Bahria University,

Karachi Campus



COURSE: Cloud Computing

TERM: FALL 2022, CLASS: BSE- 6 (B)

PROJECT NAME

Insurance Management System

|  |  |
| --- | --- |
| Name | Enrollment |
| Muhammad Amjad | 02-131202-041 |
| Jamshed Ali | 02-131202-008 |
| Abu Hurrerah | 02-131202-016 |
| Waleed Bin Tahir | 02-131202-034 |

Engr. Muhammad Faisal/ Engr. Ramsha Mashood

Signed Remarks: Score:

**TABLE OF CONTENT**

1. Introduction & problem\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3
2. Cloud technology\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3
3. Flow diagram\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4
4. Customize in current project\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7
5. Functionalities\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7
6. Module distribution\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8
7. Code\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8
8. Interfaces\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_22
9. Conclusion\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_26

**INTRODUCTION & PROBLEM:**

Our project focuses on developing an Insurance Management System that caters to the needs of three key stakeholders: the admin, the agent, and the client. The system aims to streamline and automate the entire insurance process, from policy creation and premium payments to claims management and policy activation. By doing so, we aim to overcome the challenges of inefficiency, delays, errors, and limited accessibility that are often associated with traditional methods of managing insurance policies. Ultimately, our goal is to enhance customer satisfaction, improve transparency, and facilitate effective communication between all parties involved.

**CLOUD TECHNOLOGY:**

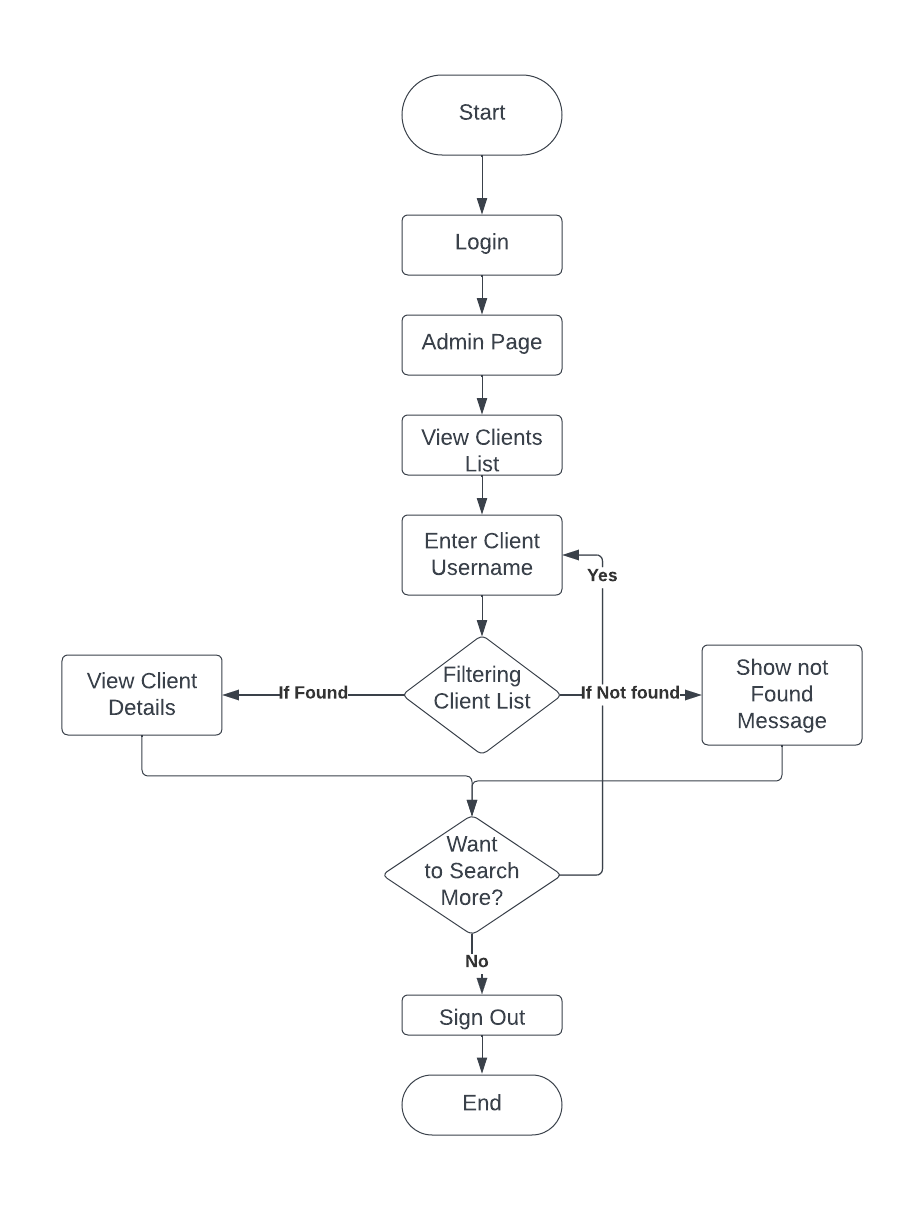
The cloud technologies used in your project are:

* **Netlify:** Netlify is a cloud-based hosting platform that provides infrastructure and services for hosting the frontend of your application. It offers features like automatic builds, continuous deployment, and CDN integration, enabling efficient deployment and management of static websites and web applications.
* **Cyclic.sh:** Cyclic.sh is a cloud-based API management platform that allows you to deploy, manage, and monitor your APIs. It provides capabilities such as API versioning, rate limiting, authentication, and analytics. Cyclic.sh ensures reliable access and efficient management of your APIs.
* **CloudCluster.io:** CloudCluster.io is a cloud hosting platform that specializes in providing SQL server databases. It offers managed SQL server instances with features like high availability, automatic backups, and scalability. CloudCluster.io simplifies the setup and management of your SQL server database, ensuring data reliability, security, and efficient performance.

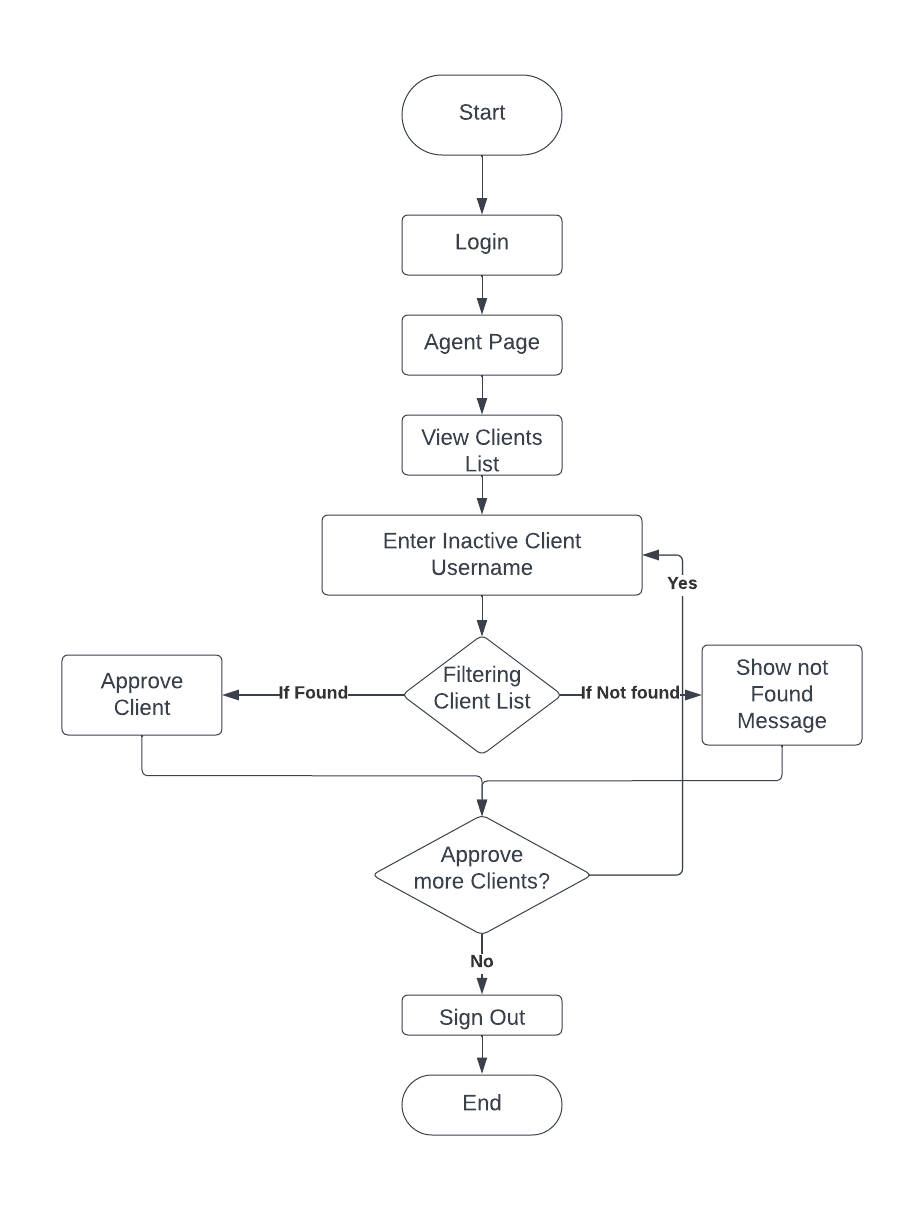
These cloud technologies collectively contribute to the infrastructure and services required for hosting, managing APIs, and maintaining a SQL server database in your insurance management system.

