OPOLE UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering, Automatics, and Informatics

Department of Computer Science

DIPLOMA THESIS

First-cycle Full-time Studies

Computer Engineering

DEVELOPMENT OF A WEB APPLICATION FOR MANAGEMENT OF PUBLIC EVENTS

Mentor:

DR HAB INŻ. WIESŁAW MARSZALEK

Author:

HIRENKUMAR SAVANI

Student ID: 98103

Opole, February 2022

Development of a web application for management of public events

Purpose and scope of work

The purpose of this thesis is to build a real-world public management event application that will be constructed with javascript. The aim of the application will be booking events online, registering events, talking with our partner to organize user events via a web application. To build the prototype technology like React, Nodejs, Expressjs, and MongoDB will be used.

Opracowanie aplikacji internetowej do zarządzania imprezami masowymi

Cel i zakres pracy

Celem tej pracy jest zbudowanie rzeczywistej aplikacji do zarządzania wydarzeniami publicznymi, która zostanie skonstruowana za pomocą javascript. Celem aplikacji będzie rezerwacja wydarzeń online, rejestracja wydarzeń, rozmowa z naszym partnerem w celu organizacji wydarzeń użytkownika za pośrednictwem aplikacji internetowej. Do budowy prototypu zostaną wykorzystane technologie takie jak React, Nodejs, Expressjs i MongoDB.

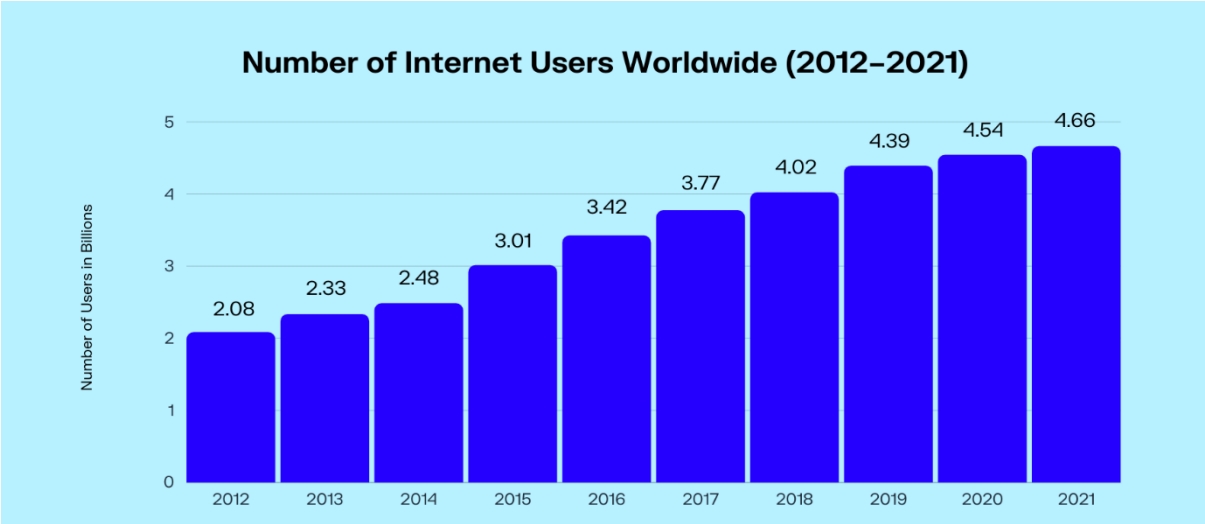
TABLE OF CONTENTS

Figures

Listings

1. INCEPTION

Technology's greatest asset is its ability to bring people together. About 20 years ago, no one could imagine meeting people while sitting at home, but technology has changed our thinking. Today, finding an industry that does not use the internet will be difficult.



Source: Oberlo - https://www.oberlo.com/statistics/how-many-people-use-internet

Figure 1.1 Number of internet users Worldwide (2012-2021)

The supplied figure 1.1 shows that the web is gaining market share and that this trend is unlikely to reverse. Analyzing this data reveals that online apps are increasingly replacing desktop programs, as well as other native applications.

However, web technology plays an important part in the internet and technology game. A web is a collection of electronic resources. The process of creating, producing, and administering websites is referred to as "web development." The package includes designing the web, publishing of web, programming of the web, and database administration. It entails the creation of an internet-based program. Tim Berners-Lee came up with the concept in 1989. HTML (hypertext markup language) can be used to make web pages, and CSS (case carding language) can be used to design web apps. JavaScript can be used to add functionalities to web pages, and after using JavaScript to construct a web page, it becomes a dynamic web page.

1.1 Intention of the work

The idea to create the EventEve application opens door to users to buy passes for the event and can register an event. Also introducing partners so if someone wants to organize their event, a partner can organize that on behalf of the application EventEve.

By doing this work done author also want to study the deep JavaScript concept, react life-cycle and create rest API (Application Programming Interface) in nodeJs.

1.2 Concrete targets

To achieve the targeted result it is necessary to have a proper plan and knowledge. So the following points need to be complete to achieve the goal.

- Investigating different techniques for web development.
- Examining both functional and non-functional requirements.
- Familiarizing with Visual studio code (IDE).
- Gaining an understanding of HTML.
- Gaining an understanding of CSS and designing principles.
- Gaining an understanding of the basics of the JavaScript programming language.
- Investigate different technology stacks and their benefits and drawbacks.
- Learning concepts of JSON.
- Learning about node package manager.
- Learning Reactjs Framework, life-cycle, components dealing with the different library for Front-end
- Learning Nodejs, HTTP request dealing, and creating the server.
- Learning Expressjs Framework, creating a server, creating REST full APIs, and dealing with different libraries for the back-end.
- Familiarizing with Robo 3T (GUI for MongoDB hosting deployments).
- Learning MongoDB database management system.
- dealing with external APIs.
- Methodologies for project implementation and testing

1.3 Project theme

In the accompanying parts of the work, the ensuing phases of planning and making the application have been depicted.

Chapter 2 will describe a basic introduction of and what are solutions available for the EventEve.

Chapter 3 will describe the requirements to accomplish the task.

Chapter 4 will describe how the web works and the basic ideas to make a website.

Chapter 5 will be described the theory of MERN.

Chapter 6 will describe How the author of the application implemented the concept of the MERN and created EventEve.

2. INTRODUCTION

This chapter of the work will describe how the web has grown in the last decade and how the future of web applications is. And what are the solutions available in the market instead of EventEve?

