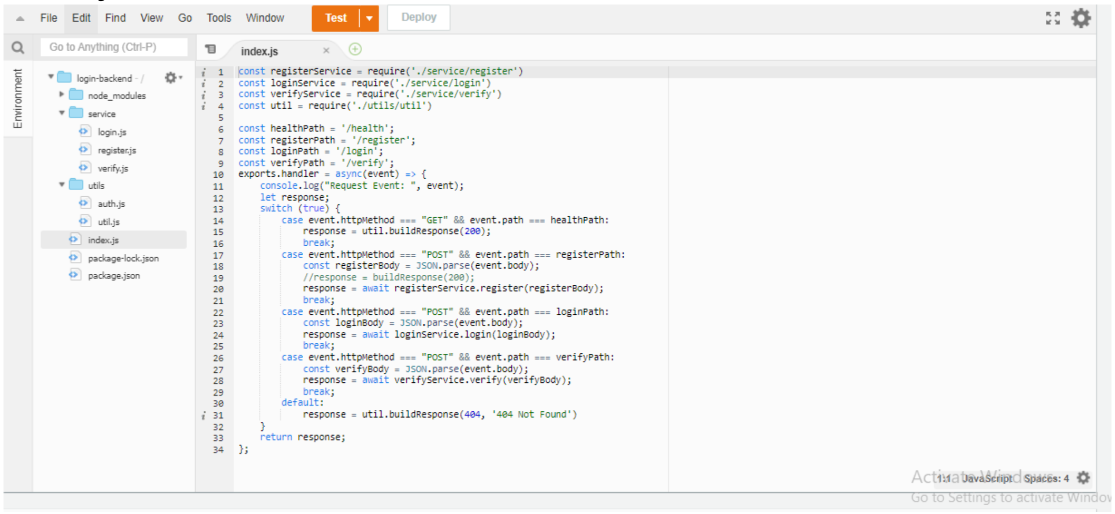
Review - 3

Name: Saptajit Banerjee
Registration Number: 20BCE1513

Code:

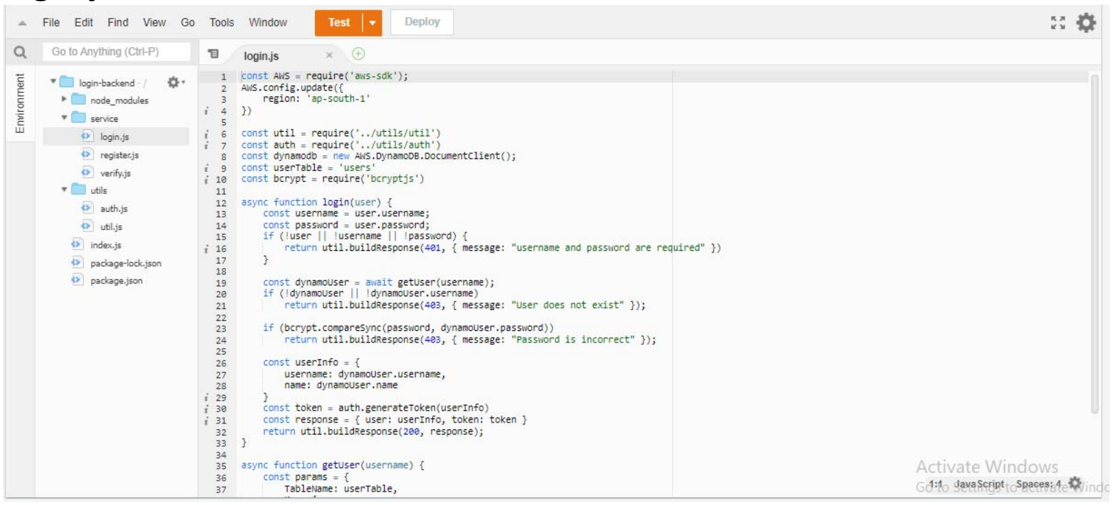
Backend:

index.js:

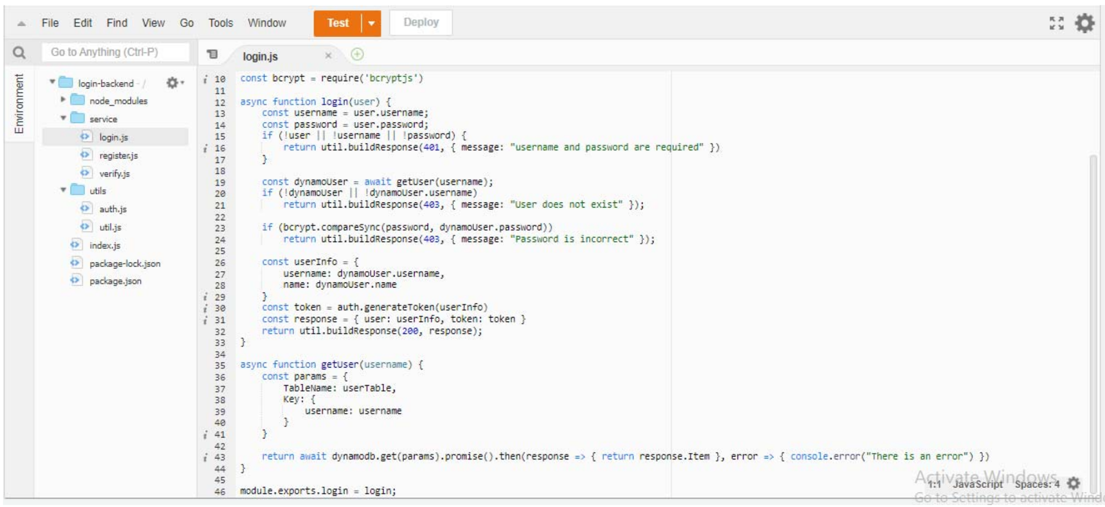


```javascript
const registerService = require('./service/register')
const loginService = require('./service/login')
const verifyService = require('./service/verify')
const util = require('./utils/util')

const healthPath = '/health';
const registerPath = '/register';
const loginPath = '/login';
const verifyPath = '/verify';
exports.handler = async(event) => {
    console.log("Request Event: ", event);
    let response;
    switch (true) {
        case event.httpMethod === "GET" && event.path === healthPath:
            response = util.buildResponse(200);
            break;
        case event.httpMethod === "POST" && event.path === registerPath:
            const registerBody = JSON.parse(event.body);
            //response = buildResponse(200);
            response = await registerService.register(registerBody);
            break;
        case event.httpMethod === "POST" && event.path === loginPath:
            const loginBody = JSON.parse(event.body);
            response = await loginService.login(loginBody);
            break;
        case event.httpMethod === "POST" && event.path === verifyPath:
            const verifyBody = JSON.parse(event.body);
            response = await verifyService.verify(verifyBody);
            break;
        default:
            response = util.buildResponse(404, '404 Not Found')
    }
    return response;
};
```

login.js:

