported, showcasing the stability and reliability of the platform.

CHAPTER 9

SNAPSHOTS

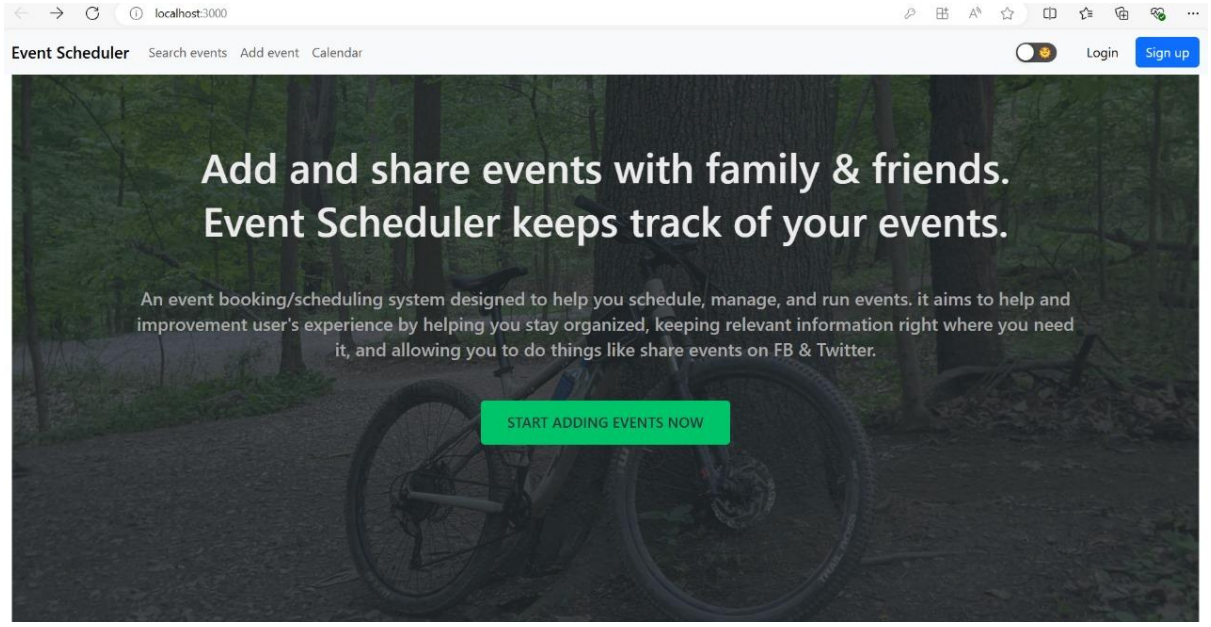

Fig 9.1:Dashboard of the website

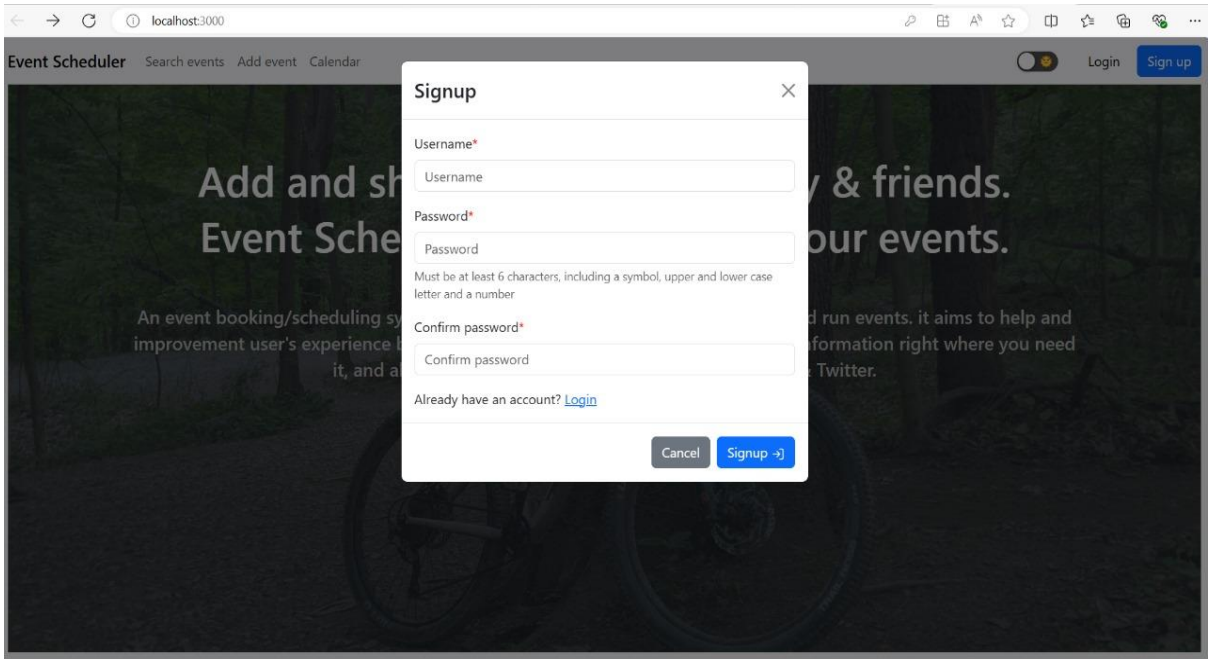

Fig 9.2:Signup page in the website

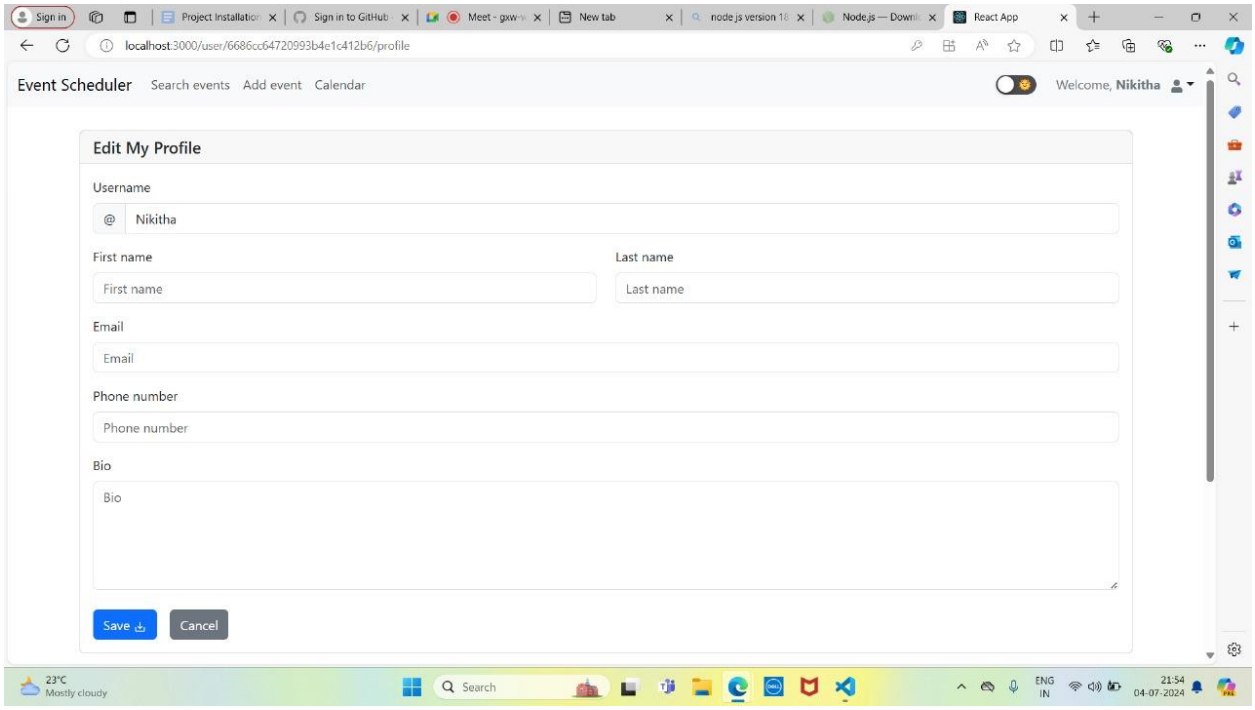

Fig 9.3:Page to edit the Profile

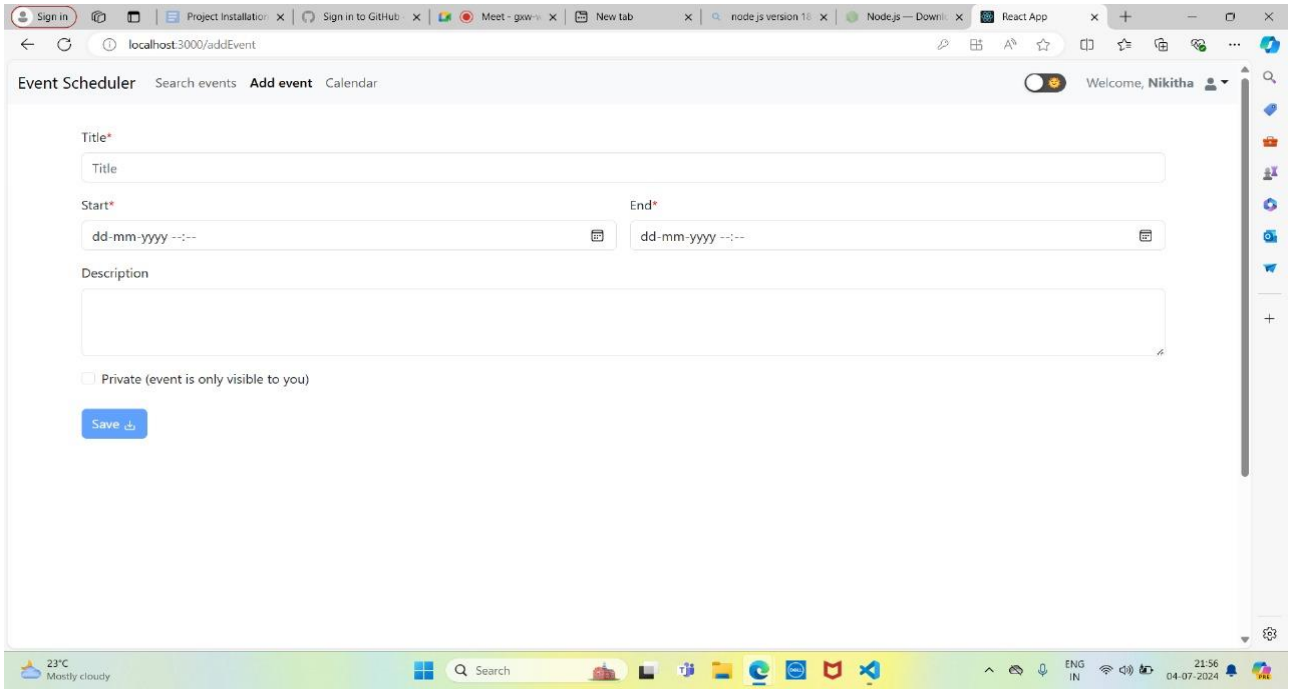

Fig 9.4: Task 1-To create or add an event

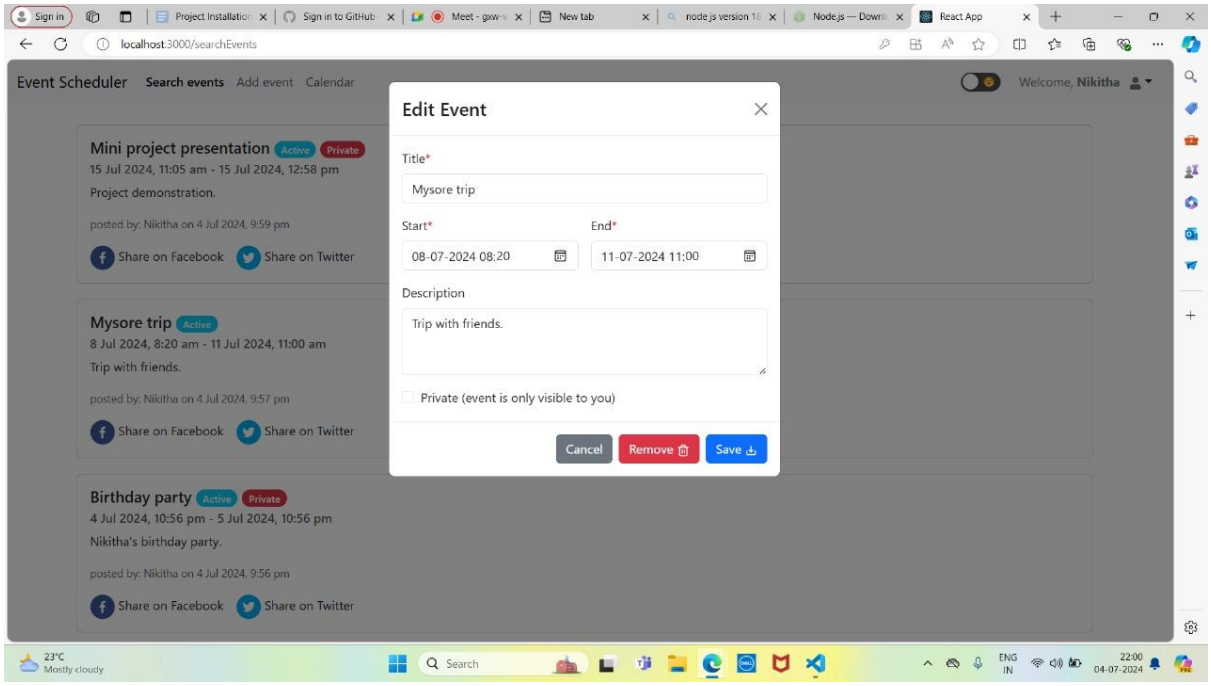