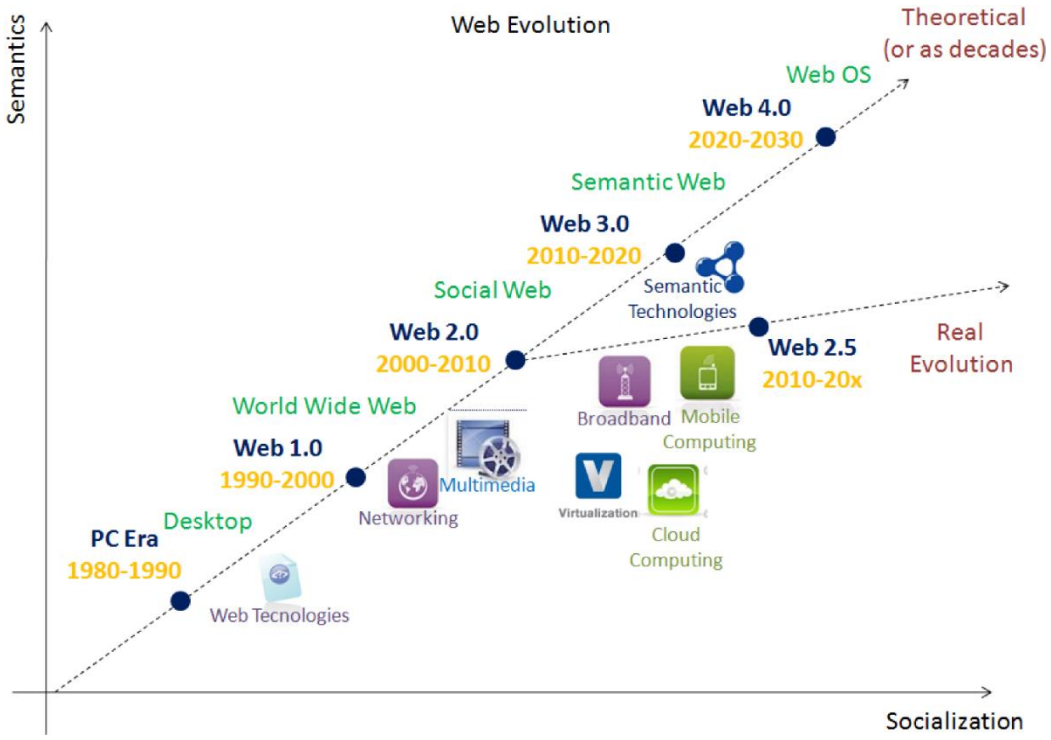
2.1 Growth of the web

The World-Wide-Web is a powerful tool of technology so every one of them exchanges data, reads data, publishes information, and engages with others via the internet. Since its inception, the World Wide Web has come a long way.

If we scan the web 1.0 that was developed to create static web-pages where the user had permission to read-only but now we are in phase 2 of the web where we can interact with people and web 3.0 journey is almost started so this version of web giving more business opportunities to people and providing effective and scalable solutions.

This is the justification for why new Web items, programming arrangements, web apparatuses, and advancements, just as focuses for online buys show up day by day. Here, individuals see their chance for an effective show. The fast advancement of the Web likewise definitely prompts the look for new skills and the rise of new kinds of work.

How future of the Web looks like



Source: mdpi.com: https://www.mdpi.com/futureinternet/futureinternet-04-00852/article_deploy/html/images/futureinternet-04-00852-g001.png

Figure 2.1 Web evolution

2.2 Available solutions

This chapter will discuss what are the solutions available in the market to solve a particular problem in the event management space, their advantages, and disadvantages, and how with this solution author can design functionalities for EventEve so it can be reached the users with a modern system design and functionalities.

2.2.1 Desktop applications

Small event companies are proving desktop solutions for their personal use case where the solution can be used by only one entity which can be that company's authorized person. The introduced solution is mostly used for the company's management so it is good for the business to use but not provide any type of service to clients.

2.2.2 Providing pass online

Existing solutions Book My Show, Web Tickets, and Tickets.com are the principal suppliers of this sort of arrangement. These are incredible organizations for purchasing a Web-based pass for an occasion however EventEve giving arrangements like interfacing with their accomplices without the expense of a penny is extraordinary for organizers but in other provided solutions that are not possible to do.

2.2.3 Static web-based apps

This sort of service is given by organizations like TRILLIUM EVENTS MANAGEMENTS where they are serving website page with what are their service, their accomplishments yet detriment of utilizing this sort of use while one needs to coordinate their occasions they need to contact by a call since they have static web-application so one cannot cooperate with applications.

3. ANALYSIS OF REQUIREMENTS

This chapter describes requirements that need to be fulfilled for the targeted result. So here author divided those requirements into two-part function and non-functional requirements.

3.1 Functional requirements

The functional requirements specify what the application is made to do or how the system must behave when certain conditions are met. So the following functionalities should be met in EventEve.

1. Create a new user: Registering an account is necessary to access further processes in the system.
2. Login, logout, and log out all users: Users can log in to the system to access data, logout(clear token from local storage) if they do not use the system for some time, log out all: if users want to logout from all device(that will delete all the tokens which are available in the database for that user)
3. Create a new partner: Registration of partner is introduced new partner in the system so user can have more options to choose a great one.
4. Login and logout partner: Partners can be log in and log out as per their needs.
5. Edit partner data, delete partner by admin: As it is free to register in the system so this part is necessary to introduce where admin really can take care of check documents provided by the partner company. This solution will keep applications away from scammers.
6. Create an event: The user can register an event to sell a pass on the system.
7. Edit event, delete the event by admin: Once the user will register an event it will push into the admin section where the admin checks everything about that user and event if the admin feels safe then they allow else they can delete it.
8. Allow communication between user and partner by messaging service :
   With this solution, it will be easy to connect with the application's partner and talk with them to know more about their services and pricing.
9. Allow payment methods: When a user buys a pass for an event they should be able to pay for that pass. To provide this solution stripe APIs played a great role.
10. Advertise events: If users want to advertise their event so that event can be shown on the landing page.
11. Buy event pass: Once users choose which pass to buy they will get to choose to pay and when payment will be successful users will get an e-pass via their email.

3.2 Non-functional requirements

Non-Functional requirements describe limitations on the created system. They determine the quality and characteristics of the product. Non-Functional requirements manage issues like adaptability, viability, execution, convey ability, security, dependability, and some more.

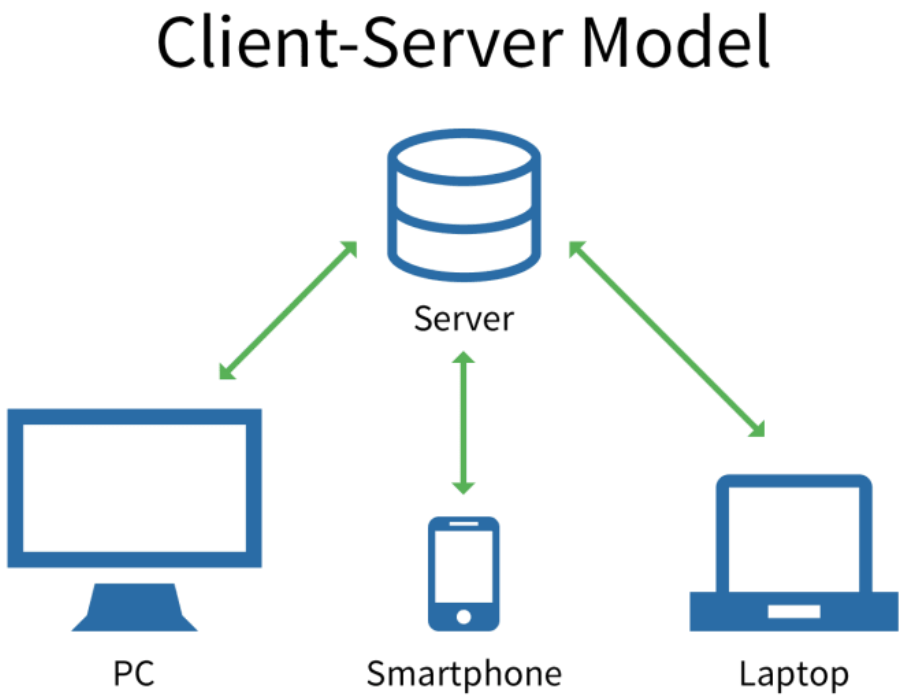
1. Handling every possible error: to handle, an error in code everywhere possible deal with a try and catch block so users can get an idea of why they are getting an error and what they are doing wrong.
2. Providing scalable and fast solutions: to provide efficient and fast solutions author used the modern async-await concept in code as well as wrote scalable code which can optimize time complexity and space complexity of code.
3. Protecting data: to protect data author use validation as well as use Bycrypt library for decrypting the password.
4. Providing a nice user interface: as of now, most components are taken from bootstrap or Reactstrap.
5. Availability 24/7