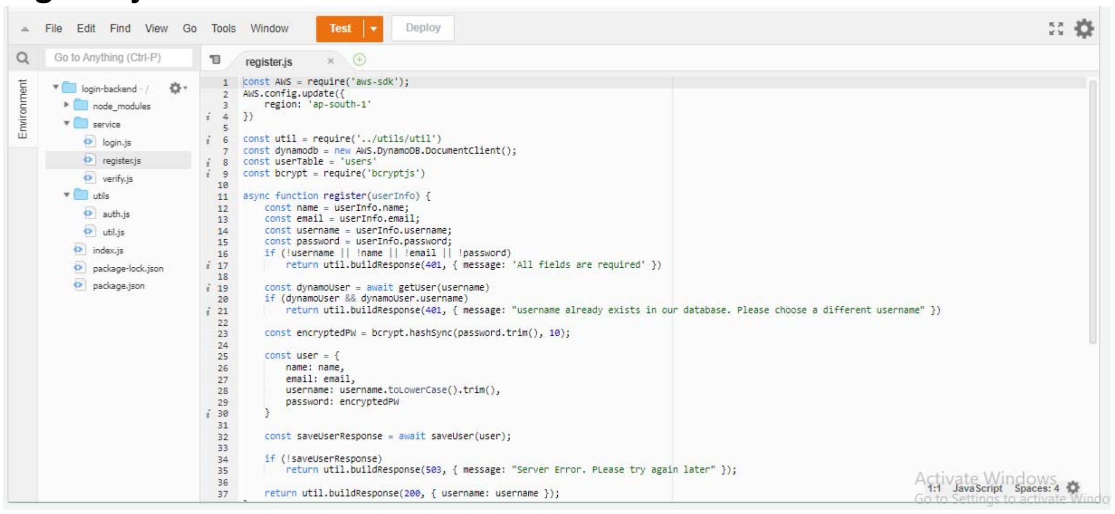


```javascript
const AWS = require('aws-sdk');
AWS.config.update({
    region: 'ap-south-1'
})

const util = require('../utils/util')
const auth = require('../utils/auth')
const dynamodb = new AWS.DynamoDB.DocumentClient();
const userTable = 'users'
const bcrypt = require('bcryptjs')

async function login(user) {
    const username = user.username;
    const password = user.password;
    if (!user || !username || !password) {
        return util.buildResponse(401, { message: "username and password are required" })
    }

    const dynamoUser = await getUser(username);
    if (!dynamoUser || !dynamoUser.username)
        return util.buildResponse(403, { message: "User does not exist" });

    if (bcrypt.compareSync(password, dynamoUser.password))
        return util.buildResponse(403, { message: "Password is incorrect" });

    const userInfo = {
        username: dynamoUser.username,
        name: dynamoUser.name
    }
    const token = auth.generateToken(userInfo)
    const response = { user: userInfo, token: token }
    return util.buildResponse(200, response);
}

async function getUser(username) {
    const params = {
        TableName: userTable,
```

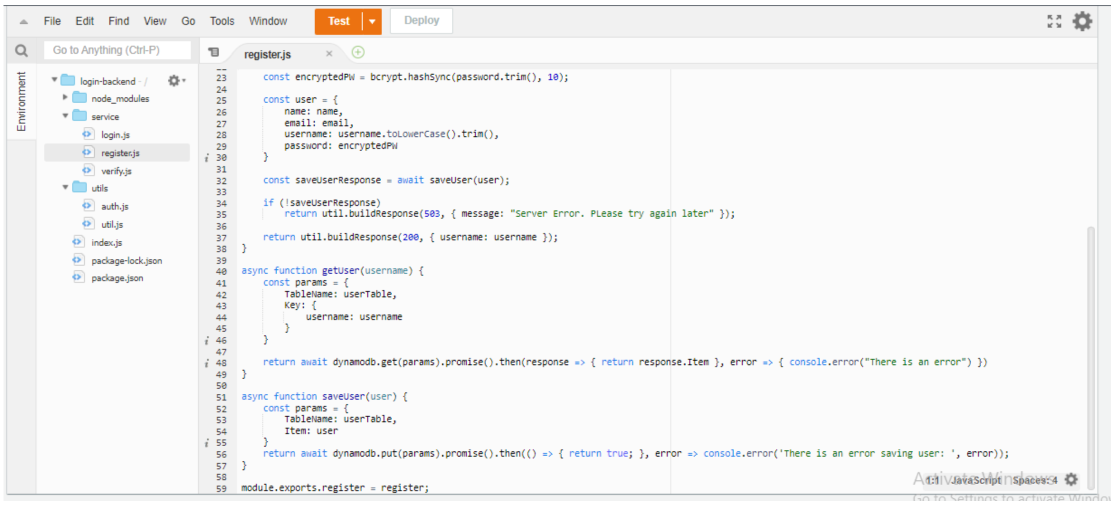


```javascript
const bcrypt = require('bcryptjs')

async function login(user) {
    const username = user.username;
    const password = user.password;
    if (!user || !username || !password) {
        return util.buildResponse(401, { message: "username and password are required" })
    }

    const dynamoUser = await getUser(username);
    if (!dynamoUser || !dynamoUser.username)
        return util.buildResponse(403, { message: "User does not exist" });

    if (bcrypt.compareSync(password, dynamoUser.password))
        return util.buildResponse(403, { message: "Password is incorrect" });

    const userInfo = {
        username: dynamoUser.username,
        name: dynamoUser.name
    }
    const token = auth.generateToken(userInfo)
    const response = { user: userInfo, token: token }
    return util.buildResponse(200, response);
}

async function getUser(username) {
    const params = {
        TableName: userTable,
        Key: {
            username: username
        }
    }

    return await dynamodb.get(params).promise().then(response => { return response.Item }, error => { console.error("There is an error") })
}

module.exports.login = login;
```

regitser.js:

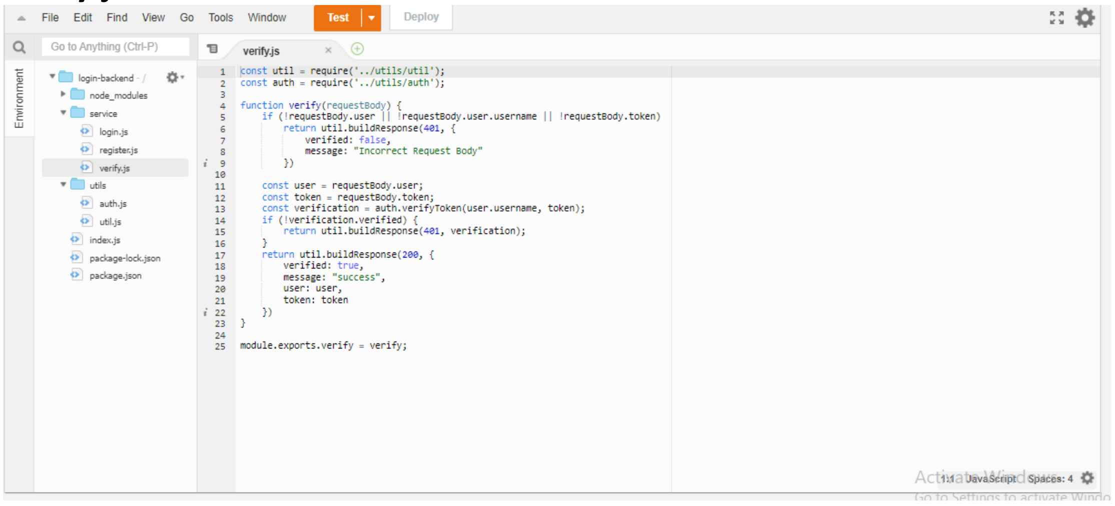


```javascript
const AWS = require('aws-sdk');
AWS.config.update({
    region: 'ap-south-1'
})

const util = require('../utils/util')
const dynamodb = new AWS.DynamoDB.DocumentClient();
const userTable = 'users'
const bcrypt = require('bcryptjs')

async function register(userInfo) {
    const name = userInfo.name;
    const email = userInfo.email;
    const username = userInfo.username;
    const password = userInfo.password;
    if (!username || !name || !email || !password)
        return util.buildResponse(401, { message: 'All fields are required' })

    const dynamoUser = await getUser(username)
    if (dynamoUser && dynamoUser.username)
        return util.buildResponse(401, { message: "username already exists in our database. Please choose a different username" })

    const encryptedPW = bcrypt.hashSync(password.trim(), 10);

    const user = {
        name: name,
        email: email,
        username: username.toLowerCase().trim(),
        password: encryptedPW
    }

    const saveUserResponse = await saveUser(user);

    if (!saveUserResponse)
        return util.buildResponse(503, { message: "Server Error. PLease try again later" });

    return util.buildResponse(200, { username: username });
```



