EVENT MANAGEMENT SYSTEM

E3 MINI PROJECT REPORT

Submitted By

M Sreenivasulu – R161098
T S Balagangadhar Thilak – R161431

In partial fulfilment of the project requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

RK Valley Campus
Under the Guidance of
Mr. N. Satyanandaram,
Lecturer,Dept.Of CSE,
RGUKT, RK Valley.

Acknowledgement

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people, who made it possible and whose constant guidance and encouragement crowns all efforts with success.

Firstly, we would humbly thank Mr. N Satyanandaram, Lecturer, CSE, for being the channel and guide for us to have availed such a rewarding opportunity.

ABSTRACT

Event Management is a web based application to register the marriage events. Here the users can see the privious events done by the team. Users can also give feedback to the management team.

In this web application users can register an marriage event through email address. The management team will communicate with the client through email.

After seeing the previous events user can fill the details of the Bride and Groom. After registration for an event user can login the application to see the status of the event.

After completion of an event management team will give marriage memories to the user registered mail address.

Here for frontend we are used Reactjs , HTML , CSS .

For backend we are used firebase

TABLE OF CONTENTS

1.Introduction

1.1 HTML

- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

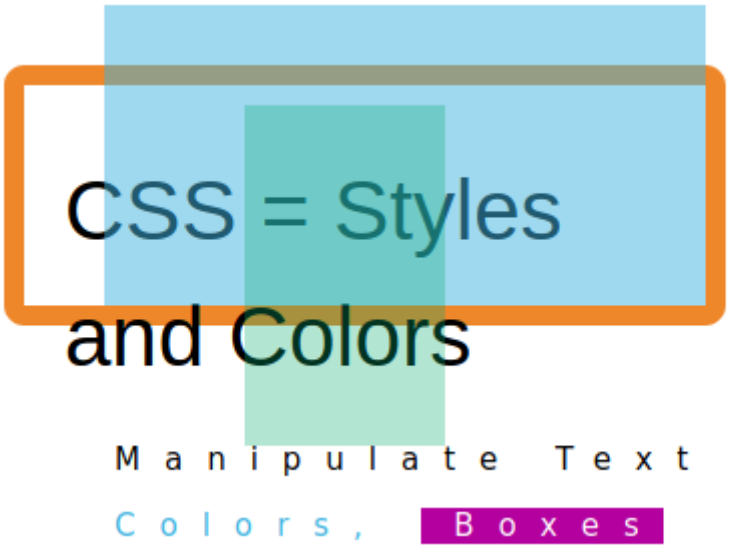
CSS

CSS stands for Cascading Style Sheets.

CSS saves a lot of work. It can control the layout of multiple web pages all at once.

Cascading Style Sheets (CSS) is used to format the layout of a webpage.

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!



1.2 About the Project

“EVENT MANAGEMENT “ is a web Application.Users can register their events in these Application.Users can also view the previous events. Manager will manage the registered events.Users & Manager can see the
status of the event.After the event completion manager will provide the event memories to the users ,to their registered mail Id.

Technologies

HTML , CSS

React JS

Firebase

1.3 Objective

The main objective is to create a user web application to register the marriages based on the ideas of the management team.

Based on the user thoughts team will manage the marriage.

2. System Configuration

2.1 Hardware Requirements

Processor : Duel core 1.6 Ghz

HDD : 250 GB

RAM : 4 GB

Mouse : Optical Mouse

2.2 Software Requirements

Operating System : Windows 7

Coding Language : JAVA

Front End : React js

IDE Tools : Tomcat

Reactjs

React

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called “components”.

React has a few different kinds of components, but we’ll start with `React.Component` subclasses .

We’ll get to the funny XML-like tags soon. We use components to tell React what we want to see on the screen. When our data changes, React will efficiently update and re-render our components.

The `render` method returns a *description* of what you want to see on the screen. React takes the description and displays the result. In particular, `render` returns a **React element**, which is a lightweight description of what to render. Most React developers use a special syntax called “JSX” which makes these structures easier to write. The `<div />` syntax is transformed at build time to `React.createElement('div')`

```
JSX comes with the full power of JavaScript.
You can put any JavaScript expressions
within braces inside JSX. Each React element
is a JavaScript object that you can store in
a variable or pass around in your program.
```

A Simple Component

React components implement a `render()` method that takes input data and returns what to display. This example uses an XML-like syntax called JSX. Input data that is passed into the component can be accessed by `render()` via `this.props`.

JSX is optional and not required to use React. Try the Babel REPL to see the raw JavaScript code produced by the JSX compilation step.

A Stateful Component

In addition to taking input data (accessed via `this.props`), a component can maintain internal state data (accessed via `this.state`). When a component's state data changes, the rendered markup will be updated by re-invoking `render()`.

A Component Using External Plugins

React allows you to interface with other libraries and frameworks. This example uses remarkable, an external Markdown library, to convert the `<textarea>`'s value in real time.

Firebase

Creating the Firebase Realtime Database is essential for most apps.

Suppose if you’re creating an app that requires storing and sharing data from a server, you have to memorize a lot of things like creating and maintaining a database is a tough job, isn’t it?

And in case you need your data to be synchronized in realtime, or you need the offline support functionality, then that might be taking a lot of time. Yes, you can save your time by using Firebase Realtime Database.

A real-time database allows you to store the data and sync automatically among the users in realtime. It lets you or the users cooperate. This makes it simpler for users to obtain their own data from any device (mobile, web).

Whenever you update or upload any data in the Firebase servers, it will automatically update to the user’s device in just a few milliseconds. Isn’t it amazing?

Also, it gives you a facility for offline support. For example, if a **person loses their data connection**, the Realtime Database SDK uses device local cache to serve the users, and whenever the user is online again, then **data is automatically synchronized.**

One important thing that will come in the developer mind is that Firebase offers two types of Database

1. Realtime database
2. Firestore database

The Firebase Realtime Database is **a cloud-hosted NoSQL database** that lets you store and sync data between your users in realtime. NEW: Cloud Firestore enables you to store, sync and query app data at global scale.

Firestore is **a NoSQL document database built for automatic scaling, high performance, and ease of application development**. While the Firestore interface has many of the same features as traditional databases, as a NoSQL database it differs from them in the way it describes relationships between data objects.