4. THE FOUNDATION OF THE WEB

This chapter is written for how one can develop a website and what are the things one needs to know to make fast, scalable, and nice-looking websites. So this chapter is going to be more theoretical to understand the process.

4.1 Web-apps

Web apps are a combination of frontend and backend, with a web browser such as Chrome acting as the client and a server acting as the backend. The logic of a web application can be divided into two parts: Frontend and backend, with the backend playing a critical role in data handling and maintenance. HTTP can be used to make all requests and responses (Hypertext transfer protocol). The most significant benefit of employing a web-based application is that users do not require any type of operating system to complete a task.



Source: https://res.cloudinary.com/practicaldev/image/fetch/s--PsOzNhBl--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https://cdn.techterms.com/img/lg/client-server_model_1253.png

Figure 4.1 Interaction between client and server

4.2 Frontend development

Frontend development is the method involved with making a pleasant user interface because the clients cannot have direct access to the backend so in the frontend one can validate what clients can see what clients can edit and all the CRUD activities. To develop the Front-end there are three primary columns named HTML, CSS, and JavaScript.

HTML (Hypertext markup language) is use for the structure of the content. By wrapping content by HTML tags like **<head>**, **<body>** , **<p>** , **<h1>** ,**** , **<table>** one can develop static web based application.

```
<p class="xyz">Hi THERE
<a href="http://www.somewhere.com ">click me</a>.</p>
```

Listing 4.1 Example a of HTML code

CSS (Cascading Style Sheets) is used to style the web application it can include colors, layout, and fonts. With the use of CSS, one can also think that how the website will look on different devices like smartphones, laptops, and tablets.

```
.xyz { font-style: italic }
```

Listing 4.2 Example of a CSS code

JavaScript is used to transfer static websites to dynamic websites. JavaScript files can contain all the logic for frontends like animation, error handling, API request, and data parsing. JavaScript can also run on the browser making it easy to use them.

4.3 Backend development

The other piece of the application is a server where all the business logic can execute, every one of the resources of the application can be there and at every single request, a server communicates with the database. One cannot straightforwardly give information to the frontend so a server has that access to interact with the database and according to request send a response to the frontend. A server can be built in different programming languages like python, Nodejs, Ruby, Go, and PHP. In the web 2 world that is not possible to website without a server or backend.

Backend is a combination of servers created by programming language and databases like SQL and NoSQL.

4.3.1 SQL (Structures Query Language)

SQL database is a relational database where all the data can be stored in form of a table, where one can execute a query to manipulate the database. The major operation on SQL can be

done by querying like retrieving data, inserting records, updating a record, deleting records, creating databases and tables, creating views, and setting permissions on tables and views.

Example of SQL database:

- MySQL
- MariaDB
- Oracle
- PostgreSQL
- MSSQL

4.3.2 NoSQL (Not only SQL)

NoSQL database is a document-oriented database where all the data can be stored in form of documents. In a NoSQL, database data is stored in BSON format so it is easy to access via index as a Javascript object which makes it easy to manipulate data for CRUD operations.

Some examples of NoSQL databases:

- MongoDB
- CouchDB
- CouchBase
- Casandra
- HBase
- Redis
- Riak
- Neo4J

5. THINKING IN MERN

The main focus point of this chapter is to discuss MongoDB, Express, React and Node.js. Following sub-chapter contains their deep understanding and workflow. By going further in this chapter, one can understand how MERN link together and this was first step for author to plane prototype.

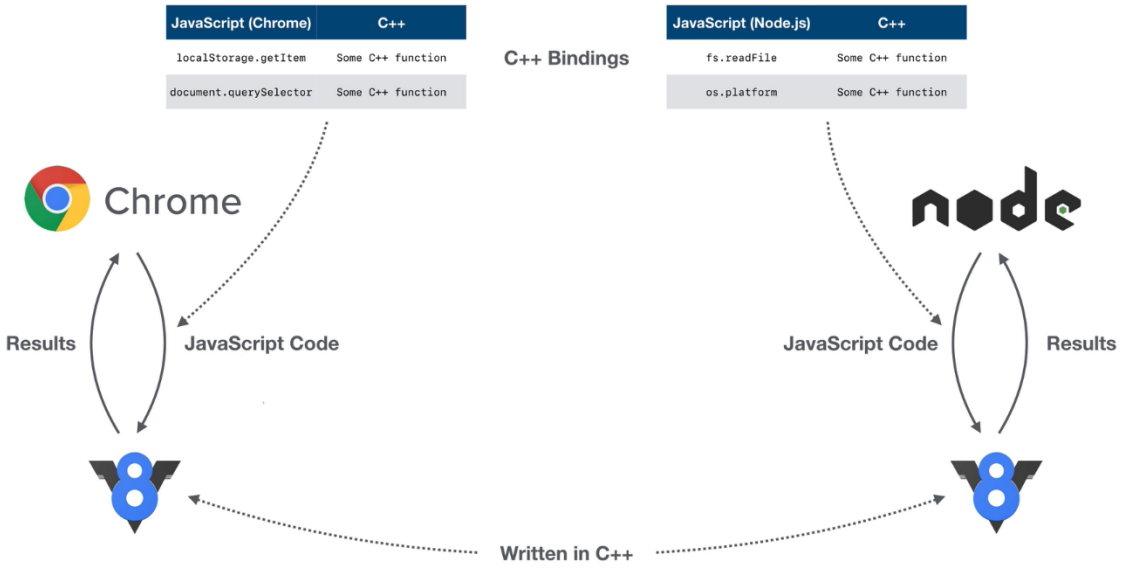
5.1 Server-side

To create a server one needs a database, programming language, and framework (to easily make a server). With the use of the backend framework, one needs to connect the server and database.

5.1.1 Node.js development

Before JavaScript was not be used as a more general-purpose programming language because it was limited to what the browser allowed it to do and there were no ways to use JavaScript outside of the browser to build things like web servers that access file server and connect to databases. But with the introduction of node.js, JavaScript developers could create web servers, command-line interface, and application backend [Nodejs 2021].

Its runtime environment is powered by a V8 engine (open source project by google), which is written in the C++ programming language.