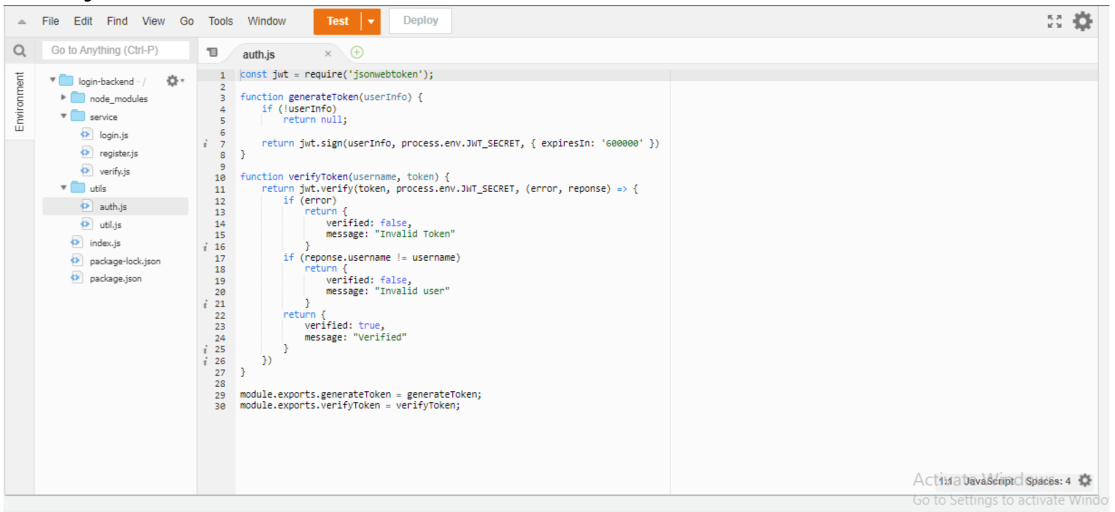
```javascript
    const encryptedPW = bcrypt.hashSync(password.trim(), 10);

    const user = {
        name: name,
        email: email,
        username: username.toLowerCase().trim(),
        password: encryptedPW
    }

    const saveUserResponse = await saveUser(user);

    if (!saveUserResponse)
        return util.buildResponse(503, { message: "Server Error. PLease try again later" });

    return util.buildResponse(200, { username: username });
}

async function getUser(username) {
    const params = {
        TableName: userTable,
        Key: {
            username: username
        }
    }

    return await dynamodb.get(params).promise().then(response => { return response.Item }, error => { console.error("There is an error") })
}

async function saveUser(user) {
    const params = {
        TableName: userTable,
        Item: user
    }
    return await dynamodb.put(params).promise().then(() => { return true; }, error => console.error('There is an error saving user: ', error));
}

module.exports.register = register;
```

verify.js



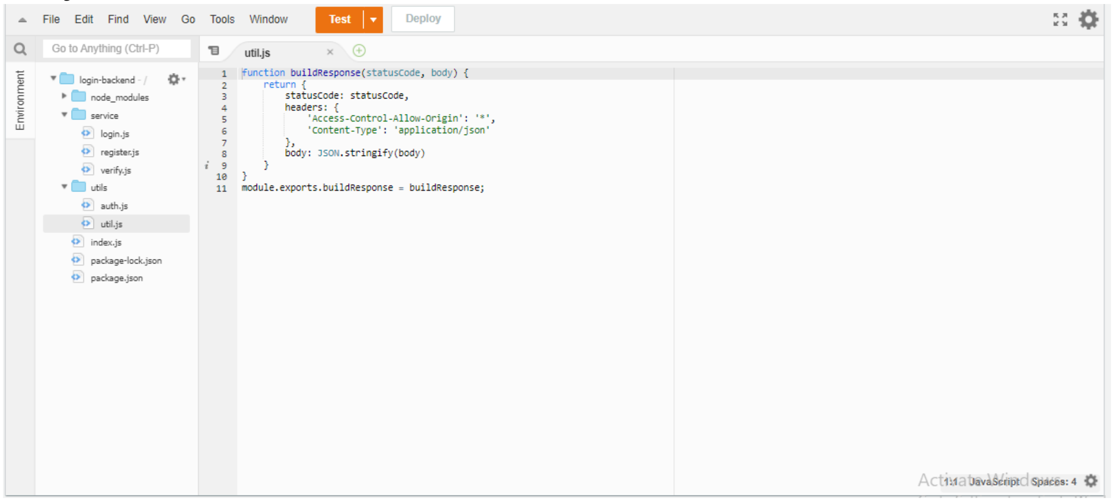
```javascript
const util = require('../utils/util');
const auth = require('../utils/auth');

function verify(requestBody) {
    if (!requestBody.user || !requestBody.user.username || !requestBody.token)
        return util.buildResponse(401, {
            verified: false,
            message: "Incorrect Request Body"
        })

    const user = requestBody.user;
    const token = requestBody.token;
    const verification = auth.verifyToken(user.username, token);
    if (!verification.verified) {
        return util.buildResponse(401, verification);
    }
    return util.buildResponse(200, {
        verified: true,
        message: "success",
        user: user,
        token: token
    })
}

module.exports.verify = verify;
```

auth.js



```javascript
const jwt = require('jsonwebtoken');

function generateToken(userInfo) {
    if (!userInfo)
        return null;

    return jwt.sign(userInfo, process.env.JWT_SECRET, { expiresIn: '600000' })
}

function verifyToken(username, token) {
    return jwt.verify(token, process.env.JWT_SECRET, (error, reponse) => {
        if (error)
            return {
                verified: false,
                message: "Invalid Token"
            }
        if (reponse.username != username)
            return {
                verified: false,
                message: "Invalid user"
            }
        return {
            verified: true,
            message: "Verified"
        }
    })
}

module.exports.generateToken = generateToken;
module.exports.verifyToken = verifyToken;
```

util.js



```javascript
function buildResponse(statusCode, body) {
    return {
        statusCode: statusCode,
        headers: {
            'Access-Control-Allow-Origin': '*',
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(body)
    }
}
module.exports.buildResponse = buildResponse;
```

Frontend:

App.jsx:

```jsx
import React,{ useState,useEffect } from 'react';
import { BrowserRouter, NavLink ,Route,Routes} from "react-router-dom";
import Home from "./Home"
import Login from "./Login"
import Register from "./Register";
import PremiumContent from "./PremiumContent";
import './App.css';
import { getUser,getToken,setUserSession,resetUserSession } from
'./Service';
import PublicRoute from './PublicRoute';
import PrivateRoute from './PrivateRoute';
import axios from 'axios';
function App() {
    const tokenUrl="https://fgs58drze0.execute-api.ap-south-
1.amazonaws.com/login/verify";
    const [isAuthenticating,setAuthenticating]=useState(true);
    useEffect(()=>{
        const token=getToken();
        if(token==='undefined'||token===undefined||token===null||!token)
        {
            return;
        }
        const requestConfig={headers:{'x-api-
key':"r8oOJoRZshaNcnJ3dJx3g6ZWhyf2NkVw18YGmgVC"}}
        const requestBody = {
            user:getUser(),
            token:token
        }
        axios.post(tokenUrl,requestBody,requestConfig).then(response=>{
            setUserSession(response.data.user,response.data.token);
            setAuthenticating(false);
        }).catch(error=>{
            resetUserSession();
            setAuthenticating(false);
        })
    },[]);
    const token = getToken();
    if(isAuthenticating && token)
    return <h4>Authenticating...</h4>;
    return (
        <div className = "App" >
            <BrowserRouter>
            <div className="header">
                <NavLink exact activeClassName="active"
to="/">Home</NavLink>
                <NavLink activeClassName="active"
to="/register">Register</NavLink>
                <NavLink activeClassName="active"
to="/login">Login</NavLink>
                <NavLink activeClassName="active" to="/premium-
content">Premium Content</NavLink>
```

```
            </div>
            <div className="content">
                <Routes>
                    <Route element={<PublicRoute/>}>
                    <Route exact path='/' element={<Home/>}/>
                    <Route path='/register' element={<Register/>}/>
                    <Route path='/login' element={<Login/>}/>
                    </Route>
                    <Route element={<PrivateRoute/>}>
                    <Route path='/premium-content'
element={<PremiumContent/>}/>
                    </Route>
                </Routes>
            </div>
            </BrowserRouter>
        </div>
    );
}
export default App;
```