3. System Design

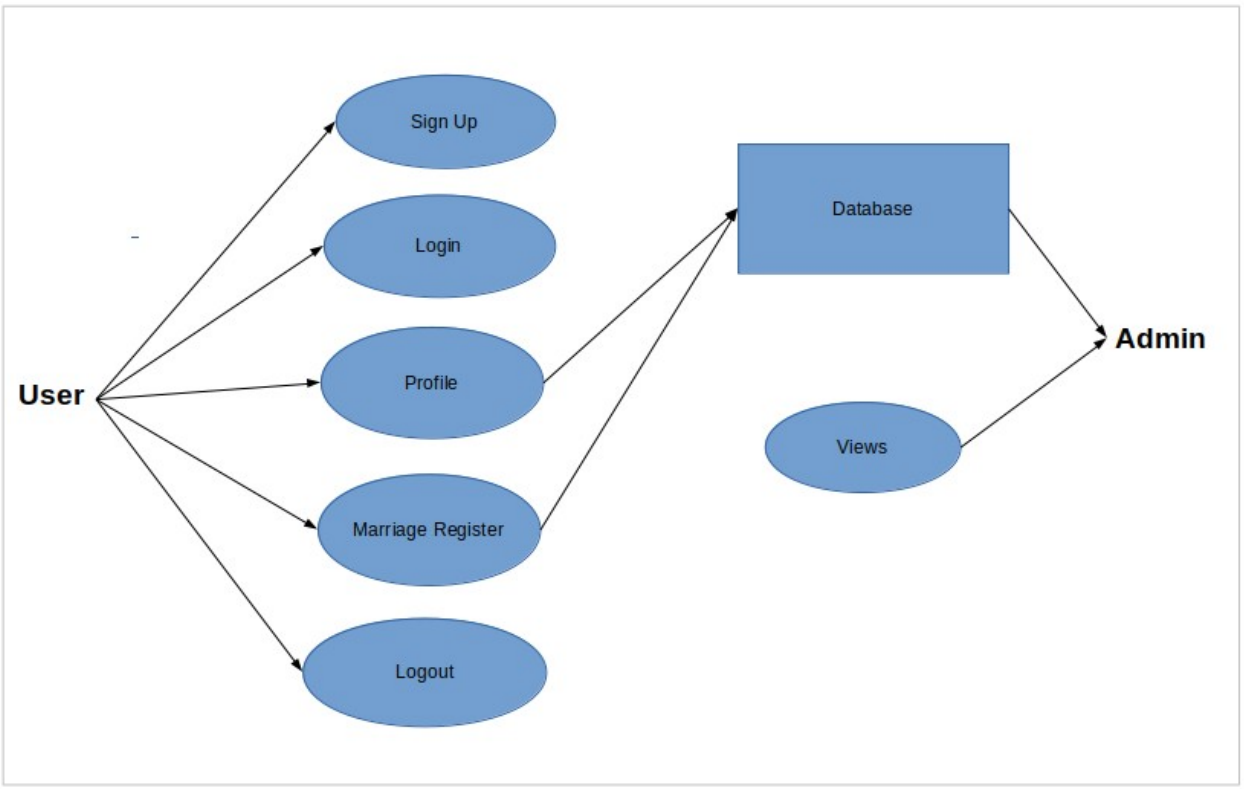
3.1 Introduction

Use case Diagram

A use case is a written description of how users will perform tasks on your website. It outlines, from a user's point of view, a system's behavior as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal and ending when that goal is fulfilled.

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

Use case Diagram

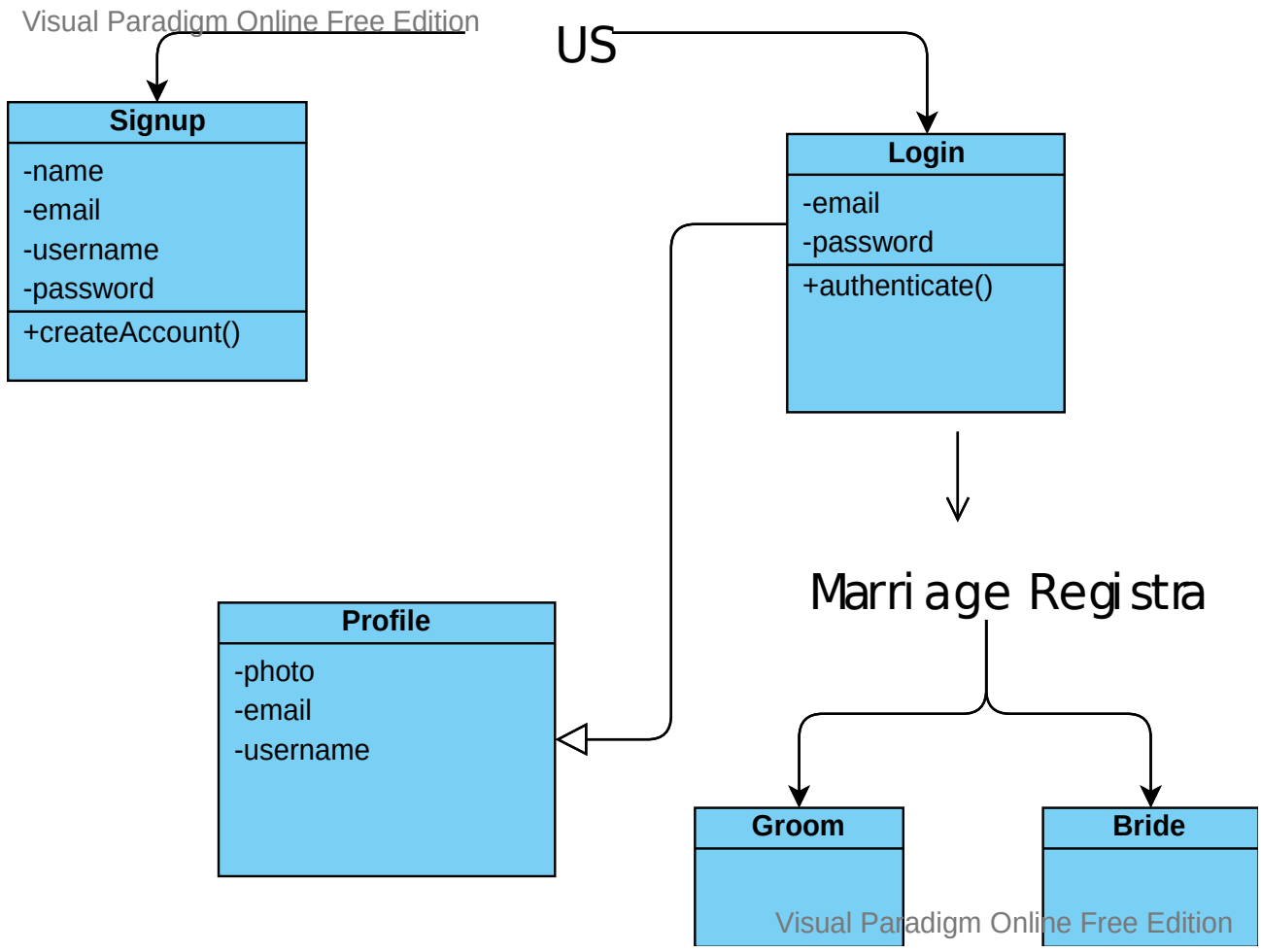


3.2 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.

Class Diagram



Details of the Bride & Groom

5. Testing

The purpose of testing is to find errors. Testing is that the method of making an attempt

to find each conceivable fault or weakness during a work product. It provides the simplest way

to examine the practicality of parts, sub assemblies, assemblies ANd/or a finished product it's

the method of workout code with the intent of making certain that the software package meets

its needs and user expectations and doesn't fail in an unacceptable manner. There ar varied sorts

of check. every check kind addresses a particular testing demand.

TYPES OF TESTS

1. Unit testing

Unit checking involves the look of test cases that validate that the inner program logic is

functioning properly, which program inputs turn out valid outputs. All call branches and internal

code flow ought to be valid. it's the testing of individual computer code units of the

applying .it is done once the completion of a personal unit before integration. this is often a

structural testing, that depends on information of its construction and is invasive. Unit checks perform basic tests at part level and test a selected business method, application, and/or system configuration. Unit tests make sure that every distinctive path of a business method performs

accurately to the documented specifications and contains clearly outlined inputs and expected

results.

2. Integration testing

Integration tests area unit designed to check integrated software system elements to work out if they really run jointly program. Testing is event driven and is additional involved with the fundamental outcome of screens or fields. Integration tests demonstrate that though the elements were on an individual basis satisfaction,

as shown by with success unit testing, the mix of elements is correct and consistent. Integration testing is specifically geared toward exposing the issues that arise from the mix of elements.