Source: https://www.udemy.com/course/the-complete-nodejs-developer-course-2/learn/lecture/13728820#overview

Figure 5.1.1.1 The v8 JavaScript engine

```
const http = require('http'); // import http module
//creating server
const server = http.createServer((req, res) => {

});

server.listen(8000);

console.log(`server is running on port ${8000}`)
```

Listing 5.1.1.1 Creation of the Nodejs server

5.1.2 Express.js development

Writing all the server-side logic is complex with alone Nodejs. Express is a framework (Helper functions, tools, and rules that help to build applications) for Nodejs which helps Nodejs to easily create a server [Nodejs, Express, MongoDB 2021].

There are many Nodejs frameworks available such as Adonis.js, Koa.js, sails.js, and Express.js But among all of these Express.js is much popular.

The great thing about Express.js is it's highly flexible it gives a certain way of building applications and working with the incoming requests that make it highly extensible and there are thousands of third-party packages built for express specifically so one can integrate into the node-express application.

```
const express = require('express') // importing express
const app = express()
const port =  8000
//listening on port
app.listen(port, () => {
   console.log(`server is up to port :${port}`);
})
```

Listing 5.1.2.1 Creation of the Express.js server

5.1.3 MongoDB

MongoDB [MongoDB], the most famous NoSQL database, is an open-source record arranged DBMS. The term 'NoSQL' signifies non-relational. It implies that MongoDB did not depend on the table-like SQL DBMS but that stored data in documents (record key-value pair) called BSON (similar to JSON). That makes it easy to use and access data for the developers. That is one of the reasons for the popularity of MongoDB [Nodejs, express, MongoDB 2021].

```
{
  title: 'EventEve',
  by: 'HirenSavani',
  url: 'https://www.EventEve.com',
  type: 'NoSQL'
}
```

Listing 5.1.3.1 Storing data in MongoDB (BSON format)

As shown in above code, how data is stored in MongoDB, where in data is stored as a record inside the documents with key-value pairs called BSON format same as JavaScript or JSON, that makes it easy to access data by their keys.

❖ **Features of MongoDB**

- **Document oriented:** Stores data in the main subject in a minimal number of documents [MongoDB].
- **Indexing:** Searching data via the use of indexing
- **Scalability:** Sharing data with various servers
- **Replication and high availability:** Create multiple copies of data on various servers

❖ **Uses of MongoDB**

- **Big data:** When one needs to handle a huge amount of data then MongoDB can be helpful because it has a build-in solution partitioning and sharding database [MongoDB].
- **Adding new schema:** Adding a new field does not affect already existing documents.
- **Distributed data:** As mentioned in the above point it makes copies of data in every different server so it's easy to recover if there is a hardware failure.

❖ **Mongoose:** Mongoose is Object document mapper. With the use of mongoose, one can define objects with a strongly typed schema which mapped to MongoDB documents. Mongoose makes it easy to create and work with schemas which can contain eight schema types that are [Mongoose],

String	Number
Date	Buffer
Boolean	Mixed
ObjectId	Array

Table 5.1.3.1 Data type can be stored with mongoose

```
const mongoose = require('mongoose')
//creating event schema
const EventPassesSchema = mongoose.Schema({
    event: {
        type: mongoose.Schema.Types.ObjectId,
        require: true
    },
    user: {
        type: mongoose.Schema.Types.ObjectId,
        require: true
    },
    NumberOfPass: {
        type: Number
    },
    Charged: {
        type: Number
    }
}, {
    timestamps: true
})
const Eventpasses = mongoose.model('eventpasses', EventPassesSchema)
module.exports = Eventpasses
```

Listing 5.1.3.2 Creation of "Eventpasses" schema

5.2 Clientside – React.js

To develop a client-side application author will use React.js, which is the most popular and more efficient solution among others.

React is a revelatory, proficient, and adaptable JavaScript library for building UIs. React is an open-source component-based frontend library maintained by Facebook.

For HTML and XML documents DOM (Document object model) manipulation is very slow and expensive. So improve that React implements a virtual DOM that is DOM tree representation in JavaScript. So whenever it needs to update DOM it will manipulate virtual DOM then virtual DOM finds an efficient solution to update real DOM [React 2021].

Not at all like browser DOM components, React components are plain objects and are modest to make. React DOM deals with refreshing the DOM to match the React components. The justification behind this is that JavaScript is exceptionally quick and it merits keeping a DOM tree in it to accelerate its control.

The component is a major part of reacting Framework. It accepts props and returns React elements that tell how UI should frame.

There are two possible ways to create React components

- Functional components
- Class-based components

5.2.1 Functional components

- Functional Components look like normal JavaScript Function [class-components-vs-functional-components 2020].
- It can be created with arrow and regular function keywords
- Use to rendering UI
- Not use of render method
- React life cycle (ComponentDidMount) cannot be used
- Below given example represent Functional based components

```
Import React from 'react'


Function EventEve(props) {
  return <h1>Hello, {props.eventName}</h1>;
}
export default EventEve
```

Listing 5.2.1.1 Functional based components

5.2.2 Class components

- With the use of ES6 class and extend keyword one can create a class-based component
- Allow implementing React lifecycle method

- Access props by this.props

```
Import React ,{Components} from ‘react’
class EventEve extends React.Component {
  render() {
    return <h1>Hello, {this.props.EventName}</h1>;
  }
}
Export default EventEve
```

Listing 5.2.2.1 React class-based components

5.3 Additional

This subchapter is going to be a brief description of tools, libraries, and environments to achieve the aimed website.

5.3.1 Npm (Node Package Manager)

It is a node pack manager which is a provider of a hundred thousand packages or a library without cost a penny. To install a package in the project via npm one just need to write “npm install” [package name]. These packages are open source projects created by someone to let developers use for free.

5.3.2 Programming environment

- **Visual studio code**

The creator of the application to carry out it utilized the Visual studio code programming environment. It incorporates the code manager and SDK apparatuses and empowers code troubleshooting [Visual studio code].

- **Robo3T**

Robo3T is an open-source and lightweight user interface tool for managing MongoDB workloads [Robo3T]. Which can connect to local applications and store data inside so if developers need to improve anything they can do without any permission.

- **Libraries**

Libraries are open source package which makes things easy for the developer. With the use of a library developer does not create that functionality on their own they can implement the functionality in the project by a piece of code.

Frontend libraries	Backend libraries
Axios	Body-parser
Bootstrap	cors
react-feather	dotenv
react-router-dom	Easy invoice
react-step-wizard	express
react-stripe-checkout	jsonwebtoken
react-toasty	multi
socket. io-client	nodemon
timeago.js	QRcode
	stripe
	UUID
	validator

6. IMPLEMENTATION

This chapter is deep dive into the implementation of the project in the MERN stack how the application's pages look like, how logic is used to create an application, and what the will be outcome. So in the first subchapter author will show graphical representation and in the second subchapter, there will be a description of the code how it looks like?

6.1 Graphical representation

This part of the chapter will be discussed the UI of the EventEve. The point of the writer during their creation is essentially comprehensibility and straightforwardness.