App.css:

```
.header {
    border-bottom: 5px solid black;
    padding-bottom: 10px;
}

.header a {
    margin: 10px;
    text-decoration: none;
    color: black;
    font-size: 18px;
    font-family: Verdana;
}
```

index.jsx:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render( <React.StrictMode>
    <App/>
    </React.StrictMode>
);
```

Home.jsx:

```
import React from 'react';
const Home=()=>{
return(
    <div>
        This is Home page!
    </div>
)
}
export default Home;
```

Login.jsx:

```
import React,{useState} from 'react';
import axios from 'axios';
import { setUserSession } from './Service';
import { useNavigate } from 'react-router-dom';
const loginAPIUrl="https://fgs58drze0.execute-api.ap-south-
1.amazonaws.com/login/login";
const Login=()=>{
    const [username,setUsername]=useState('');
    const [password,setPassword]=useState('');
    const [errorMessage,setErrormessage]=useState(null);
    const navigate = useNavigate();
const submitHandler = (event) =>{
    event.preventDefault();
    if(username.trim()===''||password.trim()==='')
    {
        setErrormessage("Both Username and Password are required");
        return;
    }
    setErrormessage(null);
    const requestConfig = {
        headers:{'x-api-key':"r8oOJoRZshaNcnJ3dJx3g6ZWhyf2NkVw18YGmgVC"}
    }
    const requestBody = {
        username:username,
        password:password
    }
    axios.post(loginAPIUrl,requestBody,requestConfig).then((response)=>
{
        setUserSession(response.data.user,response.data.token);
        setErrormessage("Login Successful");
        //window.open("http://localhost:3000/");
        //window.close();
        navigate("/premium-content");
    }).catch((error)=>{
        if(error.response.status===401 || error.response.status===403)
        {setErrormessage(error.response.data.message);}
        else
        {setErrormessage("Backend Server Is Down, Please Try Again");}
    });
}
return(
    <div>
        <form onSubmit={submitHandler}>
            <h2>Login</h2>
            Username<br/>
            <input type="text" value={username} onChange={event =>
setUsername(event.target.value)}/><br/>
            Password<br/>
            <input type="password" value={password} onChange={event =>
setPassword(event.target.value)}/><br/>
            <input type="submit" value="LOGIN"/>
        </form>
        <p>{errorMessage}</p>
    </div>
)
}
export default Login;
```

Register.jsx:

```
import React,{useState}from 'react';
import axios from 'axios';
const registerUrl="https://fgs58drze0.execute-api.ap-south-
1.amazonaws.com/login/register";
const Register=()=>{
    const [name,setName]=useState('');
    const [email,setEmail]=useState('');
    const [username,setUsername] = useState('');
    const [password,setPassword]=useState('');
    const [message,setMessage]=useState('');
    const submitHandler = (event) =>{
        event.preventDefault();
        if(username.trim()===''||email.trim()===''||username.trim()==='
'||password.trim()==='')
        {
            setMessage("All Fields Are Required");
            return;
        }
        setMessage(null);
    const requestConfig = {
        headers:{'x-api-key':"r8oOJoRZshaNcnJ3dJx3g6ZWhyf2NkVw18YGmgVC"}
    }
    const requestBody = {
        username:username,
        email:email,
        name:name,
        password:password
    }
    axios.post(registerUrl,requestBody,requestConfig).then(response=>{
        setMessage("Registration Successful");
    }).catch(error=>{
        if(error.response.status===401){
            setMessage(error.response.data.message);
        }
        else{
            setMessage("Backend Server is Down Please Try Again");
        }
    });
    }
return(
    <div>
        <form onSubmit={submitHandler}>
            <h2>Register</h2>
            Name<br/><input type="text" value={name}
onChange={event=>setName(event.target.value)}/><br/>
            Email<br/><input type="email" value={email}
onChange={event=>setEmail(event.target.value)}/><br/>
            Username<br/><input type="text" value={username}
onChange={event=>setUsername(event.target.value)}/><br/>
            Password<br/><input type="password" value={password}
onChange={event=>setPassword(event.target.value)}/><br/>
            <br/><input type="submit" value="REGISTER"/>
        </form>
        <p>{message}</p>
    </div>
)
}
export default Register;
```

PremiumContent.jsx:

```
import React from 'react';
import { getUser,resetUserSession } from './Service';
import { useNavigate } from 'react-router-dom';
const PremiumContent=()=>{
    const navigate = useNavigate();
    const user = getUser();
    const name = user !== 'undefined' && user ? user.name:'';
    const logoutHandler=()=>{
        resetUserSession();
        navigate("/login");
    }
return(
    <div>
```

```
        Hello {name} !<br/>
        You Have Been Logged In!<br/>
        This is Premium Content page!<br/>
        <input type="button" value="LOG OUT" onClick={logoutHandler}/>
    </div>
)
}
export default PremiumContent;
```

PrivateRoute.jsx:

```
.header {
    border-bottom: 5px solid black;
    padding-bottom: 10px;
}

.header a {
    margin: 10px;
    text-decoration: none;
    color: black;
    font-size: 18px;
    font-family: Verdana;
}
```

PublicRoute.jsx:

```
import React from 'react';
import { Navigate,Outlet} from 'react-router-dom';
import { getToken } from './Service';