**FLOW DIAGRAM:**

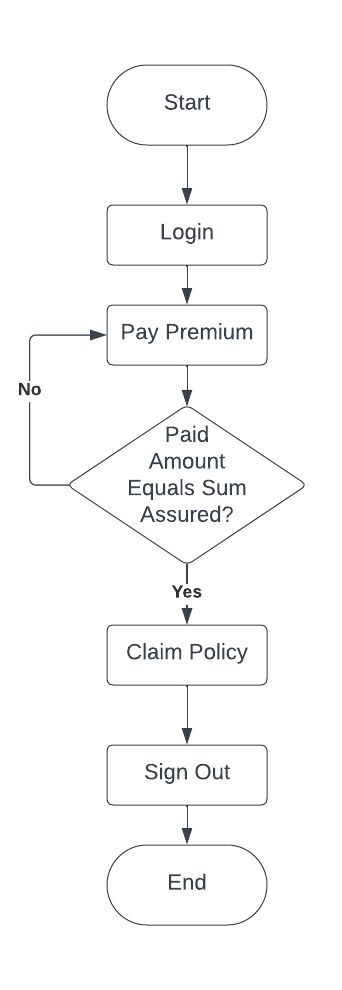
1. **Admin:**



1. **Agent:**



1. **Client:**



**CUSTOMIZE IN CURRENT PROJECT:**

Customizing the cloud technologies in the Insurance Management System involves configuring a custom domain and SSL certificate for Netlify, optimizing performance, implementing API security and rate limiting in Cyclic.sh, and configuring automated backup and scalability features in CloudCluster.io. These customizations enhance security, performance, and reliability in the system while catering to the specific needs of the project.

**FUNCTIONALITIES:**

1. **ADMIN:**

* **View client details:** The admin can access the system and view the details of specific clients by using his username.
* **Check policy status:** The admin can check the status of client policies, including inactive, active, and claimed policies.

1. **AGENT:**

* **View client details:** The agent can access the system and view the details of clients.
* **Check policy status:** The agent can check the status of client policies, including active and inactive policies.
* **Activate client policies:** The agent has the authority to activate policies for clients.

1. **CLIENT:**

* **Sign up for a policy:** The client can sign up for an insurance policy by providing necessary information.
* **Pay premium:** The client can make premium payments for their policy.
* **Claim a policy:** If needed, the client can file a claim for their policy.
* **Sum Assured and Paid Amount:** Client can view his sum assured and paid amount.

1. **OTHER FUNCTIONS:**

**Sum Assured Calculation:** The calculation of the policy will be automatically determined by the system based on the client's policy details, following the activation of the client's policy.