Fig 9.5: Task 2-To edit an event

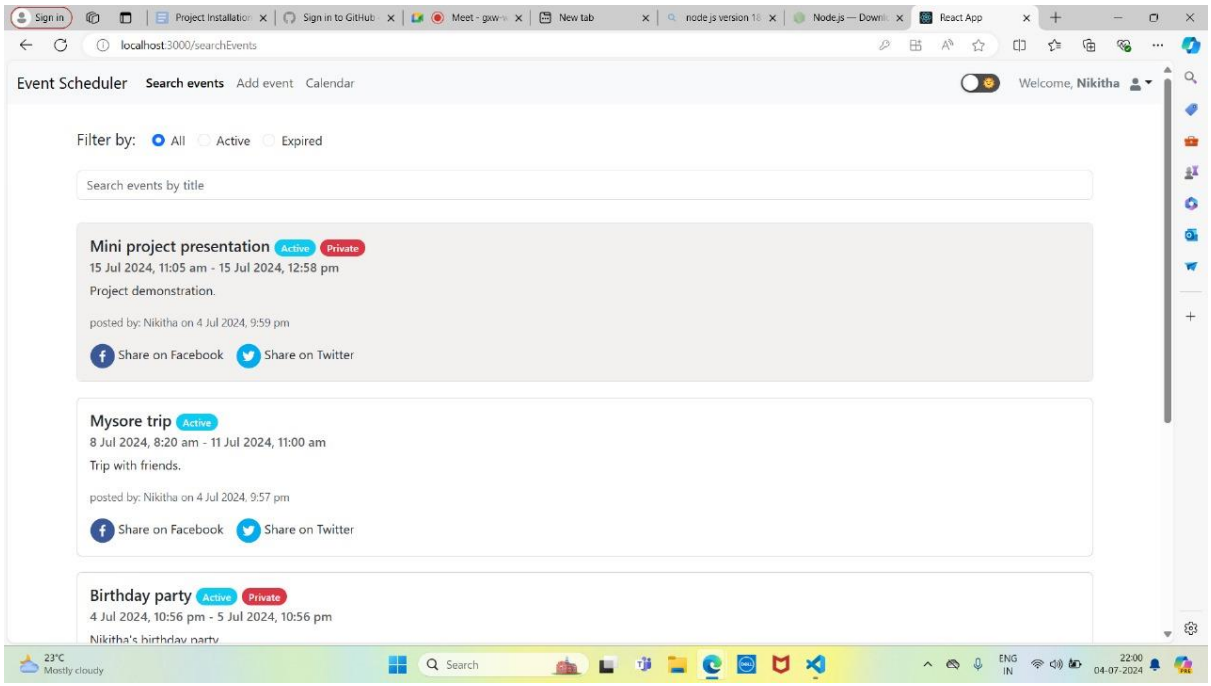

Fig 9.6: Task 3-To search an event

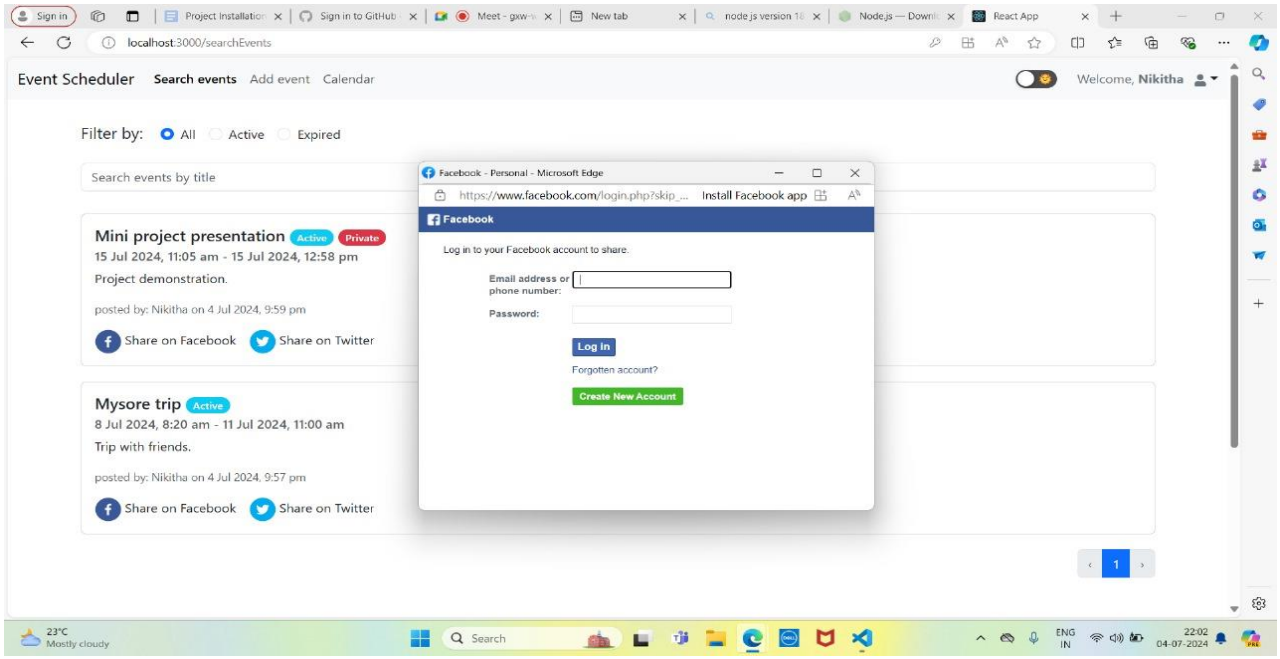

Fig 9.7: Sharing an event on Facebook

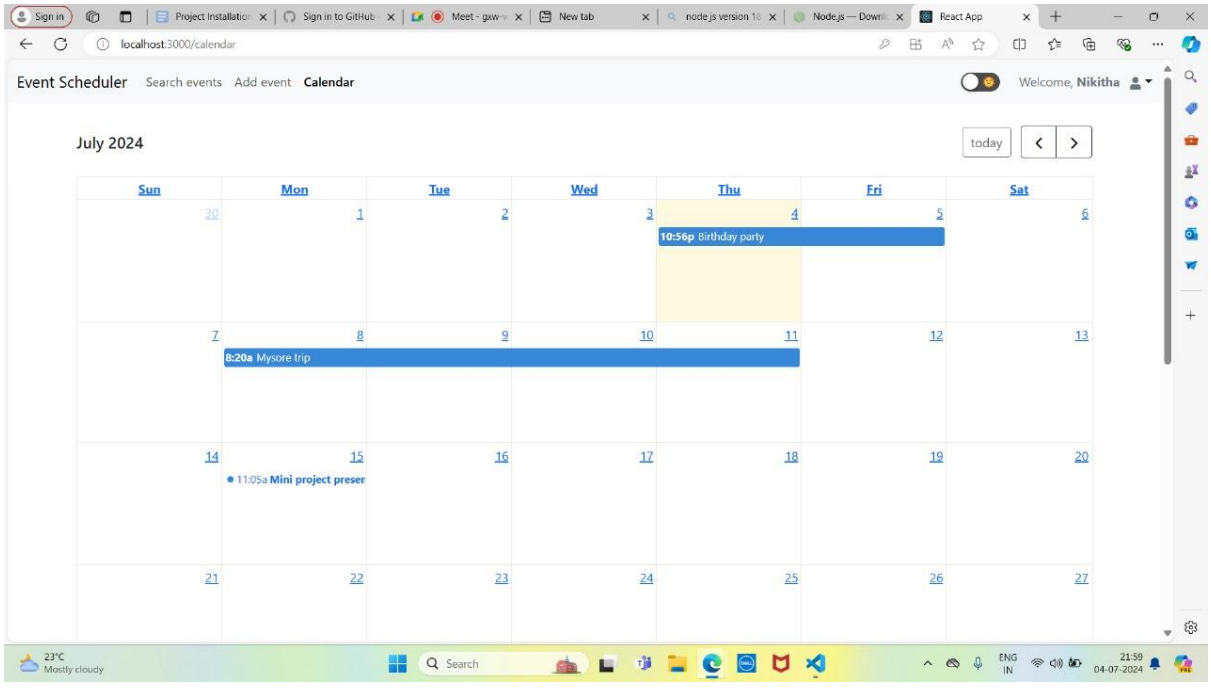

Fig 9.8: Event Calendar

CHAPTER 10

CONCLUSION

The detailed content of the Event Calendar Website is the result of a complete and thought-through effort to solve the drawbacks and limitations of old, conventional scheduling. Thus, the website is going to utilize modern web technologies in order to offer the users a maximum convenient and user-friendly platform for them, to utilize their personal and professional schedules more effectively.

This website will combine security with user authentication, by which the event management will be personalized and data privacy is maintained. They would be capable of making detailed events with titles, descriptions, dates, times and optionally they could also delete or edit these events. The facility to provide the recurring events feature through users to automate their scheduling, for that reason, people would make use of their time efficiently and wouldn't have the chance to forget their regular duties, week after week. Users can see their daily, weekly, and monthly schedules through the use of multiple calendar views, thus, making them more efficient in managing their time. The use of color-coded events will be a greater benefit by adding user experience through quick recognition of different types of events.

There is no doubt that the Event Calendar Website is going to reinvent the time management process of the users. Through secured authentication, event creation and management, notifications, and multiple calendar views, the people will experience all possible secure and fast approaches. Also, the unmistakable project plan and proactive response to potential difficulties further ensure the project's success.

In conclusion the Event Calendar website will not just help the user to do the work in great efficiency but also will be the most effective and easy in managing personal and professional events at all.

REFERENCES