Functional test

Functional tests offer systematic demonstrations that functions tested square measure out there as such by the business and technical necessities, system documentation, and user manuals.

Functional testing is cantered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and

preparation of useful tests is targeted on needs, key functions, or special check cases. Additionally, systematic coverage concerning establish Business method flows; knowledge fields, predefined processes, and ordered processes should be thought of for

testing. Before useful testing is complete, further tests area unit known and therefore the effective worth of current tests is decided.

3. System Test

System testing ensures that the whole integrated computer code meets necessities. It tests a configuration to make sure known and certain results. AN example of system take a look ating is that the configuration directed system integration test. System testing is predicated on method descriptions and flows, accentuation pre-driven method links and integration points.

4. White Box Testing

White Box Testing may be a testing during which during which the computer code tester has information of the inner workings, structure and

language of the computer code, or a minimum of its purpose. it's purpose. it's wont to check areas that can't be reached from a recorder level.

5. Black Box Testing

Black Box Testing is testing the software package with none information of the inner

workings, structure or language of the module being tested. recording equipment tests, as most

different kinds of tests, should be written from a definitive supply document, like specification or necessities document, like specification or necessities document. it's a take a look rating during which the software package beneath test is treated, as a recording equipment.

You cannot “see” into it. The take a look at provides inputs and responds to outputs while not considering however the software package works.

TEST PROCEDURE

The following are the Testing Methodologies:

o Unit Testing.

o Integration Testing.

o User Acceptance Testing.o Output Testing.

o Validation Testing.

Unit Testing

Unit testing focuses verification effort on the littlest unit of software package style that's the module. Unit testing exercises specific methods during a module’s management structure to make sure complete coverage and most error detection. This take a look at focuses on every

module separately, guaranteeing that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, every module is tested separately and also the module interfaces square measure verified for the consistency with style specification. All necessary process path square measure tested for the expected results. All error handling methods are tested

Integration Testing

Integration testing addresses the problems related to the twin problems of verification and program construction. Once the code has been integrated a collection of high order tests

area unit conducted. The main objective during this testing method is to require unit tested modules and builds a program structure that has been set advisedly. The following are the types of Integration Testing:

1) Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2) Bottom-up Integration

This technique begins the development and testing with the modules at rock bottom level within the program structure. Since the modules area unit integrated from all-time low up, process needed for modules subordinate to a given level is usually on the market and also the want for stubs is eliminated. all-time low up integration strategy is also enforced with the

subsequent steps:

The low-level modules area unit combined into clusters into clusters that perform a selected software package sub-function.

A driver (i.e.) the management program for testing is written to coordinate action input

and output.

The cluster is tested.

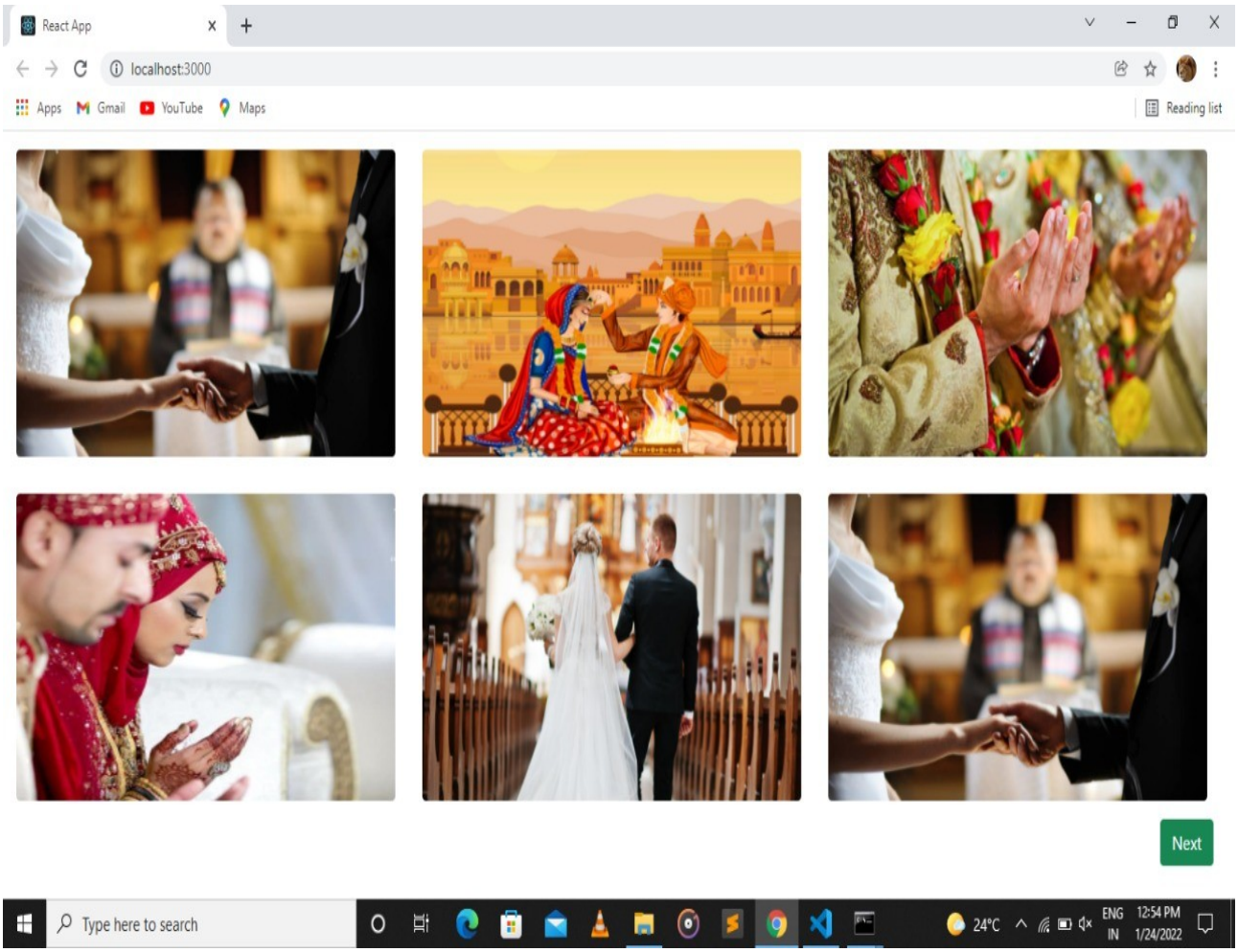
Drivers area unit removed and clusters area unit combined moving upward within the