6.1.1 Landing page

A landing page is the first impression of any application so it is followed up to any promises that the application made in content. It is the next step toward a visitor becoming a customer. Figure 6.1.1.1 is the application's landing page once users will open EventEve they will automatically redirect to a landing with a nice user interface where they will get options for a sign, signup, and Be-partner. As users scroll down they will be able to see some events near them and about EventEve and what we are providing.

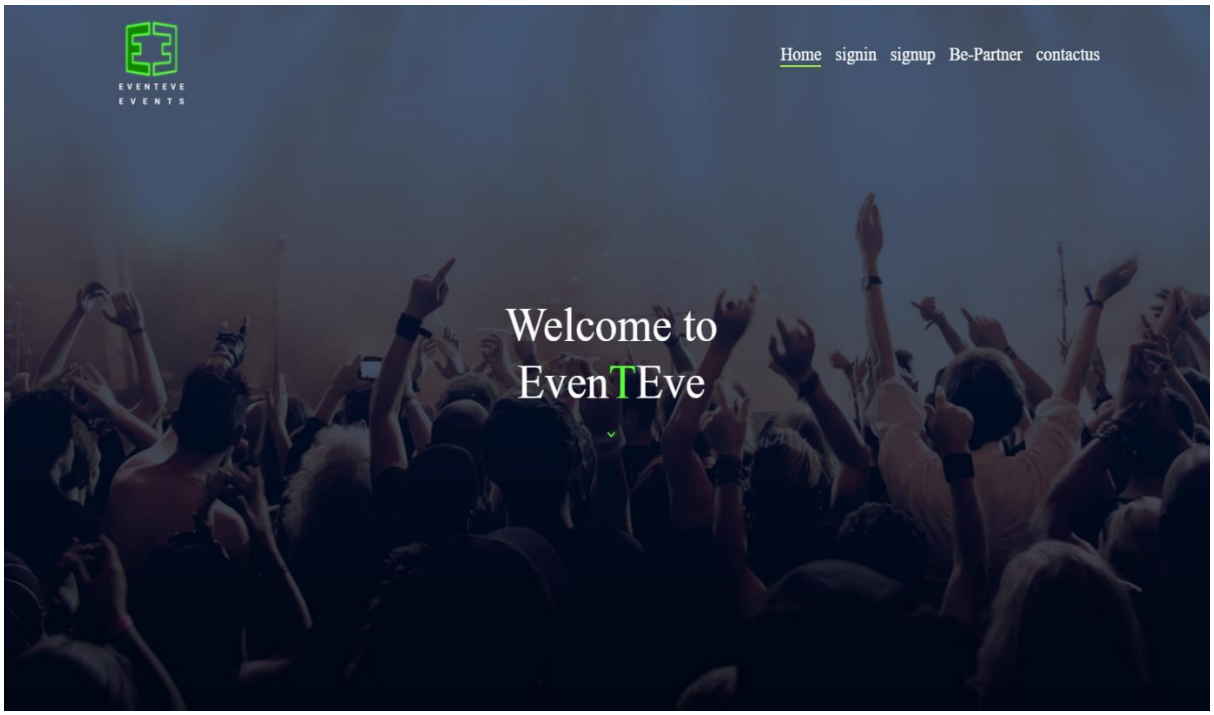


Figure 6.1.1.1 Application landing page

6.1.2 Homepage

The main goal of the home page delivers the content to the user or deliver them closer to the page that has content. So it is the area that people notice at the first glance and decide their basic judgments about applications. As shown in figure 6.1.2.1 home page is having 3 different options where users can find an event near them, register an event so they can sell the pass on the system and the last option is to let us manage your event where users can directly connect to partners.

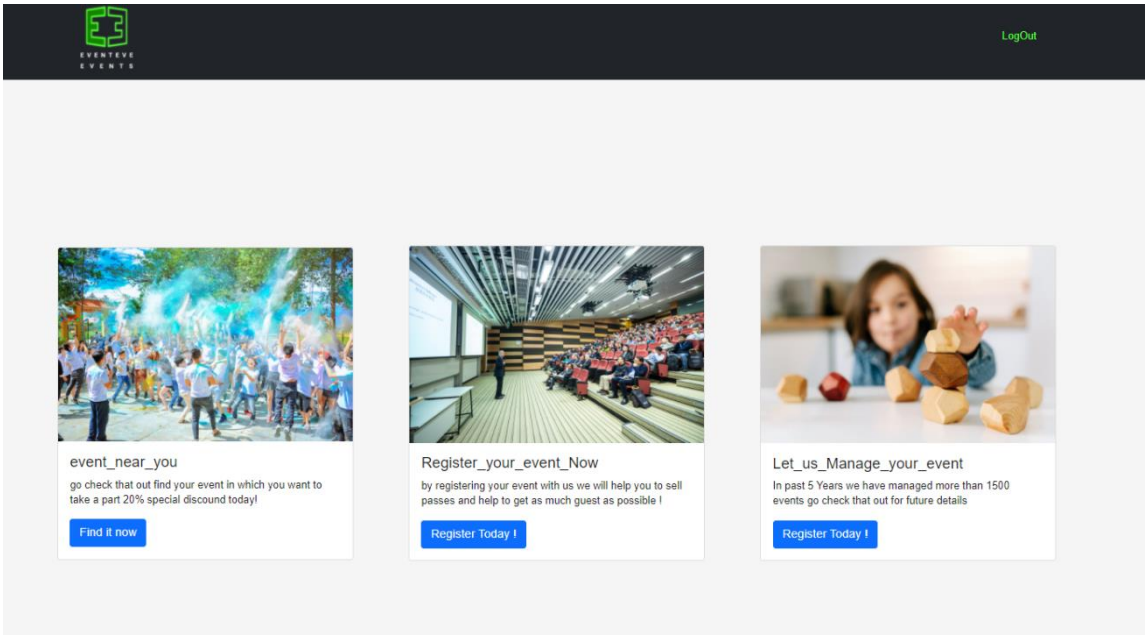


Figure 6.1.2.1 Application page home page

6.1.3 Event near you

The main goal of this page is to attract people to buy a pass so as shown in figure 6.1.3.1 “event near you” page contains a list of events that are registered before the date and pass available. Where users can see events in more detail and choose whether to or not. If a user wants to buy then the author has already integrated Stripe API to make payment successful.