**Paid Amount Calculation:** The system tracks and calculates the total payments made by the client for their insurance policy

**MODULE DISTRIBUTION:**

* **Muhammad Amjad:** APIs Creation + Consumption
* **Jamshed Ali:** Frontend Creation
* **Abu Hurrerah:** Frontend Creation
* **Waleed Bin Tahir:** Database Design

**CODE:**

**API Code:**

const express = require('express');

const app = express();

const sql = require('mssql');

const bodyParser = require('body-parser');

const session = require('express-session');

const cors = require('cors');

const cookieParser = require('cookie-parser');

app.use(bodyParser.json());

app.use(cors())

app.use(bodyParser.urlencoded({ extended: true }));

app.use(cookieParser());

app.use(

session({

secret: '5azx', // Replace with your own secret key

resave: false,

saveUninitialized: false,

})

);

let checkingSession=''

// app.use((req, res, next) => {

// res.locals.username = req.session.username || null;

// next();

// });

//Database configuration

const dbConfig = {

user: 'Amjad',

password: 'Amjad@123',

server: 'mssql-132079-0.cloudclusters.net',

port:12097,

database: 'Insurance',

options: {

trustServerCertificate: true

}

};

// const dbConfig = {

// user: 'sa',

// password: 'amjad',

// server: 'DESKTOP-27TFO05',

// database: 'Insurance',

// options: {

// trustServerCertificate: true

// }

// };

// For Practice only

app.get('/users', (req, res) => {

sql.connect(dbConfig, (err) => {

if (err) {

console.log(err);

res.status(500).send('Database connection error');

} else {

const request = new sql.Request();

request.query('select \* from user\_details', (err, recordset) => {

if (err) {

console.log(err);

res.status(500).send('Error retrieving users from database');

} else {

res.send(recordset.recordset);

}

});

}

});

});

app.post('/users', (req, res) => {

const { Id, Firstname, Lastname, Email, Username, Password } = req.body;

sql.connect(dbConfig, (err) => {

if (err) {

console.error('Database connection error:', err);

return res.status(500).send('Database connection error');

}

const request = new sql.Request();

const query = `

INSERT INTO Users (Id, Firstname, Lastname, Email, Username, Password)

VALUES (${Id}, '${Firstname}', '${Lastname}', '${Email}', '${Username}', '${Password}')

`;

request.query(query, (err) => {

if (err) {

console.error('Error inserting data:', err);

return res.status(500).send('Error inserting data');

}

res.status(201).send('Data inserted successfully');

});

});

});

app.get('/users/:id', (req, res) => {

const userId = req.params.id;

sql.connect(dbConfig, (err) => {

if (err) {

console.log(err);

res.status(500).send('Database connection error');

} else {

const request = new sql.Request();

request.query(`SELECT \* FROM Users WHERE Id = ${userId}`, (err, recordset) => {

if (err) {

console.log(err);

res.status(500).send('Error retrieving user from database');

} else if (recordset.recordset.length === 0) {

res.status(404).send('User not found');

} else {

res.send(recordset.recordset[0]);

}

});

}

});

});

//Project Post

app.post('/signup', (req, res) => {

const { nametxt, usernametxt, passtxt, phonetxt, dtext, citytxt, addresstxt, nomineetxt,selected\_policy } = req.body;

// Data validation

if (!nametxt || !usernametxt || !passtxt || !phonetxt || !dtext || !citytxt || !addresstxt || !nomineetxt||!selected\_policy) {

return res.status(400).json({ error: 'Missing required fields' });

}

// Connect to the database

const con = new sql.ConnectionPool(dbConfig);

con.connect(err => {

if (err) {

console.log(err);

return res.status(500).json({ error: 'Database connection error' });

}

// Perform data validation

con.query(`SELECT username FROM user\_details WHERE username = '${usernametxt}'`, (err, result) => {

if (err) {

console.log(err);

con.close();

return res.status(500).json({ error: 'Error checking existing username' });

}

console.log(result.recordset[0]==undefined)

// console.log(usernametxt==result.recordset[0].username)

// console.log(result.recordset[0].username);

if (result.recordset[0]!=undefined) {

con.close();

console.log("User exists")

return res.json({ message: 'User exists' });

}

// Execute SQL queries

con.query(`INSERT INTO user\_details (name, username, password, phone\_no, dob) VALUES ('${nametxt}', '${usernametxt}', '${passtxt}', '${phonetxt}', '${dtext}')`, (err, result) => {

if (err) {

console.log(err);

con.close();

return res.status(500).json({ error: 'Error inserting user details' });