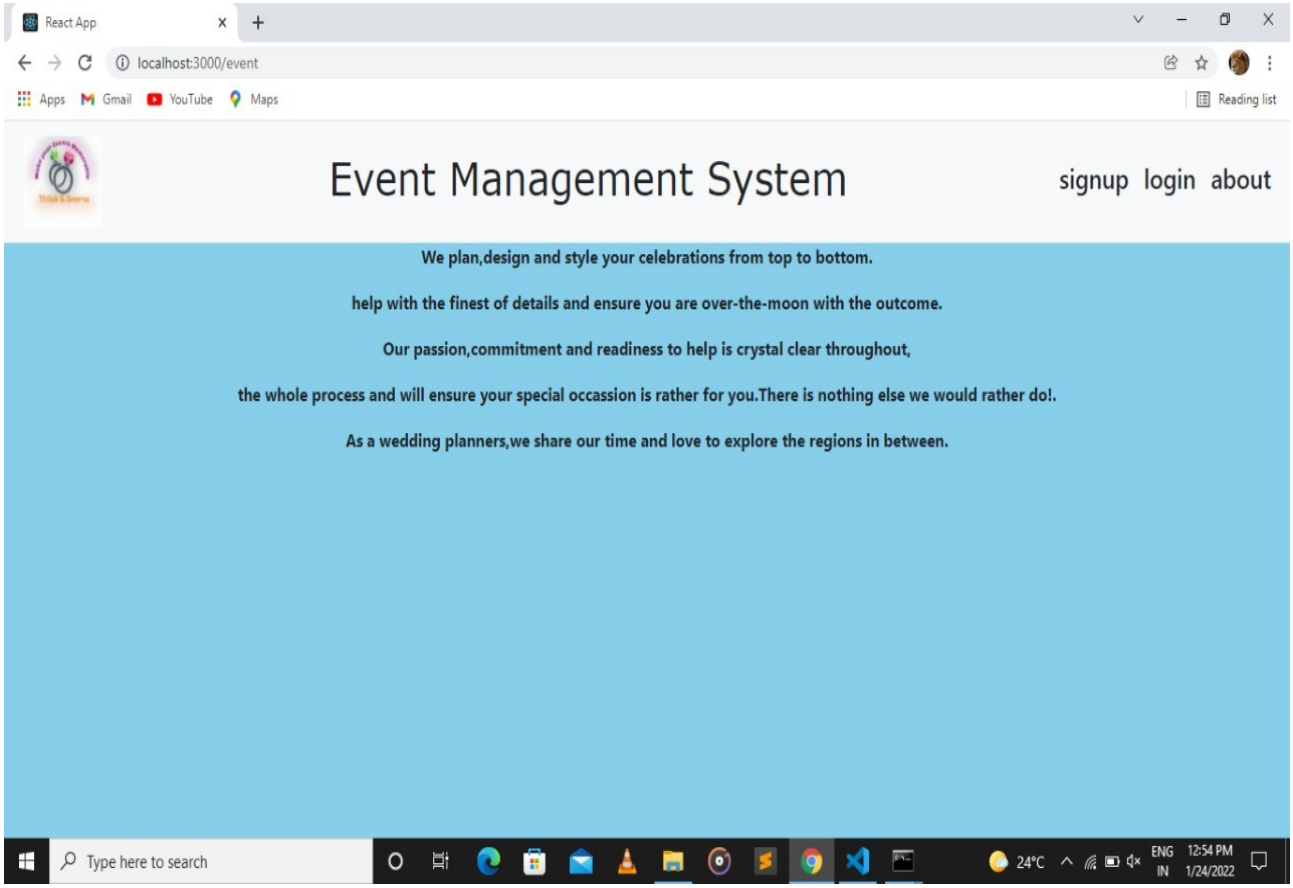
program structure

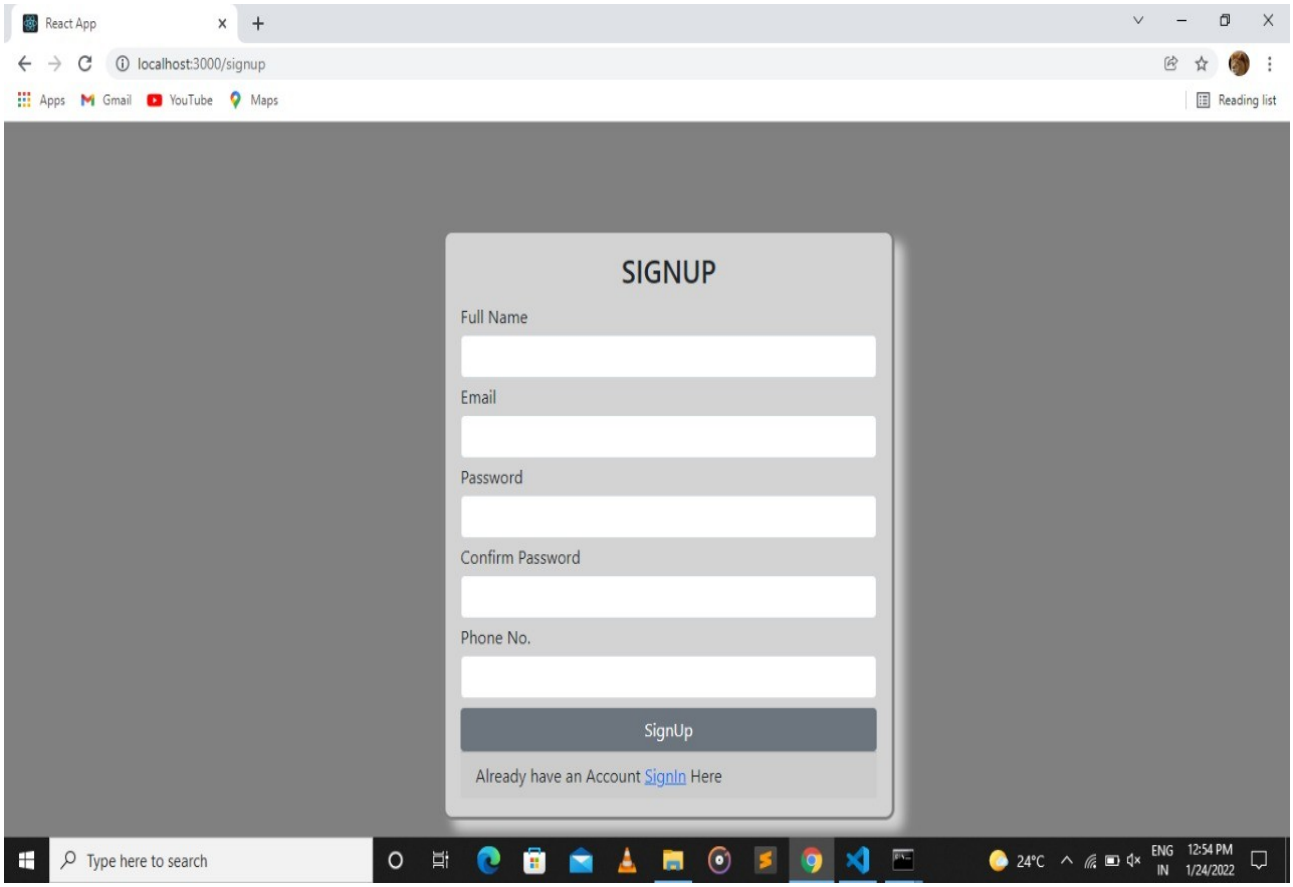
The bottom up approaches tests every module severally then every module is module is