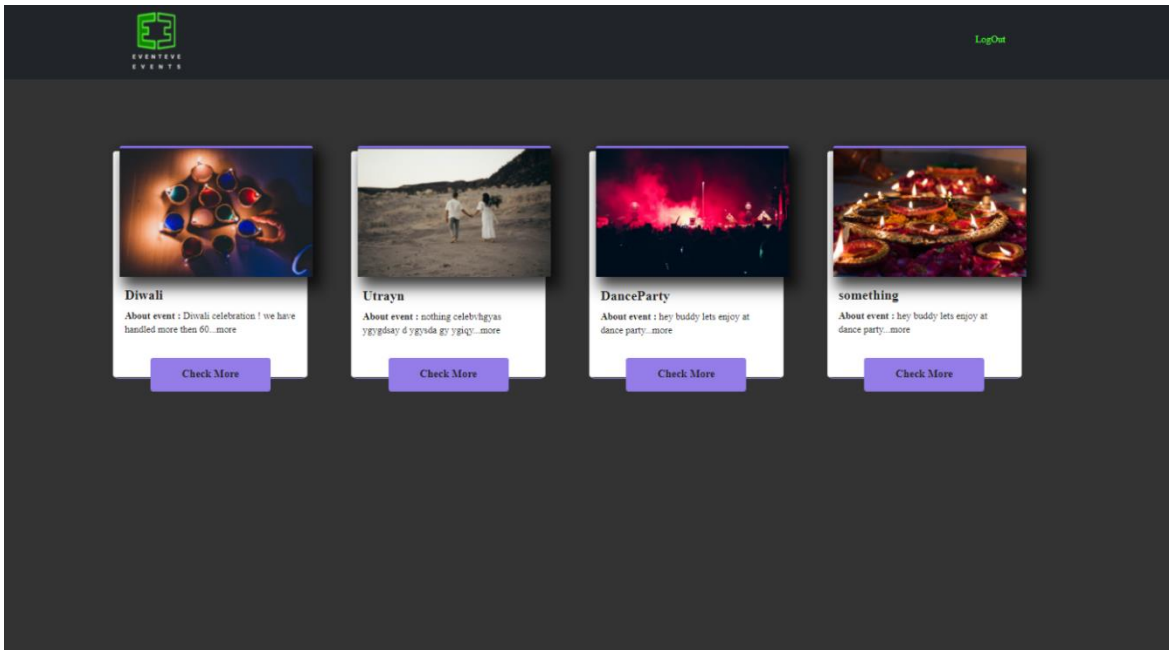


Figure 6.1.3.1 Application page "event near you"

Once customers will get attracted by the event and they will check for more details application will do its work to show some nice pictures that the organizer has uploaded. Once the user clicks on pay then this payment gateway will pop up. As the author set Stripe API in test mode only some limited cards will be valid.

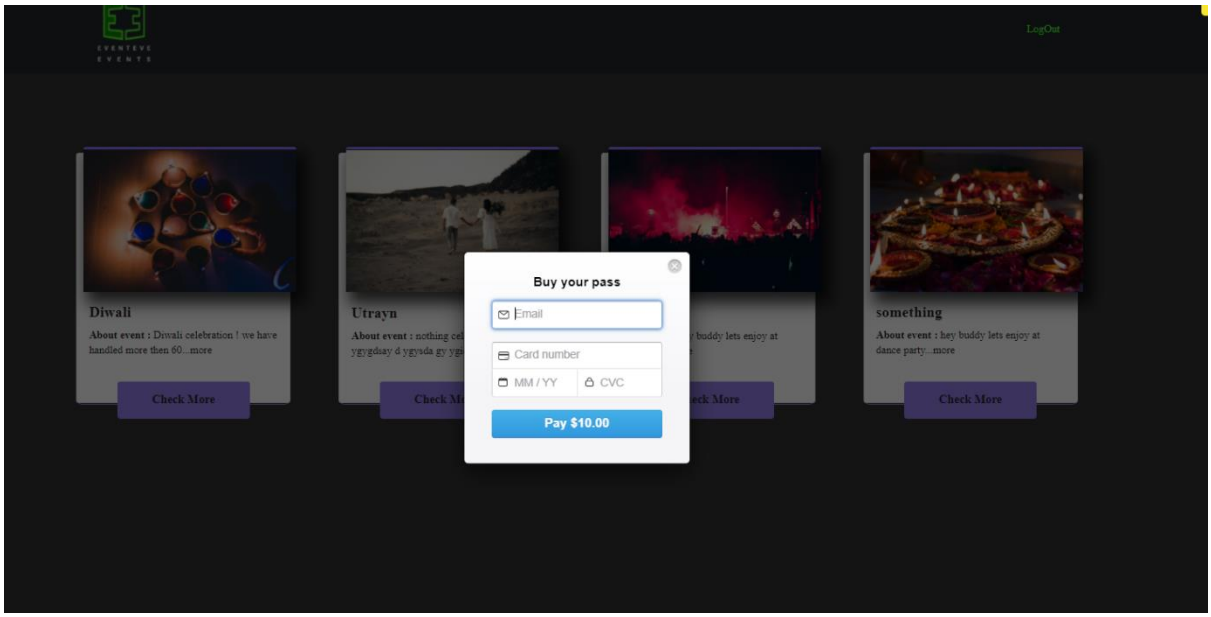


Figure 6.1.3.2 Application page "event near you" payment

6.1.4 Register your event

The register event page is for the user who organizing events and wants to sell event passes on the system. To register event user needs to provide the below data.

Even type	Picture for event
Event name	Organizer documents for verification
Address	Advertisement
Description about event	
Number of passes	

Table 6.1.4.1 Must provide data to register

Once the user will provide the above data and submit the application admin of the application will receive the application. Once the admin will review the application then the user will be able to see that in the system.

6.1.5 Let us manage

Users can find a partner via visiting this page where users can sort data as several guests, base price, and city so they can get available partners accordingly.

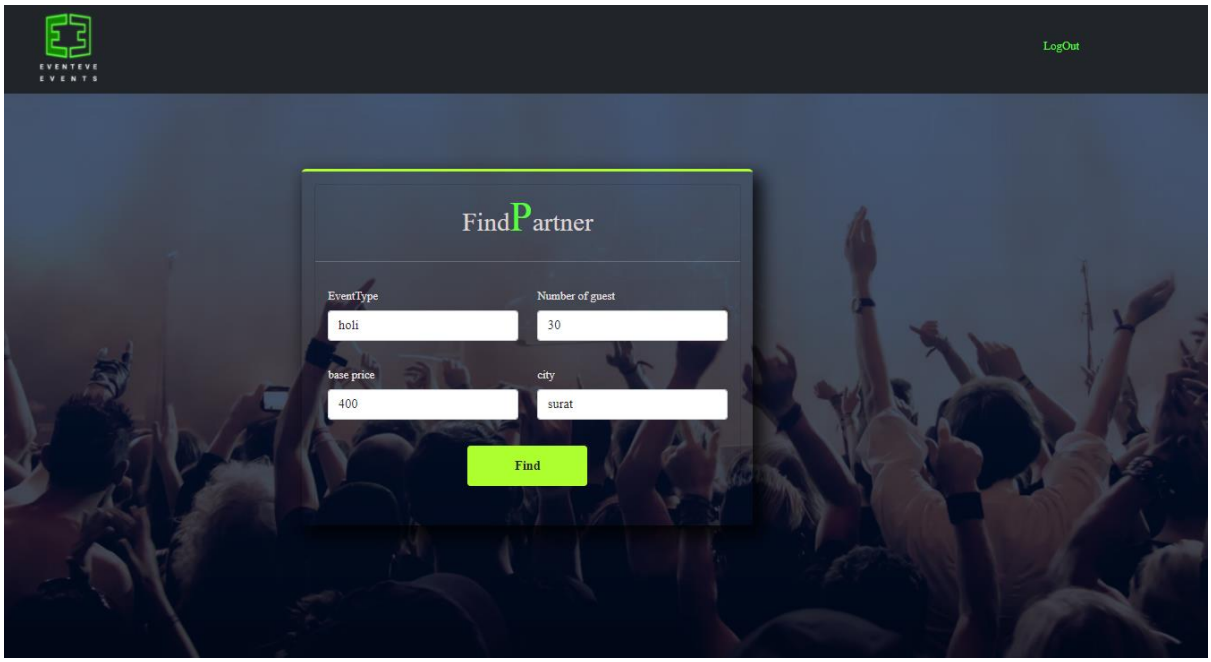


Figure 6.1.5.1 Search organizer

As shown in figure 6.1.5.2 by searching organizer users will redirect to the page where users can find our partner and can know more about them to work with them.