}

con.query(`INSERT INTO address\_details (home\_address, city) VALUES ('${addresstxt}', '${citytxt}')`, (err, result) => {

if (err) {

console.log(err);

con.close();

return res.status(500).json({ error: 'Error inserting address details' });

}

con.query(`INSERT INTO Nominee\_Detail (nominee\_name) VALUES ('${nomineetxt}')`, (err, result) => {

if (err) {

console.log(err);

con.close();

return res.status(500).json({ error: 'Error inserting nominee details' });

}

con.query('SELECT MAX(nominee\_id) AS max\_nominee\_id FROM Nominee\_Detail', (err, result) => {

if (err) {

console.log(err);

con.close();

return res.status(500).json({ error: 'Error retrieving max nominee ID' });

}

console.log(result)

const nomid = result.recordset[0].max\_nominee\_id

console.log(nomid)

con.query('SELECT MAX(address\_id) AS max\_address\_id FROM address\_details', (err, result) => {

if (err) {

console.log(err);

con.close();

return res.status(500).json({ error: 'Error retrieving max address ID' });

}

const addit = result.recordset[0].max\_address\_id;

con.query('SELECT MAX(user\_id) AS max\_user\_id FROM user\_details', (err, result) => {

if (err) {

console.log(err);

con.close();

return res.status(500).json({ error: 'Error retrieving max user ID' });

}

const uid = result.recordset[0].max\_user\_id;

const addingIdsQuery = `Update user\_details set nominee\_id=${nomid},address\_id=${addit} where user\_id=${uid}`

con.query(addingIdsQuery, (err, result) => {

if (err) {

console.log(err);

con.close();

return res.status(500).json({ error: 'Error inserting IDs into user\_details' });

}

const dt = new Date().toISOString().split('T')[0];

const policyTypeQuery = `SELECT policy\_type\_code FROM ref\_policy\_types WHERE policy\_type\_name = '${selected\_policy}'`;

con.query(policyTypeQuery, (err, result) => {

if (err) {

console.log(err);

con.close();

return res.status(500).json({ error: 'Error retrieving policy type code' });

}

const policyCode = result.recordset[0].policy\_type\_code;

const userPolicyQuery = `INSERT INTO user\_policies (user\_id, nominee\_id, date\_registered, policy\_type\_id, policy\_status) VALUES (${uid}, ${nomid}, '${dt}', '${policyCode}', 'Inactive')`;

con.query(userPolicyQuery, (err, result) => {

if (err) {

console.log(err);

con.close();

return res.status(500).json({ error: 'Error inserting user policy details' });

}

// Close the database connection

con.close();

// Redirect the user or send a response

return res.json({ message: 'done' });

});

});

});

});

});

});

});

});

});

});

});

});

//sign in

app.post('/signin', async (req, res) => {

const { usernametxt, passtxt } = req.body;

if (usernametxt === '' || passtxt === '') {

let errors = [];

if (usernametxt === '') {

errors.push({ field: 'username', message: 'Username is required' });

}

if (passtxt === '') {

errors.push({ field: 'password', message: 'Password is required' });

}

return res.status(400).json({ errors });

}

if (usernametxt.toLowerCase() === 'agent') {

req.session.username = usernametxt;

checkingSession=req.session.username

console.log('Checking: '+req.session.username);

return res.json({ message: 'agent' });

} else if (usernametxt.toLowerCase() === 'admin') {

req.session.username = usernametxt;

checkingSession=req.session.username

console.log('Checking: '+req.session.username);

return res.json({ message: 'admin' });

} else {

try {

const pool = await sql.connect(dbConfig);

const query = `SELECT username, Password FROM user\_details WHERE username = '${usernametxt}' AND password = '${passtxt}'`;

const result = await pool.request().query(query);

sql.close();

if (result.recordset.length > 0) {

req.session.username = usernametxt;

checkingSession=req.session.username

return res.json({ message: 'user' });

} else {

return res.status(401).json({ error: 'Data does not match' });

}

} catch (err) {

console.log(err);

return res.status(500).json({ error: 'Database error' });

}

}

});