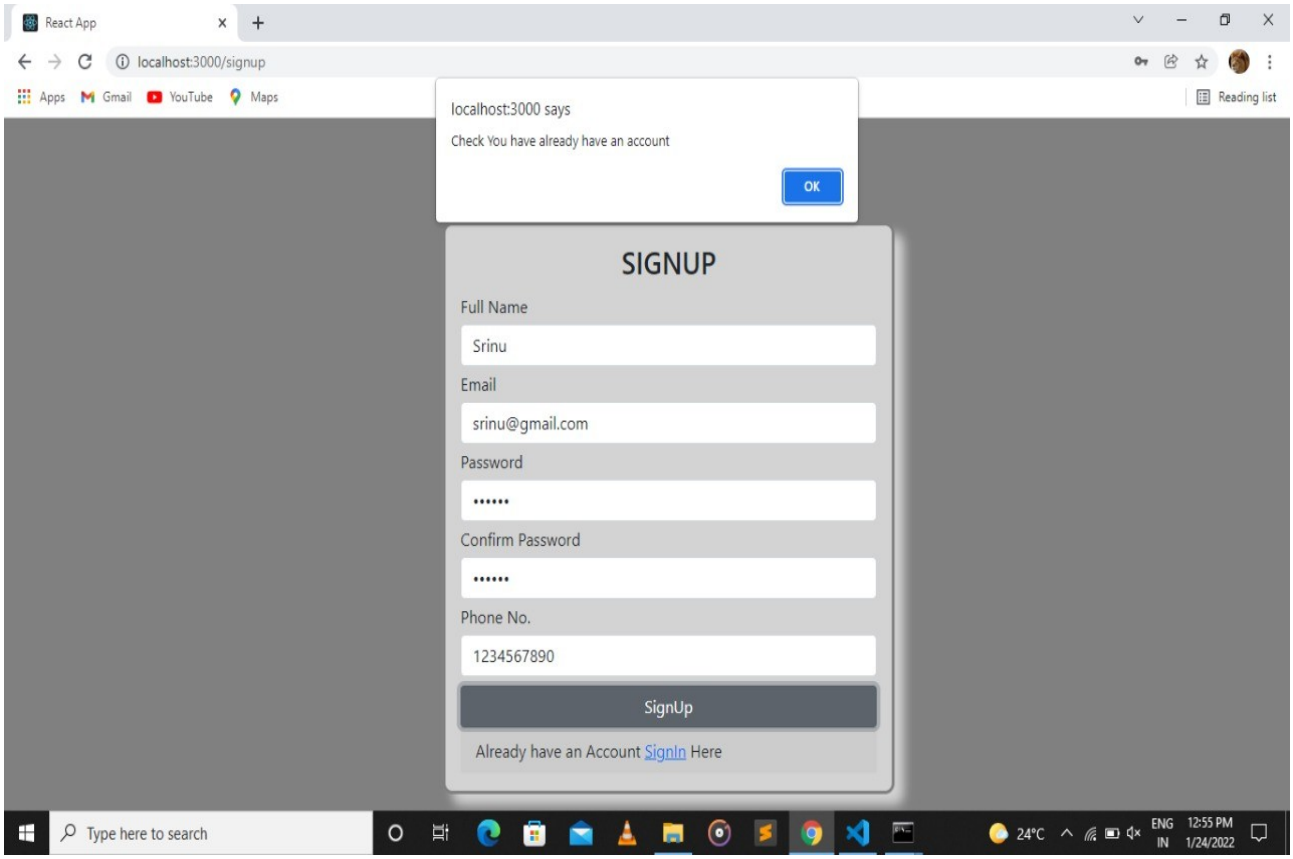
integrated with a main module and tested for practicality.

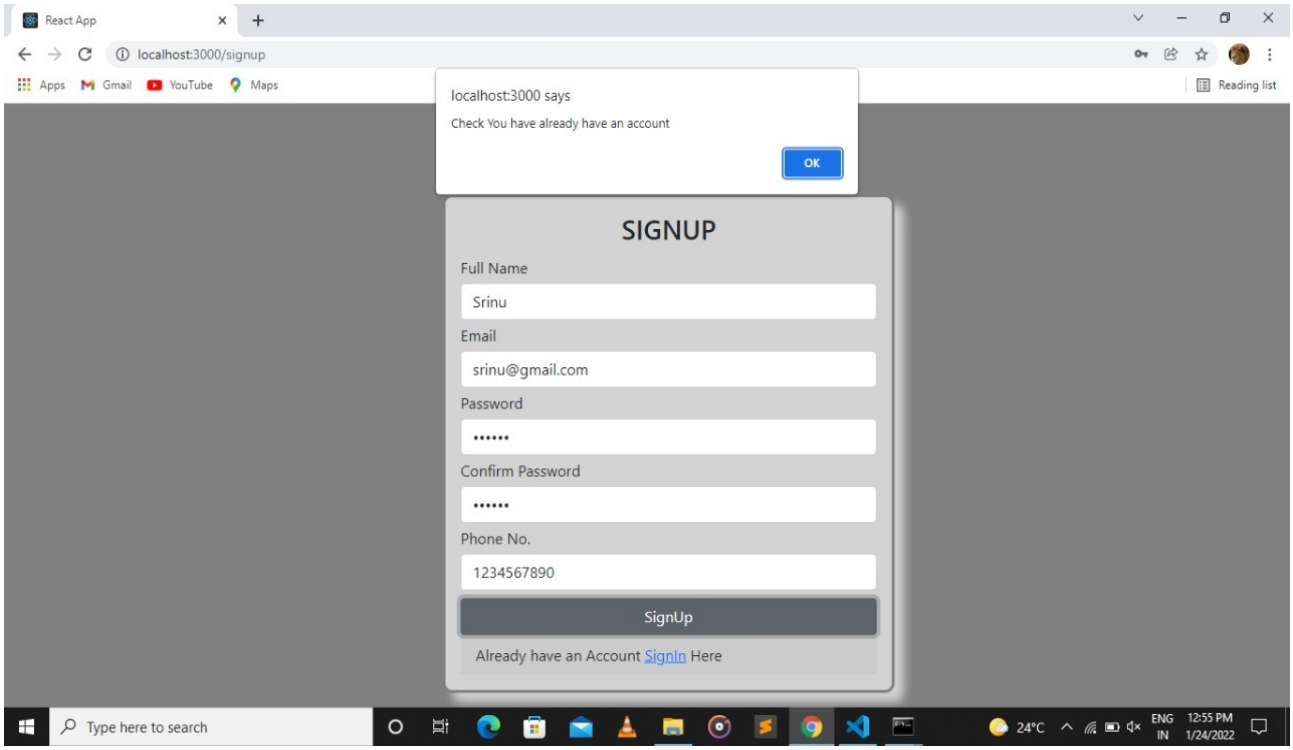
Outputs :