const PublicRoute = () => {
    return (
            !getToken() ? <Outlet/>: < Navigate to ='/premium-
content'/>
    )
}
export default PublicRoute
```

Service.js:

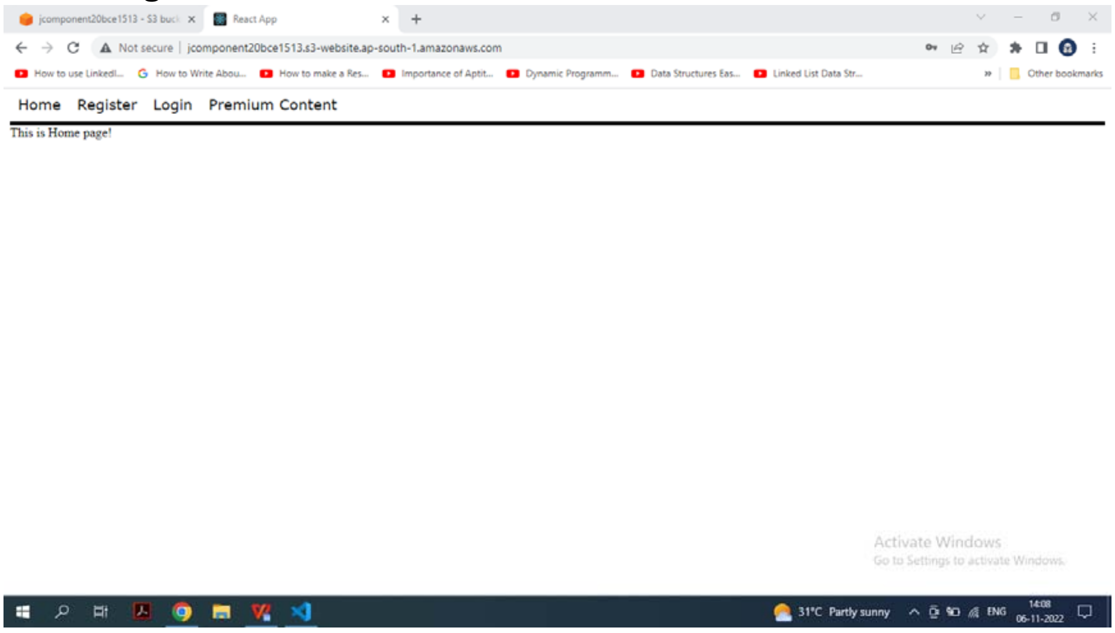
```
module.exports = {
    getUser: function() {
        const user = sessionStorage.getItem('user');
        if (user === 'undefined' || !user) {
            return null;
        } else {
            return JSON.parse(user);
        }
    },
    getToken: function() {
        return sessionStorage.getItem('token');
    },
    setUserSession: function(user, token) {
        sessionStorage.setItem('user', JSON.stringify(user));
        sessionStorage.setItem('token', token);
    },
    resetUserSession: function() {
        sessionStorage.removeItem('user');
        sessionStorage.removeItem('token');
    }
}
```

Output:

The system allows only the registered users to access the Premium Content page. To register, the users must go to Registration Page and fill all the details in that page and then click on the 'REGISTER' button to submit the details.

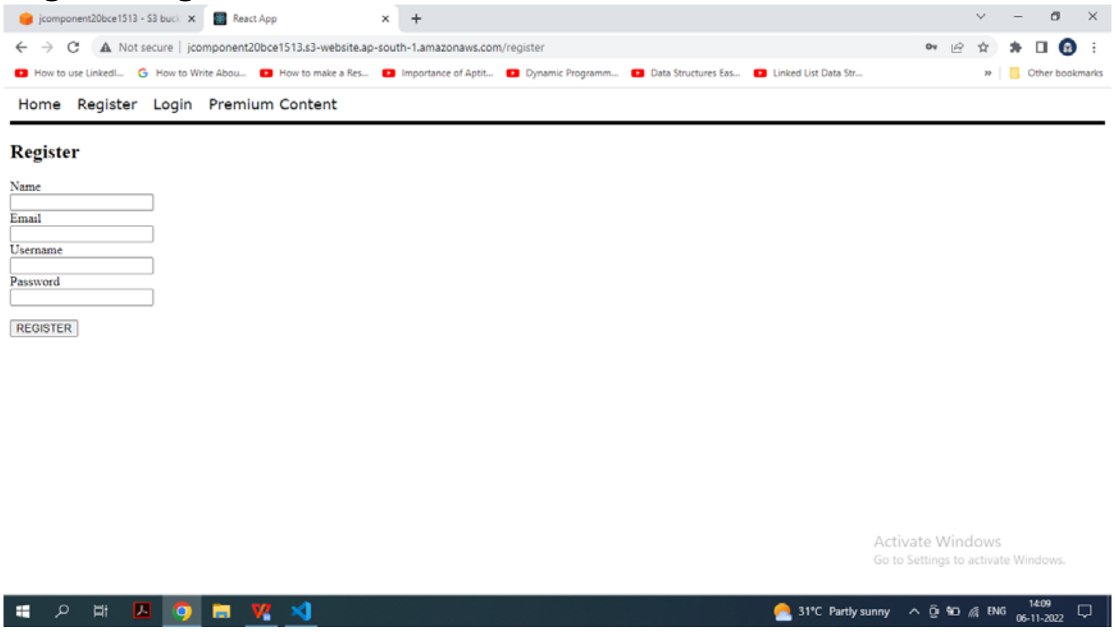
The whole system is put in a AWS S3 bucket. **The link to the S3 bucket is:** http://jcomponent20bce1513.s3-website.ap-south-1.amazonaws.com/

Home Page:


jcomponent20bce1513 - S3 buck
React App
Not secure | jcomponent20bce1513.s3-website.ap-south-1.amazonaws.com
Home Register Login Premium Content
Activate Windows
Go to Settings to activate Windows.
31°C Partly sunny
14:08
06-11-2022

This is Home page!

Register Page:



Home Register Login Premium Content

Register

Name

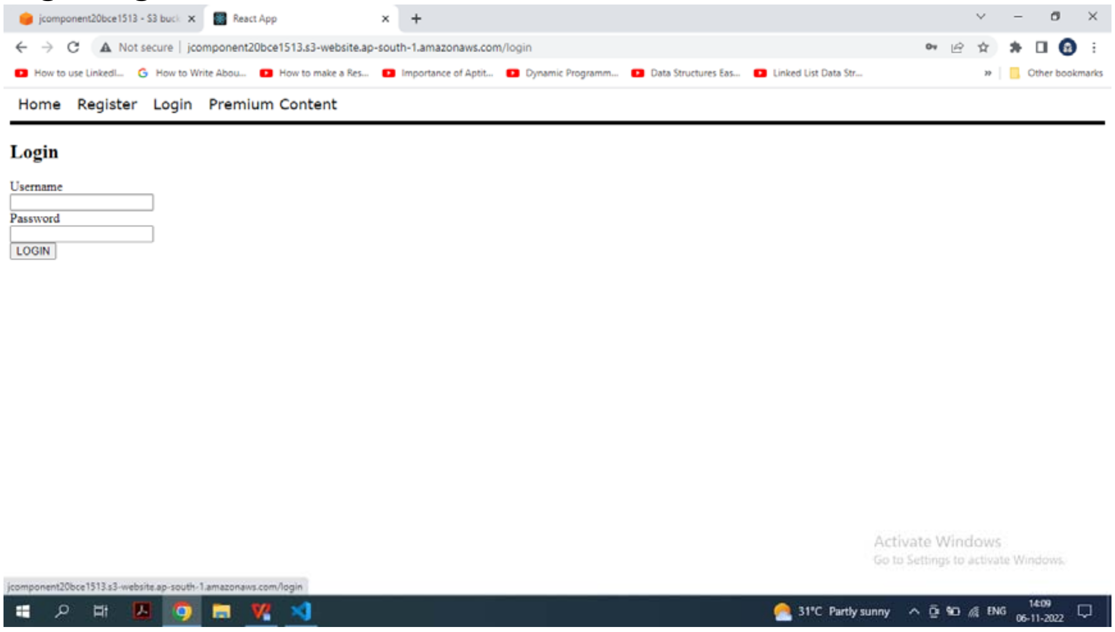
Email

Username

Password

REGISTER

Login Page:



Home Register Login Premium Content

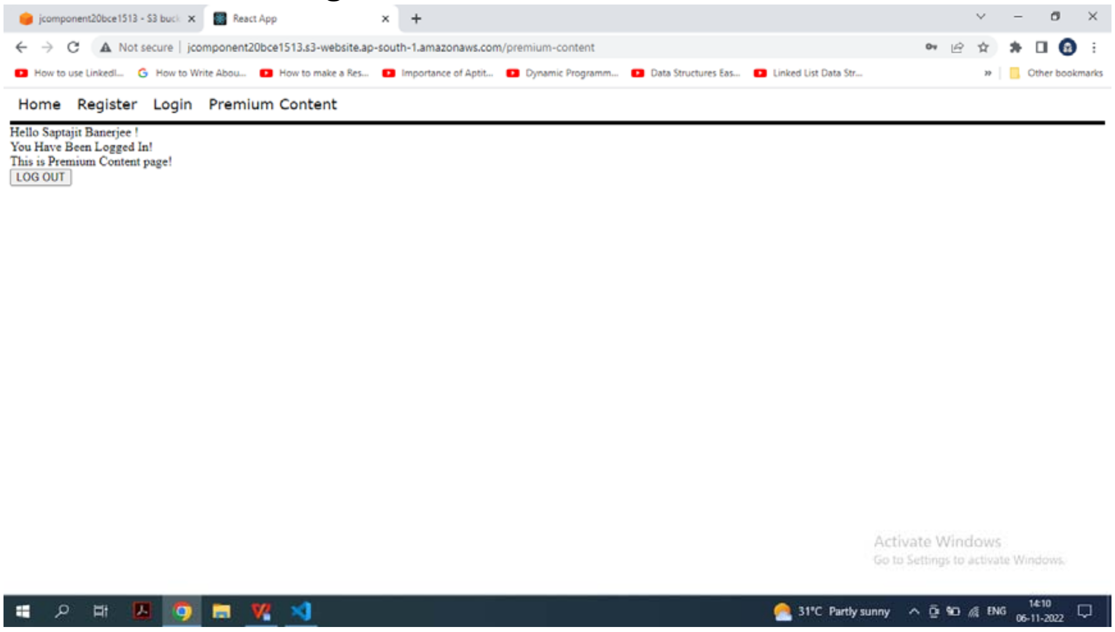
Login

Username

Password

LOGIN

Premium Content Page:

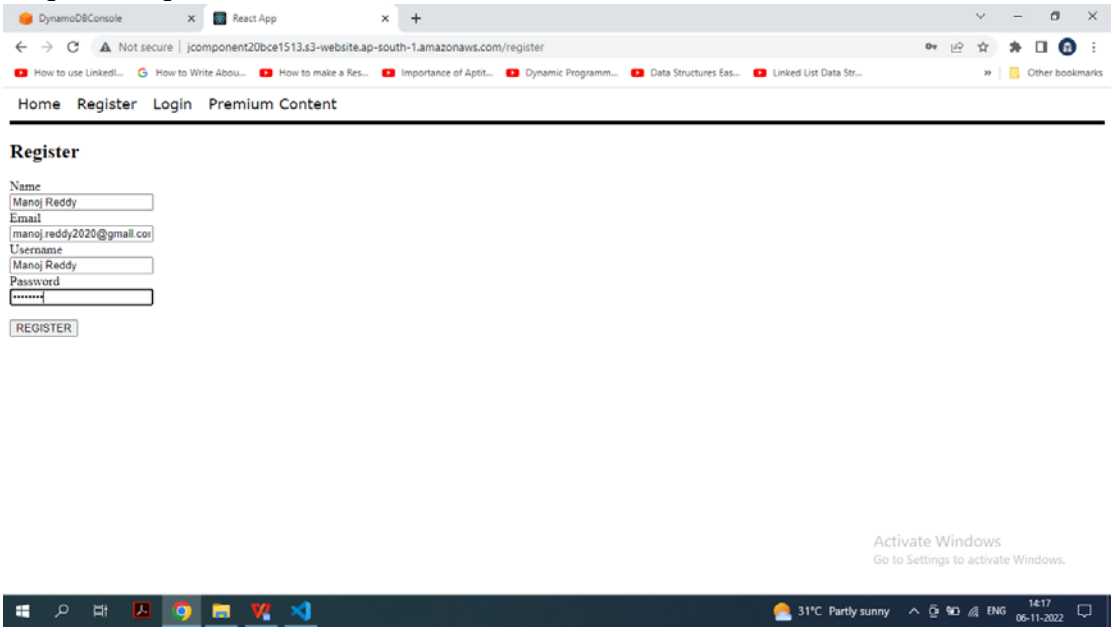


Home Register Login Premium Content

Hello Saptajit Banerjee !
You Have Been Logged In!
This is Premium Content page!

Demonstration:

Registering User:



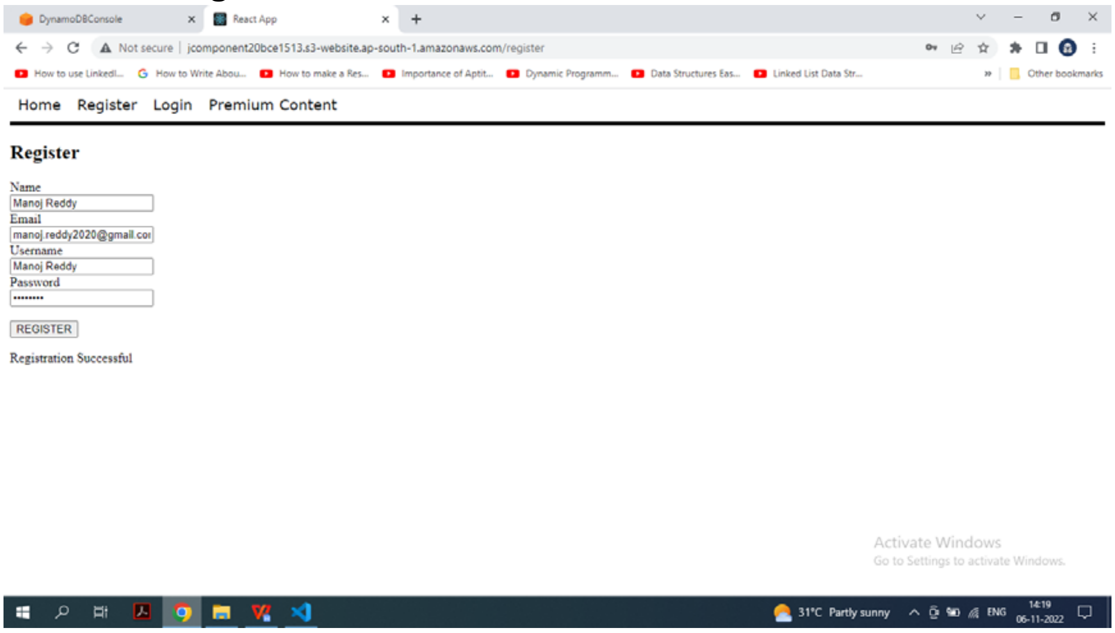
Home Register Login Premium Content

Register

Name: Manoj Reddy
Email: manoj.reddy2020@gmail.co
Username: Manoj Reddy
Password: ••••••••

Password is set as “abc1234”

Successful Registration:



jcomponent20bce1513.s3-website.ap-south-1.amazonaws.com/register

Home Register Login Premium Content

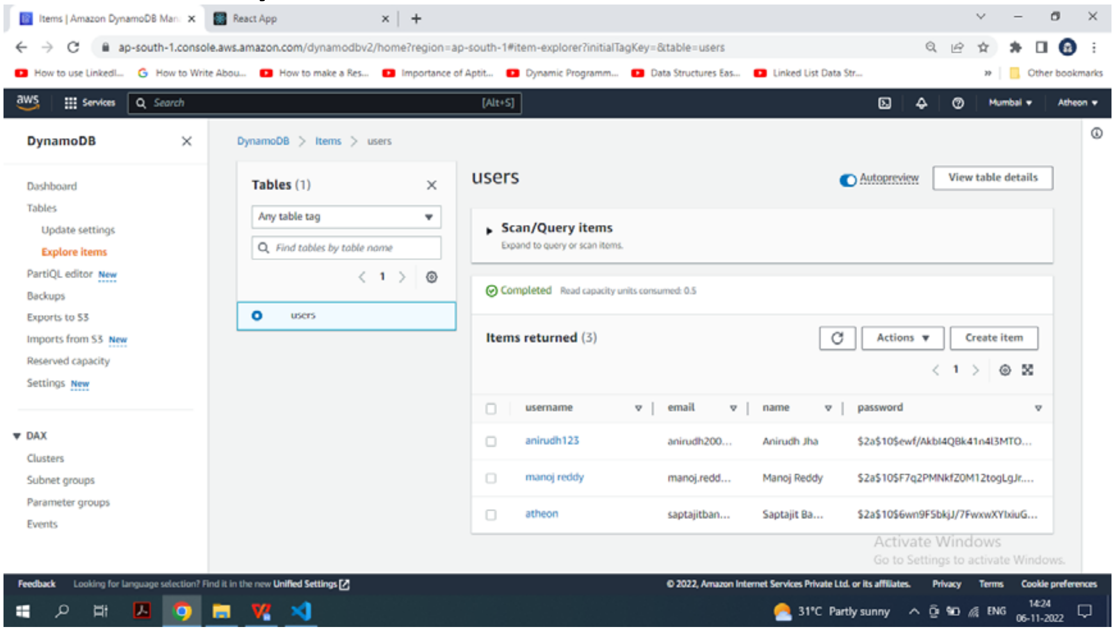