//sign out

app.post('/sign-out', async (req, res) => {

try {

await new Promise((resolve, reject) => {

req.session.destroy((err) => {

if (err) {

reject(err);

} else {

console.log('destroyed');

checkingSession=''

resolve();

}

});

});

res.redirect('/login');

} catch (err) {

console.log(err);

res.status(500).json({ error: 'Failed to sign out' });

}

});

//Policy Detail

app.get('/user-policy', async (req, res) => {

try {

const pool = new sql.ConnectionPool(dbConfig);

const connection = await pool.connect();

const uidQuery = `SELECT user\_id FROM user\_details WHERE username = '${checkingSession}'`;

const uidResult = await connection.query(uidQuery);

const uid = uidResult.recordset[0]?.user\_id;

const polNameQuery = `SELECT ref\_policy\_types.policy\_type\_name FROM user\_details

INNER JOIN user\_policies ON user\_details.user\_id = user\_policies.user\_id

INNER JOIN ref\_policy\_types ON ref\_policy\_types.policy\_type\_code = user\_policies.policy\_type\_id

WHERE user\_details.username = '${checkingSession}'`;

const preAmountQuery = `SELECT premium\_amount FROM user\_details WHERE username = '${checkingSession}'`;

const sumQuery = `SELECT Sum\_Assured FROM user\_details WHERE username = '${checkingSession}'`;

const statQuery = `SELECT user\_policies.policy\_status FROM user\_policies

INNER JOIN user\_details ON user\_details.user\_id = user\_policies.user\_id

WHERE user\_details.username = '${checkingSession}'`;

const payQuery = `SELECT SUM(policy\_payments.amount) as totalPayment FROM policy\_payments

INNER JOIN user\_details ON policy\_payments.user\_id = user\_details.user\_id

WHERE policy\_payments.user\_id = ${uid}`;

const [polNameResult, preAmountResult, sumResult, statResult, payResult] = await Promise.all([

connection.query(polNameQuery),

connection.query(preAmountQuery),

connection.query(sumQuery),

connection.query(statQuery),

connection.query(payQuery)

]);

const policyName = polNameResult.recordset[0]?.policy\_type\_name || '';

const premiumAmount = preAmountResult.recordset[0]?.premium\_amount || '';

const sumAssuredAmount = sumResult.recordset[0]?.Sum\_Assured || '';

const policyStatus = statResult.recordset[0]?.policy\_status || '';

const payments = payResult.recordset[0]?.totalPayment || '0';

let checkUserID;

for (let i = 1; i <= 2; i++) {

if (i === 2) {

const payQuery = `SELECT user\_id FROM Claim\_Detials WHERE user\_id = ${uid}`;

checkUserID = await connection.query(payQuery);

}

}

const checkUID = checkUserID.recordset[0]?.user\_id;

const response = {

policyName,

premiumAmount: premiumAmount || 'Wait for policy Approval',

sumAssuredAmount: sumAssuredAmount || 'Wait for policy Approval',

payments,

policyStatus,

showButton1: premiumAmount !== ''&&payments<sumAssuredAmount,

showButton2: sumAssuredAmount === payments&&policyStatus==='Active',

showLabel3: policyStatus==='Claimed',

};

connection.release();

res.json(response);

} catch (err) {

console.error(err);

res.status(500).json({ error: 'Database error' });

}

});

//Paying Premium

app.post('/pay-premium', async (req, res) => {

try {

const connection = new sql.ConnectionPool(dbConfig);

await connection.connect();

const uidQuery = `SELECT user\_id FROM user\_details WHERE username = '${checkingSession}'`;

const uidResult = await connection.query(uidQuery);

const uid = uidResult.recordset[0]?.user\_id;

const pAmountQuery = `SELECT premium\_amount FROM user\_details WHERE username = '${checkingSession}'`;

const pNoQuery = `SELECT policy\_no FROM user\_policies WHERE user\_id = ${uid}`;

const [pAmountResult, pNoResult] = await Promise.all([

connection.query(pAmountQuery),

connection.query(pNoQuery)

]);

const pAmount = pAmountResult.recordset[0]?.premium\_amount || 0;

const pNo = pNoResult.recordset[0]?.policy\_no || 0;

const payQuery = `INSERT INTO policy\_payments (user\_id, policy\_no, amount, date\_of\_payment)