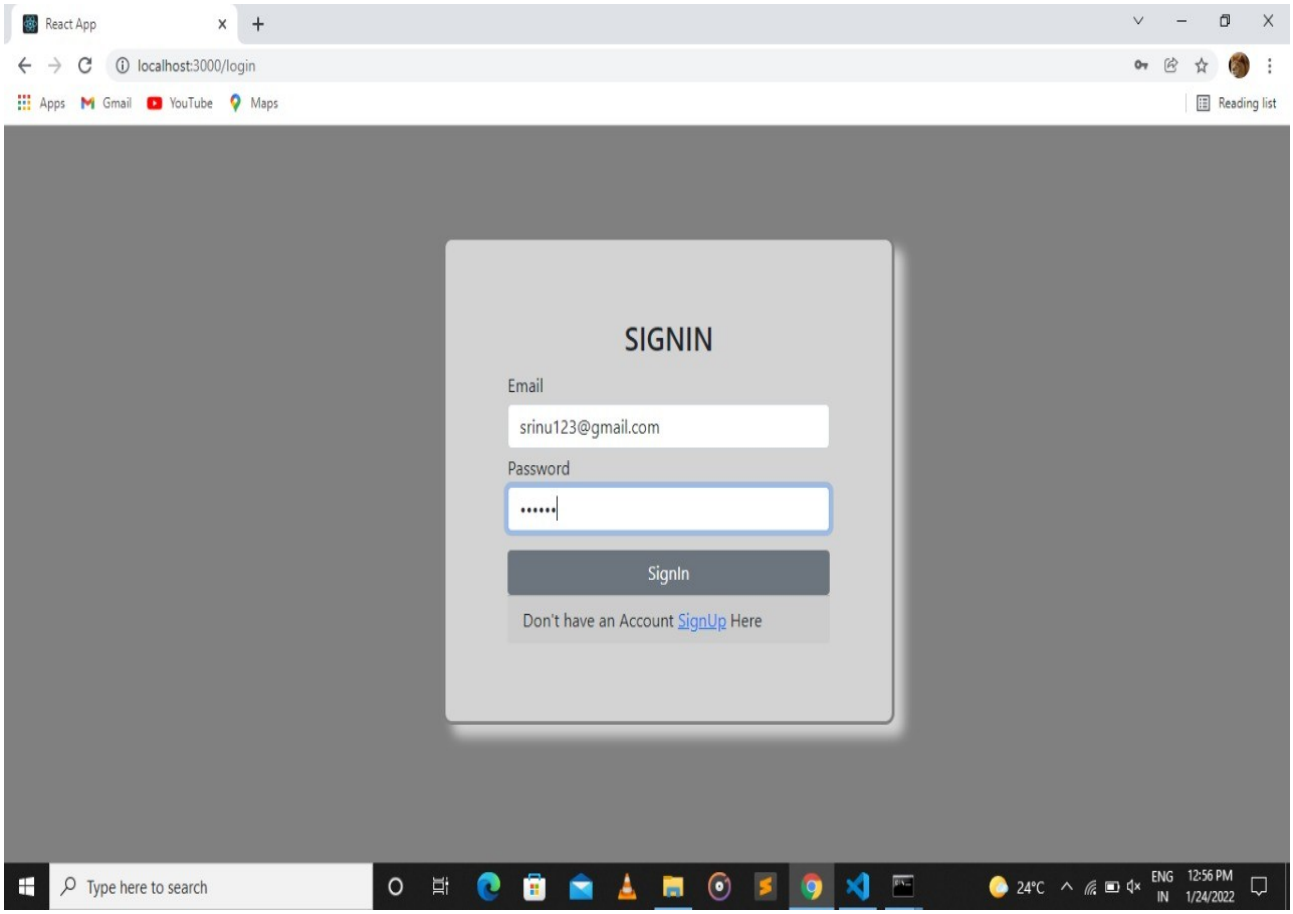


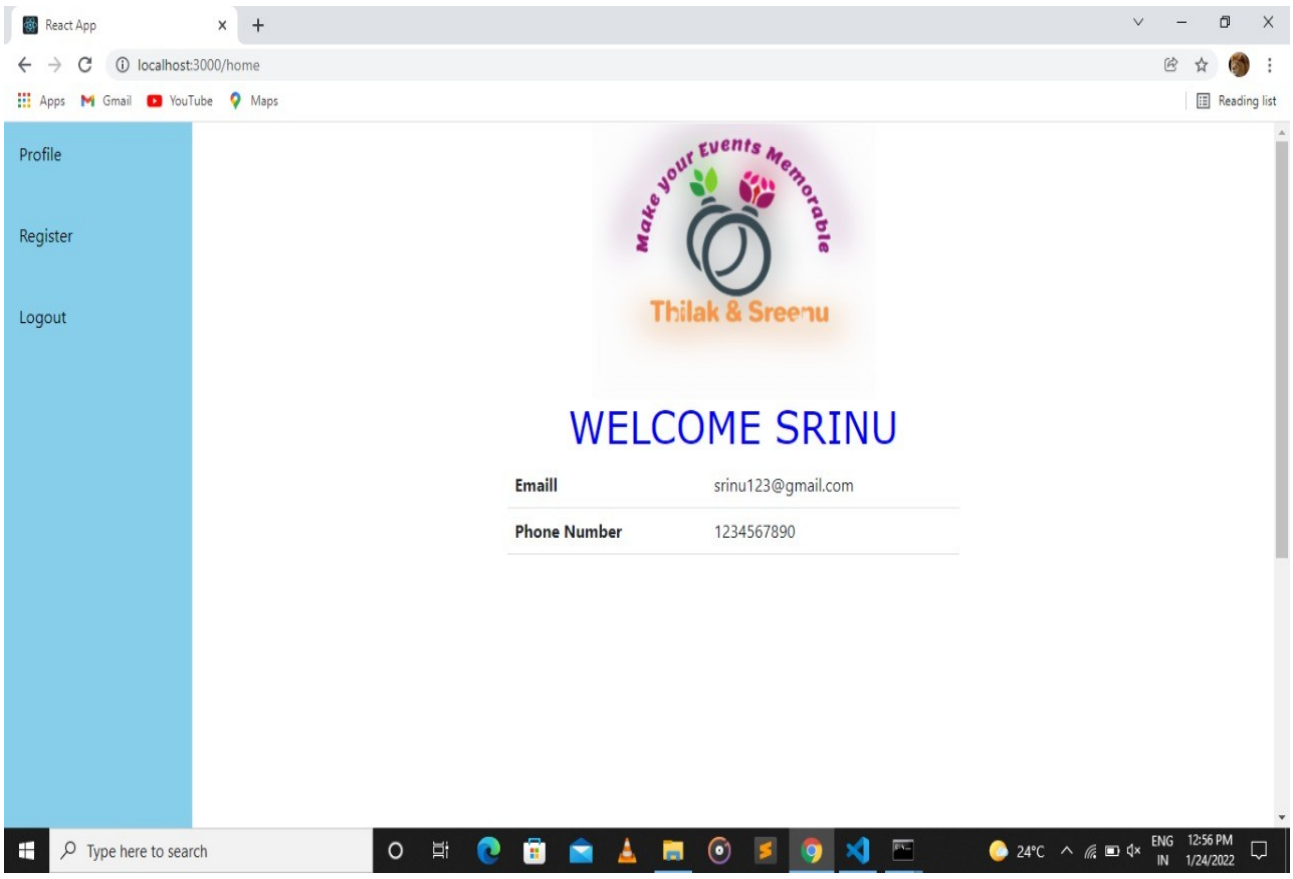

React App
localhost:3000/event
Apps
Gmail
YouTube
Maps
Reading list
Event Management System
signup
login
about
We plan,design and style your celebrations from top to bottom.
help with the finest of details and ensure you are over-the-moon with the outcome.
Our passion,commitment and readiness to help is crystal clear throughout,
the whole process and will ensure your special occassion is rather for you.There is nothing else we would rather do!.
As a wedding planners,we share our time and love to explore the regions in between.
Type here to search
24°C
ENG
IN
12:54 PM
1/24/2022