Register

Name: Manoj Reddy
Email: manoj.reddy2020@gmail.co
Username: Manoj Reddy
Password: ••••••••

REGISTER

Registration Successful

Data stored in DynamoDB:



ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#item-explorer?initialTagKey=&table=users

DynamoDB > Items > users

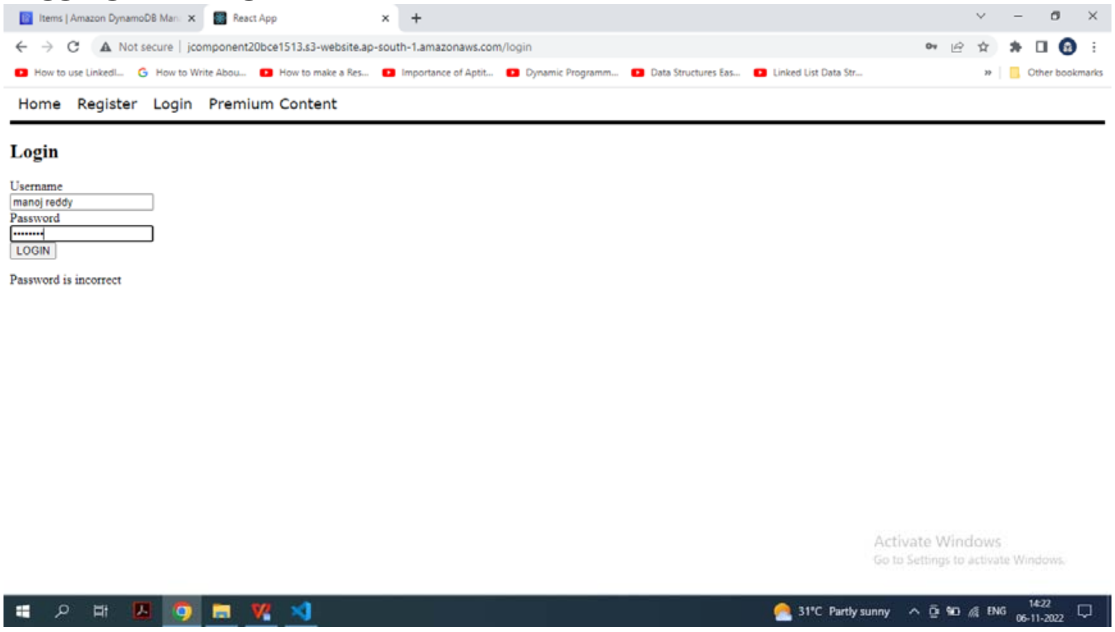
users

Completed Read capacity units consumed: 0.5

Items returned (3)

username	email	name	password
anirudh123	anirudh200...	Anirudh Jha	$2a$10$ewf/AkbI4QBk41n4l3MTO...
manoj reddy	manoj.redd...	Manoj Reddy	$2a$10$F7q2PMNkfZ0M12togLgJr....
atheon	saptajitban...	Saptajit Ba...	$2a$10$6wn9F5bkjJ/7FwxwXYIxiuG...

Logging into Registered Account:



Items | Amazon DynamoDB Man | React App

Not secure | jcomponent20bce1513.s3-website.ap-south-1.amazonaws.com/login

How to use LinkedI... | How to Write Abou... | How to make a Res... | Importance of Aptit... | Dynamic Programm... | Data Structures Eas... | Linked List Data Str... | Other bookmarks

Home Register Login Premium Content

Login

Username: manoj reddy

Password: ••••••••

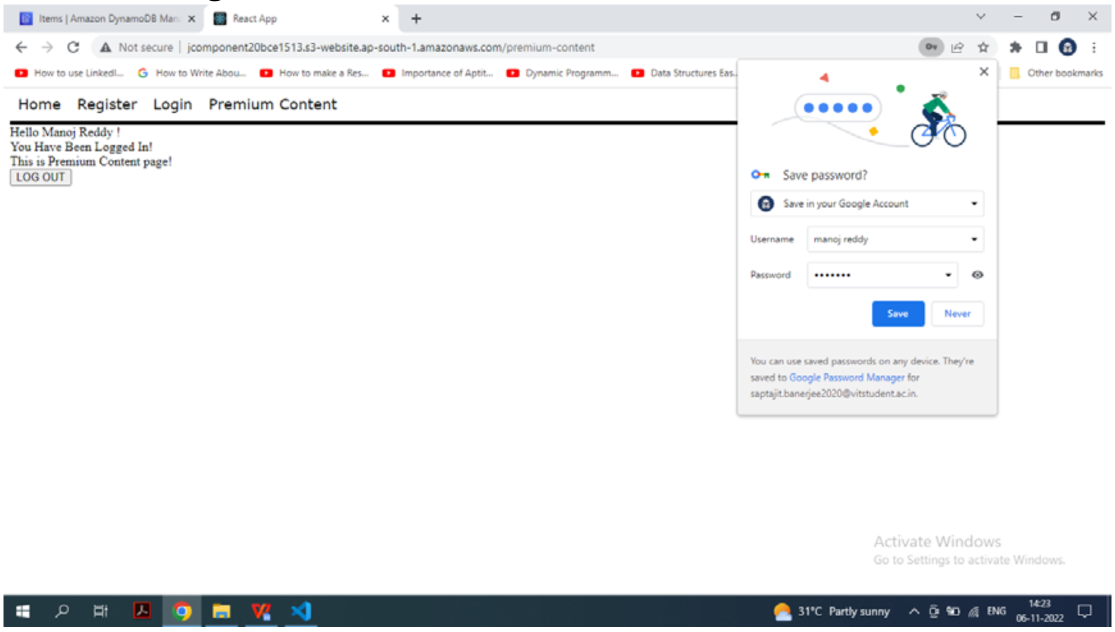
LOGIN

Password is incorrect

Activate Windows
Go to Settings to activate Windows.

31°C Partly sunny ENG 14:22 06-11-2022

Successful Login:



Items | Amazon DynamoDB Man | React App

Not secure | jcomponent20bce1513.s3-website.ap-south-1.amazonaws.com/premium-content

Home Register Login Premium Content

Hello Manoj Reddy !
You Have Been Logged In!
This is Premium Content page!
LOG OUT

Save password?

Save in your Google Account

Username: manoj reddy

Password: •••••••

Save | Never

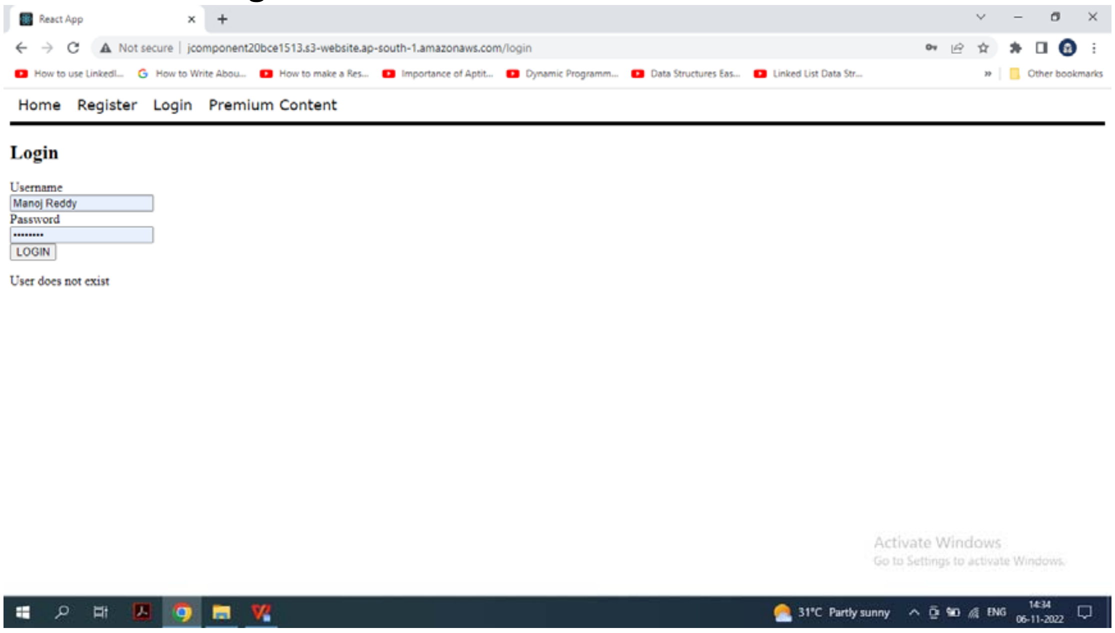
You can use saved passwords on any device. They're saved to Google Password Manager for saptajit.banerjee2020@vitstudent.ac.in.

Activate Windows
Go to Settings to activate Windows.

31°C Partly sunny ENG 14:23 06-11-2022

User will be automatically logged out after 1 minute if no user activity is detected.

Unsuccessful Logins:

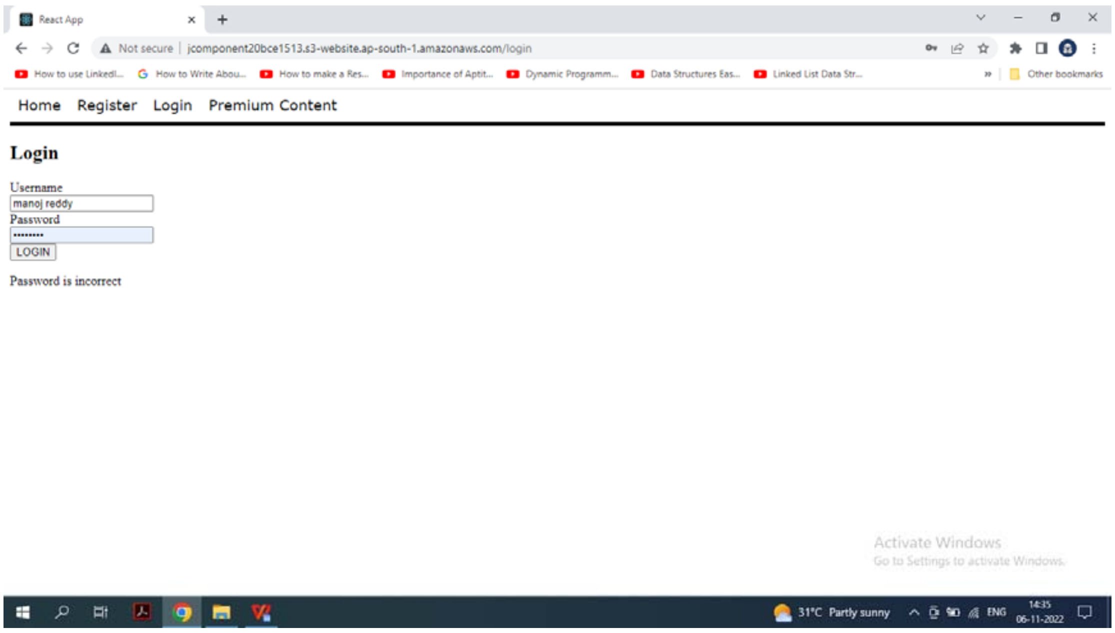

React App
Not secure | jcomponent20bce1513.s3-website.ap-south-1.amazonaws.com/login
Home Register Login Premium Content
LOGIN
Activate Windows
Go to Settings to activate Windows.
31°C Partly sunny
14:34
06-11-2022

Login

Username: Manoj Reddy

Password: ••••••••

User does not exist


React App
Not secure | jcomponent20bce1513.s3-website.ap-south-1.amazonaws.com/login