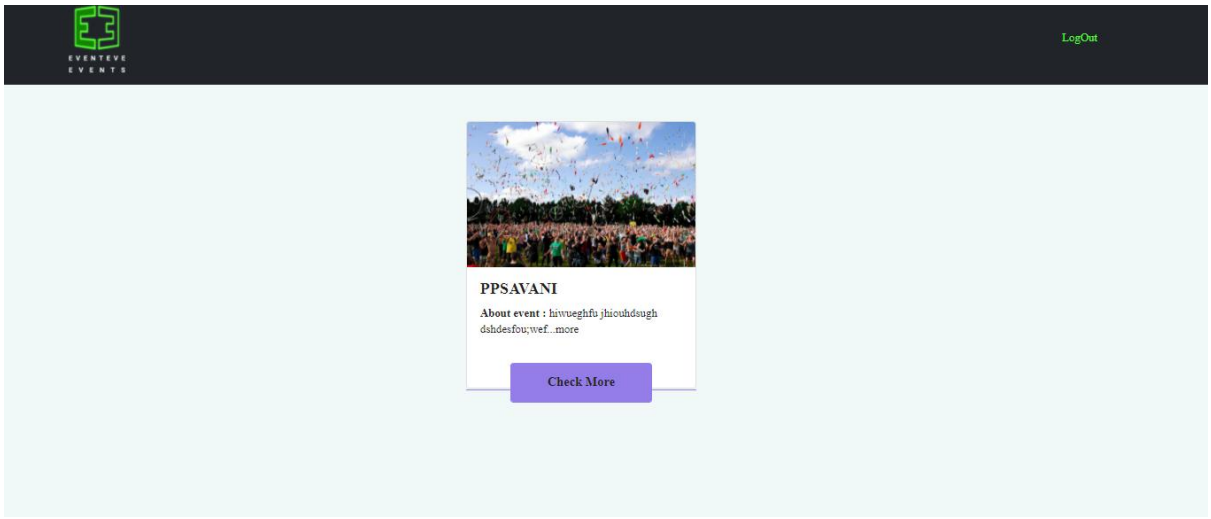


Figure 6.1.5.2 Found the organizer

Once users found the partner they will redirect to messenger where users can able to talk with them and know more about charges and other services.

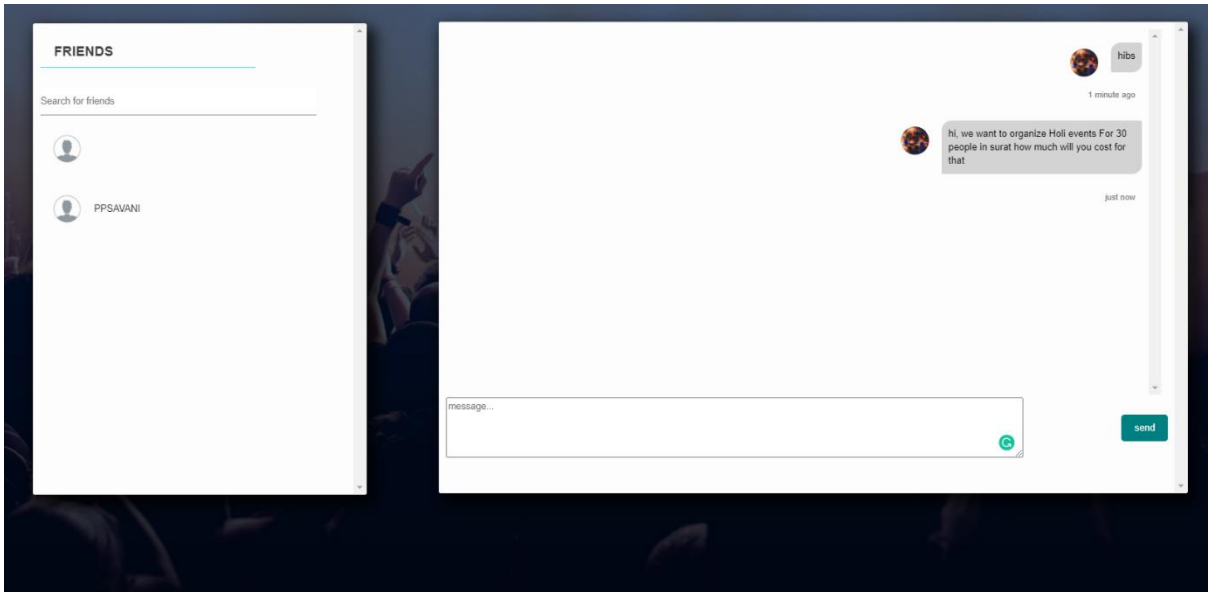


Figure 6.1.5.3 Messenger page

6.2 Implementation of code

By this part of the chapter, the author wants to show chunks of code that were used for building the prototype.

6.2.1 Frontend development

This chapter illustrate front-end development, which focuses on React.js and other frontend helper libraries. This chapter also contains code from some important frontend file. Below picture represent React file structure.

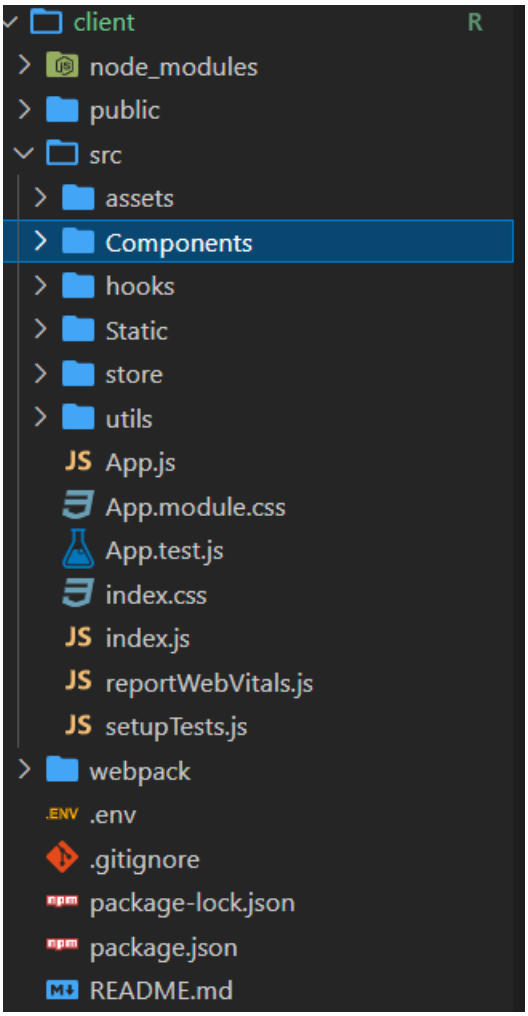


Figure 6.2.1.1 How does the application's frontend files structure looks like

- **Entry point**

Index.js file is the entry point for react applications that handle app start up, all the routing in the application, and other functions of the application.