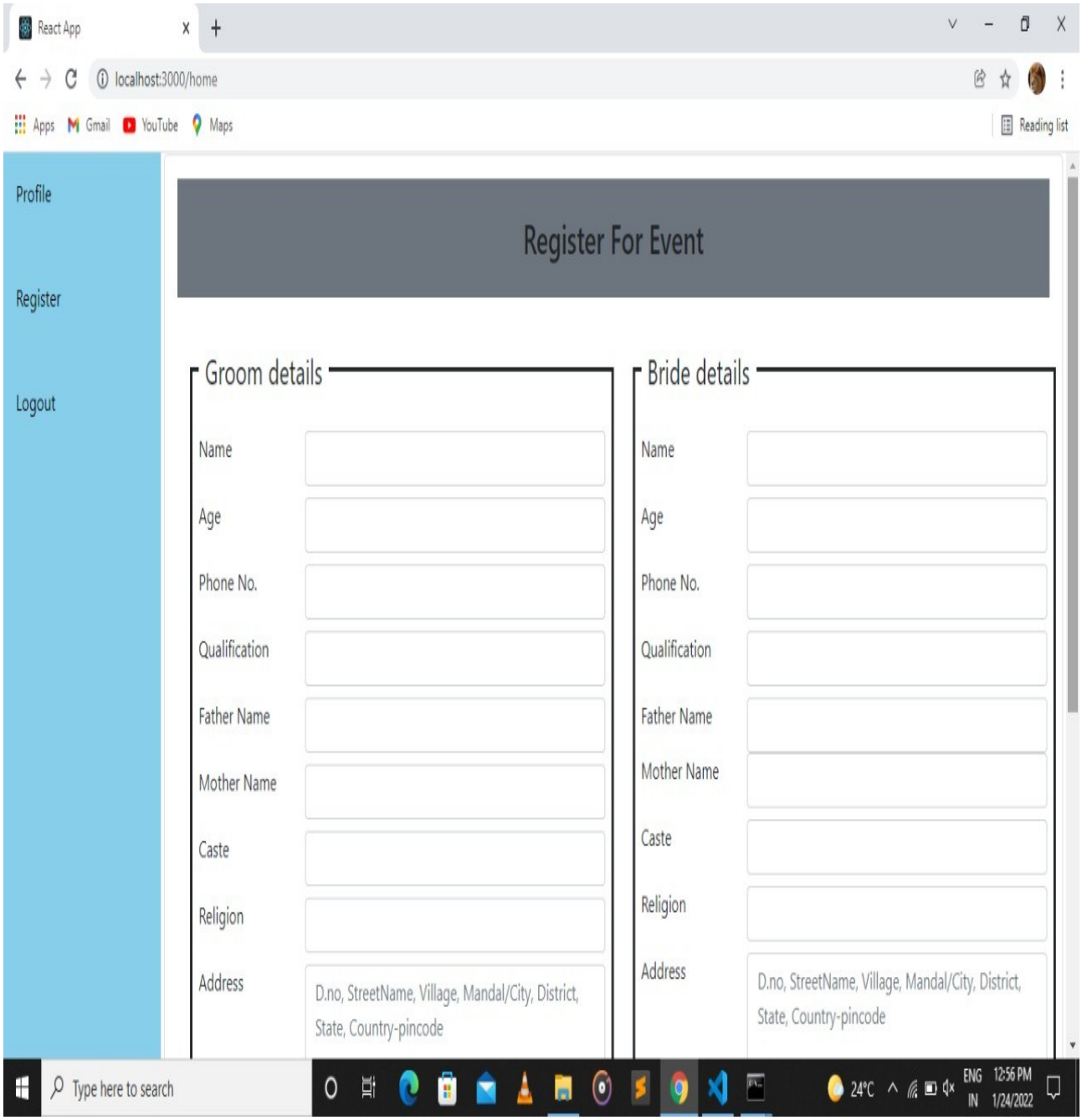

React App
localhost:3000/signup
Apps
Gmail
YouTube
Maps
Reading list
SIGNUP
Full Name
Email
Password
Confirm Password
Phone No.
SignUp
Already have an Account SignIn Here
Type here to search
24°C
ENG
IN
12:54 PM
1/24/2022


React App
localhost:3000/signup
Apps
Gmail
YouTube
Maps
Reading list
localhost:3000 says
Check You have already have an account
OK
SIGNUP
Full Name
Srinu
Email
srinu@gmail.com
Password
Confirm Password
Phone No.
1234567890
SignUp
Already have an Account SignIn Here
Type here to search
24°C
ENG
IN
12:55 PM
1/24/2022


React App
localhost:3000/signup
Apps
Gmail
YouTube
Maps
Reading list
localhost:3000 says
Check You have already have an account
OK
SIGNUP
Full Name
Srinu
Email
srinu@gmail.com
Password
Confirm Password
Phone No.
1234567890
SignUp
Already have an Account SignIn Here
Type here to search
24°C
ENG
IN
12:55 PM
1/24/2022


React App
localhost:3000/login
Apps
Gmail
YouTube
Maps
Reading list
SIGNIN
Email
srinu123@gmail.com
Password
SignIn
Don't have an Account SignUp Here
Type here to search
24°C
ENG
IN
12:56 PM
1/24/2022


React App
localhost:3000/home
Apps
Gmail
YouTube
Maps
Reading list
Profile
Register
Logout
Make your Events Memorable
Thilak & Sreenu
WELCOME SRINU
Emaill
srinu123@gmail.com
Phone Number
1234567890
Type here to search
24°C
ENG
IN
12:56 PM
1/24/2022


React App
localhost:3000/home
Apps
Gmail
YouTube
Maps
Reading list
Profile
Register
Logout
Register For Event
Groom details
Name
Age
Phone No.
Qualification
Father Name
Mother Name
Caste
Religion
Address
D.no, StreetName, Village, Mandal/City, District, State, Country-pincode
Bride details
Name
Age
Phone No.
Qualification
Father Name
Mother Name
Caste
Religion
Address
D.no, StreetName, Village, Mandal/City, District, State, Country-pincode
Type here to search
24°C
ENG
IN
12:56 PM
1/24/2022