Home Register Login Premium Content
LOGIN
Activate Windows
Go to Settings to activate Windows.
31°C Partly sunny
14:35
06-11-2022

Login

Username: manoj reddy

Password: ••••••••

Password is incorrect

Advantages:

1. Back-up and restore data

Once the data is stored in the cloud, it is easier to get back-up and restore that data using the cloud.

2. Improved collaboration

Cloud applications improve collaboration by allowing groups of people to quickly and easily share information in the cloud via shared storage.

3. Excellent accessibility

Cloud allows us to quickly and easily access store information anywhere, anytime in the whole world, using an internet connection. An internet cloud infrastructure increases organization productivity and efficiency by ensuring that our data is always accessible.

4. Low maintenance cost
Cloud computing reduces both hardware and software maintenance costs for organizations.

5. Mobility
Cloud computing allows us to easily access all cloud data via mobile.

6. Intelligent Services in the pay-per-use model
Cloud computing offers Application Programming Interfaces (APIs) to the users for access services on the cloud and pays the charges as per the usage of service.

7. Unlimited storage capacity
Cloud offers us a huge amount of storing capacity for storing our important data such as documents, images, audio, video, etc. in one place.

8. Data security
Data security is one of the biggest advantages of cloud computing. Cloud offers many advanced features related to security and ensures that data is securely stored and handled.

9. Scalable
Scalability is one of the quality of AWS cloud in which we can easily scale up or scale down the space as per our requirements.

10. Quick updates are possible
By using serverless infrastructure or an architecture it is easy to deploy or update the web application. This property is generally used by the app developers.

Disadvantages:

1. Internet Connectivity
As we know, in cloud computing, every data (image, audio, video, etc.) is stored on the cloud, and we access these data through the cloud by using the internet connection. If you do not have good internet connectivity, you cannot access these data. However, we have no any other way to access data from the cloud.

2. Vendor lock-in