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter } from 'react-router-dom';
import { AuthContextProvider } from './store/auth-context'

import App from './App';

ReactDOM.render(

  <React.StrictMode>

    <BrowserRouter>
      <AuthContextProvider>
        <App />
      </AuthContextProvider>
    </BrowserRouter>

  </React.StrictMode>
  ,
  document.getElementById('root')
);
```

Listing 6.2.1.1 Frontend index.js file code

- **Sending a post request to the server**

In a Listing 6.2.1.2 it is shown how sending of a post API requests to the backend server is being handled.Example of code is available in file component/letusmanage/Form/form.js from Line Number 35 to 56.

```
const body = {
        city: enteredCity,
        guest: enteredNumberOfGuest,
        basePrice: enteredBasePrice
    }
    try {    //with async function only
        const response = await axios.post('http://localhost:8000/findrelated', body, { headers: {
"Authorization": `Bearer ${token}` } })
        if (response.status === 200) {
            toast.success(`${response.status} : request successFull wait a min we will serve you`)
            props.onSearchHandler(response.data)
        }
        if (response.status === 204) {
            toast.warn(`${response.status} : oops! no match found`);
        }
    } catch (e) {
        toast.error("connection losed! ");
    }
```

Listing 6.2.1.2 Frontend handling post request

6.2.2 Backend development

This chapter illustrate backend development, which spread light on Nodejs and other backend helper libraries. This chapter also contains code from some important backend file. Below picture represent backend file structure.

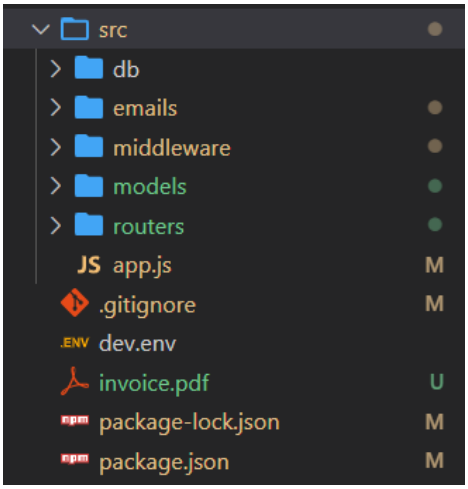


Figure 6.2.2.1 How does the application's backend files structure looks like?

❖ Entry Point

The main.js file is the entry point for node applications that handle app startup, all the routing in the application, and other functions of the application.

```
const express = require('express')
require('./db/mongoose')

const UsersRouter = require('./routers/Users')
const EventsRouter = require('./routers/Events')
const EventPasses = require('./routers/EventPasses')
const Partner = require('./routers/Partner')
const Conversation = require('./routers/conversation')
const Message = require('./routers/message')
const bodyParser = require('body-parser')
const cors = require('cors')

const app = express()

app.use(cors())

const port = process.env.PORT || 8000

app.use(bodyParser.json())
app.use(UsersRouter)
app.use(EventsRouter)
app.use(EventPasses)
app.use(Partner)
app.use(Conversation)
app.use(Message)
app.listen(port, () => {
   console.log(`server is up to port :${port}`);
})
```

Listing 6.2.2.1 Backend main.js file code

❖ **Schema**

This sub-chapter contains some of the created Schema from the prototype's code. The Schema is validated with datatypes and created with Mongoose library.

```
const mongoose = require('mongoose')
const validator = require('validator')

const ConversationSchema = mongoose.Schema({
   members: {
      type: Array
   }
}, { timestamps: true })

const Conversation = mongoose.model('Conversation', ConversationSchema)

module.exports = Conversation;
```

Listing 6.2.2.2 Backend main.js file code

❖ **Router**

Router can be usefull for performing some task by visiting particular route. Router contains all the business logic inside it. As shown in code given below, it is wrapped by try and catch block for error handling. One can create "get", "post", "patch", "delete" and "put" types of route.

```
//getting allevents by seen
router.get('/getallEvents', auth, async (req, res) => {
    try {
        let event;
        if (req.user.isadmin) {
            event = await Events.find({ seen: false })
        } else {
            event = await Events.find({ seen: true })
        }



        res.status(200).send(event)
    } catch (e) {
        res.status(400).send(e)
    }
})
```

Listing 6.2.2.3 Backend route

7. TESTS AND PLANS

The author tested code with every possible input and every possible loophole. The code works fine in the applications, but the author did not use modern testing frameworks like Mocha or Jest. To improve the solution and check every request and response author will use this type of framework.

The other way to test an application is to let the users use the application and then they will review it. So the author kept the application on git hub which is easy to clone and download.

The author is planning to improve the user interface, provide more functionalities like third party login, organize the events by EventEve itself, recruit people like Glovo and Pyznepl and 3D canvas as well as generate more scalable solutions to improve user experience.

Another plan is to recreate this application for web3 users and create a mobile version of the application too.For both android and IOS to reach more customers.

8. CONCLUSION

The main focus of this work is to study real-world event management problems and improve them with more modern functionalities, user experience, and connect them with people to make successful events.

- The application is creating B2B and B2C business so doing this work can be helpful in the real world to serving people.
- As the author has given functionality to register an event and advertise them that will be very helpful for the micro event and mini-event to achieve numbers.
- By providing functionalities like “let us manage” it can be easy for the users to reach the organizer and talk with them.
- As the application is created in full-stack JavaScript author got opportunities to deal with implementing different functionalities like making successful payments, generating pdf passes, generating QR code, and using web sockets.
- As part of future work, the author is looking to improve functionalities that will improve the interest of the users and help to reach more users.

BIBLIOGRAPHY

[React 2021] React - react-the-complete-guide, [Access via:]
https://www.udemy.com/course/react-the-complete-guide-incl-redux/learn/lecture
[Access:2021-08-19]

[class-components-vs-functional-components 2020], [Access via]
https://www.linkedin.com/pulse/class-components-vs-functional-components-react-dinuka-fernando/
[Access: 2021-9-13]

[Nodejs 2021] The Complete Node.js Developer Course, [Access via:]
https://www.udemy.com/course/the-complete-nodejs-developer-course-2/
[Access: 2021-09-30]

[Nodejs,express,mongodb 2021] NodeJS - The Complete Guide, [Access Via]
https://www.udemy.com/course/nodejs-the-complete-guide/
[Access: 2021-10-13]

[MongoDB]: mongodb-an-introduction, [Access via]
https://www.geeksforgeeks.org/mongodb-an-introduction/
[Access: 2021-10-10]

[Mongoose]: an-introduction-to-mongoose, [Access via]
https://code.tutsplus.com/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527
[Access: 2021-10-10]

[Robo3T]: robomongo,[Access via]
https://robomongo.org/
[Access:2021-09-10]

[Visual studio code]: visual studio code, [Access via]
https://code.visualstudio.com/docs
[Access:2021-12-28]