React Js code

```
import React from 'react';
import Login from './Login';
import HomePage from "./components/HomePage"
import Home from './Home';

import Signup from "./components/Accounts/Signup";
import Event from "./components/Event";
import Auth from "./components/Auth";
import About from "./components/About"
import {BrowserRouter as Router,Route,Switch} from
'react-router-dom';

function App() {
  return (
   <Router>
    <Switch>
        <Route exact path="/" ><HomePage /> </Route>
        <Route exact path="/event" ><Event />
</Route>
          <Route exact path="/login"><Login /></Route>
          <Route exact path="/signup"><Signup
/></Route>
          <Route exact path="/about"><About
/></Route>
```

```
          <Auth exact path="/home"
component={Home} />




    </Switch>
   </Router>
  );
}

export default App;

import { render, screen } from '@testing-library/react';
import App from './App';

test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});


import React from 'react';
import './Home.css';
import Sidebar from './Sidebar';
```

```
import './Sidebar.css';

export default function Home(props){
    return(
        <div className="fluid-container">
            <Sidebar />
        </div>
    )
}


import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import "bootstrap/dist/css/bootstrap.min.css";
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);

// If you want to start measuring performance in your
app, pass a function
// to log results (for example:
reportWebVitals(console.log))
```

```
// or send to an analytics endpoint. Learn more:
https://bit.ly/CRA-vitals

import React from 'react';
import './Login.css';
import axios from "axios";
import {useHistory, Link} from "react-router-dom";
import {BASE_URL} from "./shared/BaseUrl"
export default function Login(props){
    const [email, setEmail] = React.useState("");
    const [password, setPassword] =
React.useState("");
    const [success, setS] = React.useState(false)
    const history = useHistory()
    const onSubmit = async (event) =>{
        event.preventDefault();
        const res = await
axios.post(`${BASE_URL}/login`, {email,
password});

        if(res.data.token){
            localStorage.setItem("token",
JSON.stringify(res.data.token));
            history.push("/home");
        }
        else setS(true)

    }
```

```
    return(
        <div className="fluid-container"
style={{background:'gray', width:'100%',
height:'100vh'}}>
            <div className="card row"
style={{padding:'50px',background:'lightgray',boxSha
dow:'10px 10px 10px lightgray',width:'30rem',
position:'absolute', top:'15%', left:'35%',
trnasform:'translateX(-15%, -35%)', border:'3px solid
gray', borderRadius:'10px'}}>



                <div className="card-body">
                    <h3 className="text-center">SIGNIN</
h3>
                    <div className="Card-body" >
                        <form>

                            <div className="form-group">
                                <label for="email"
className="py-1">Email</label>
                                <input id="email" type="email"
name="email"
onChange={(event)=>setEmail(event.target.value)}
className="form-control" autoComplete="off" />
                            </div>
                            <div className="form-group">
```

```
                        <label for="password"
className="py-1">Password</label>
                        <input id="password"
type="password" name="password"
onChange={(event)=>setPassword(event.target.value)
} className="form-control" autoComplete="off"/>
                    </div>
                    <div className="d-grid">
                        <button className="btn btn-
secondary mt-3 "
onClick={onSubmit}>SignIn</button>
                    </div>

                </form>
            </div>
            <div className="card-footer">
                Don't have an Account <Link
to="/signup">SignUp</Link> Here
            </div>
            {success && <div className="alert
alert-danger" role="alert">
                Check Credentials
            </div>}
        </div>
    </div>

</div>
```

```
    )
}

import React from 'react';
import './Login.css';
export default function Registration(props){
    return(
        <>
        <div>
            <form className="register">
                <h1>Register Here</h1>
                <label>Name<br />
                <input type="text" placeholder="Name"
/></label>
                <label>Email<br />
                <input type="email" placeholder="Email" /
></label>
                <label>Groom Name<br />
                <input type="text" placeholder="Groom"
/></label>
                <label>Bride Name<br />
                <input type="text" placeholder="Bride"
/></label>
                <label>Address<br />
                <input type="text" placeholder="Address" /
></label>
                <label>Mobile<br />
```

```
            <input type="varchar"
placeholder="Mobile" /></label>
            <label aria-label="left">Marriage
Type</label>
            <select className="select">
               <option value="Hindu">Hindu</option>
               <option
value="Muslim">Muslim</option>
               <option
value="Christian">Christian</option>
               <option value="Others">Others</option>
            </select>
            <label>Event Location<br />
            <input type="text" placeholder="Event
Location" /></label>
            <div>
            <button type="submit">Submit</button>
            <button type="submit">Reset</button>
            </div>
         </form>
      </div>
      </>
   )

}

const reportWebVitals = onPerfEntry => {
```

```
  if (onPerfEntry && onPerfEntry instanceof
Function) {
    import('web-vitals').then(({ getCLS, getFID,
getFCP, getLCP, getTTFB }) => {
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};

export default reportWebVitals;

import React from 'react';
import Content from "./components/Content"
import {useHistory} from "react-router-dom";
export default function Sidebar(props){
    const [page, setPage] = React.useState("");
    const history = useHistory();
    return(
        <>
            <div class="sidebar">
                <p
onClick={()=>setPage("Profile")}>Profile</p>
                <p
onClick={()=>setPage("Register")}>Register</p>
```

```
                <p onClick={()=>{
                        setPage("/home")
                        localStorage.clear();
                        history.push("/");
                    }}>Logout</p>
            </div>
            <Content pageName={page}/>


        </>
    );
}
```

CSS Code

```
.top{
    font-family: Verdana;
    background-color: lightsalmon;
    padding-top: 10px;
    display: flex;

}
.top p{
    padding: 0px 5px;

}
.top .list{
    display: flex;
```

```
    margin-left: 70%;

}
.below{
    background-color: lightskyblue;
    width: 100%;
    position: fixed;
    height: 100vh;
}
.logo{
    height: 50px;
    width: 50px;
    border-radius: 50%;
    line-height: 50px;
}
.list p:hover{
    background-color:springgreen;
}

*{
    padding: 0px;
    margin:0px;

}

.form{
    background-color:lightgreen;
    align-items: center;
```

```
    padding: 30px;
    max-width: 25%;
    font-family: Arial;
    text-align: center;
    display:flex;
    flex-direction: column;
    font-family: verdana;
    margin: auto;
    width: 50%;
    border: 5px solid red;
}
h1{
    text-align: center;
    text-decoration: none;
    color:blue;
    font-family: verdana;
}
.h2{
    text-align: center;
    text-decoration: none;
    color:red;
    font-family: verdana;
}
.form input[type="text"],.form
input[type="password"]{
      border:0;
      background:grey;
      margin:20px auto;
```

```
      text-align: center;
      border:2px solid #3498bd;
      padding: 14px 10px;
    width: 200px;
      outline: none;
      color:white;
      border-radius: 24px;
      transition: 0.25s;
}
.radio{
    display:flex;
    flex-direction:row;
}
.button{
    padding: 10;
    font-family: verdana;
}
.register{
    text-align: left;
    display:flex;
    flex-direction:column;
    align-items: left;
    padding: 20px;
    max-width: 40%;
    border: 5px solid blue;
    width: 400px;
      margin: 100px auto 0px;
    background-color: rgba(0,0,0,0.5);
```

```
    font-size: 18px;
    border-radius: 10px;
    border: 1px solid rgba(255,255,255,0.3);
    box-shadow: 2px 2px 15px rgba(0,0,0,0.3);
    color: #ffff;
}
.register input{
    width:300px;
    border: 1px solid #ddd;
    border-radius: 3px;
    outline: 0;
    padding: 7px;
    background-color: #fff;
    box-shadow: inset 1px 1px 5px rgba(0,0,0,0.3);
}
.register button{
    width: 200px;
    padding: 10px;
    font-size: 16px;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-weight: 600;
    border-radius: 3px;
    background-color: rgba(250,100,0,0.8);
    color: #fff;
    cursor: pointer;
    border: 1px solid rgba(255,255,255,0.3);
    box-shadow: 1px 1px 5px rgba(0,0,0,0.3);
    margin-bottom: 20px;
```

```
    margin-top: 20px;
}
.select{
    border-radius: 10px;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    width:320px;
    border: 1px solid #ddd;
    border-radius: 3px;
    outline: 0;
    padding: 7px;
    background-color: #fff;
    box-shadow: inset 1px 1px 5px rgba(0,0,0,0.3);
}

.sidebar {

  width: 200px;
  background-color: skyblue;
  position: fixed;
  height: 100vh;
  overflow: auto;
}

.sidebar p {
  display: block;
  color: black;
  padding: 16px;
  text-decoration: none;
```

```
}

.sidebar p.active {
  background-color: #04AA6D;
  color: white;
}

.sidebar p:hover:not(.active) {
  background-color: #555;
  color: white;
}

div.content {
  margin-left: 200px;
  padding: 1px 16px;
  height: 1000px;
}

@media screen and (max-width: 700px) {
  .sidebar {
    width: 100%;
    height: auto;
    position: relative;
  }
  .sidebar a {float: left;}
  div.content {margin-left: 0;}
}
```

```
@media screen and (max-width: 400px) {
  .sidebar a {
    text-align: center;
    float: none;
  }
}
```

Conclusion

This web application Event Management fullfill the requirements of the user who needs to register for an Marriage event. User can give details to the management team. Based on the user requirements Management team will manage the event.

BIBILOGRAPHY

References

https://www.google.com/search?channel=fs&client=ubuntu&q=html+w3schools

https://www.javatpoint.com/reactjs-tutorial

https://www.javatpoint.com/firebase

https://www.javatpoint.com/firebase-realtime-database-reading-and-writing