Vendor lock-in is the biggest disadvantage of cloud computing. Organizations may face problems when transferring their services from one vendor to another. As different vendors provide different platforms, that can cause difficulty moving from one cloud to another.

3. Limited Control
As we know, cloud infrastructure is completely owned, managed, and monitored by the service provider, so the cloud users have less control over the function and execution of services within a cloud infrastructure.

4. Security
Although cloud service providers implement the best security standards to store important information. But, before adopting cloud technology, you should be aware that you will be sending all your organization's sensitive information to a third party, i.e., a cloud computing service provider. While sending the data on the cloud, there may be a chance that your organization's information is hacked by Hackers.

5. Not for long-running processes
In AWS cloud platform the serverless architecture is not made for long term process or we can call as long running processes.

Conclusion:

Building serverless applications on AWS shows that the responsibilities that servers introduce. Using AWS Lambda as our serverless logic layer used to build faster and focus our development efforts on what differentiates the website. Lambda, AWS provides additional serverless capabilities so that we can build robust, reliable, secure, and cost-effective website. Understanding the capabilities and recommendations described in this paper can help to ensure the success when building serverless website of our own with authentication.

References:

- https://www.researchgate.net/publication/328765590_Analysis_of_Web_Authentication_Methods_Using_Amazon_Web_Services
- https://www.jetir.org/view?paper=JETIR2107029
- https://www.ripublication.com/ijaer18/ijaerv13n22_87.pdf
- https://www.mdpi.com/2073-431X/8/2/34/htm
- https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1149&context=msia_etds
- https://link.springer.com/article/10.1007/s40747-021-00305-0
- https://www.researchgate.net/publication/311530769_On_the_Network_Performance_of_Amazon_S3_Cloud-Storage_Service
- https://ijirt.org/Article?manuscript=151299
- https://journal.scsa.ge/papers/implementation-of-chatbot-using-aws-and-gupshup-api/
- https://ijcrt.org/papers/IJCRT2107116.pdf
- https://www.researchgate.net/publication/338391132_Case_Study_Use_of_AWS_Lambda_for_Building_a_Serverless_Chat_Application