VALUES (${uid}, ${pNo}, ${pAmount}, GETDATE())`;

await connection.query(payQuery);

connection.close()

res.json({message:"done"})

} catch (err) {

console.error(err);

res.status(500).json({ error: 'Database error' });

}

});

app.post('/claim-policy', async (req, res) => {

try {

const connection = new sql.ConnectionPool(dbConfig);

// Connect to the database

await connection.connect();

const uidQuery = `SELECT user\_id FROM user\_details WHERE username = '${checkingSession}'`;

const nomIdQuery = `SELECT nominee\_id FROM user\_details WHERE username = '${checkingSession}'`;

const preAmountQuery = `SELECT Sum\_Assured FROM user\_details WHERE username = '${checkingSession}'`;

const [uidResult, nomIdResult, preAmountResult] = await Promise.all([

connection.query(uidQuery),

connection.query(nomIdQuery),

connection.query(preAmountQuery),

]);

const uid = uidResult.recordset.length > 0 ? uidResult.recordset[0].user\_id : null;

const nomId = nomIdResult.recordset[0]?.nominee\_id;

const preAmount = preAmountResult.recordset[0]?.Sum\_Assured || 0;

const payQuery = `SELECT SUM(policy\_payments.amount) AS total\_paid FROM policy\_payments INNER JOIN user\_details ON policy\_payments.user\_id = user\_details.user\_id WHERE policy\_payments.user\_id = ${uid}`;

const payResult = await connection.query(payQuery);

const totalPaid = payResult.recordset[0]?.total\_paid || 0;

const claimInsertQuery = `INSERT INTO Claim\_Detials (user\_id, nominee\_id, claim\_by, claim\_amount) VALUES (${uid}, ${nomId}, 'amjad53azx', ${totalPaid})`;

const updateQuery = `UPDATE user\_policies SET policy\_status = 'Claimed' WHERE user\_id = ${uid}`;

await connection.query(updateQuery);

await connection.query(claimInsertQuery);

connection.close();

res.json({ message: 'Policy claimed' });

} catch (err) {

console.error(err);

res.status(500).json({ error: 'Database error' });

}

});

//Profile

app.get('/profile', async (req, res) => {

try {

const nametxt = `SELECT name FROM user\_details WHERE username='${checkingSession}'`;

const addtxt = `SELECT address\_details.home\_address FROM address\_details INNER JOIN user\_details ON address\_details.address\_id=user\_details.address\_id WHERE user\_details.username='${checkingSession}'`;

const nomtxt = `SELECT Nominee\_Detail.nominee\_name FROM Nominee\_Detail INNER JOIN user\_details ON Nominee\_Detail.nominee\_id=user\_details.nominee\_id WHERE user\_details.username='${checkingSession}'`;

const phonetxt = `SELECT phone\_no FROM user\_details WHERE username='${checkingSession}'`;

const \_password = `SELECT password FROM user\_details WHERE username='${checkingSession}'`;

const city\_name = `SELECT address\_details.city FROM address\_details INNER JOIN user\_details ON address\_details.address\_id=user\_details.address\_id WHERE user\_details.username='${checkingSession}'`;

const connection = await sql.connect(dbConfig);

const [nameResult, addressResult, nomineeResult, phoneResult, passwordResult, cityResult] = await Promise.all([

connection.query(nametxt),

connection.query(addtxt),

connection.query(nomtxt),

connection.query(phonetxt),

connection.query(\_password),

connection.query(city\_name),

]);

sql.close();

const name = nameResult.recordset[0]?.name || '';

const address = addressResult.recordset[0]?.home\_address || '';

const nominee = nomineeResult.recordset[0]?.nominee\_name || '';

const phone = phoneResult.recordset[0]?.phone\_no || '';

const password = passwordResult.recordset[0]?.password || '';

const city = cityResult.recordset[0]?.city || '';

console.log(city);

return res.json({ name, address, nominee, phone, password, city });

} catch (err) {

console.log(err);

return res.status(500).json({ error: 'Database error' });

}

});

//Update Phone

app.put('/update-phone', (req, res) => {

const { phone } = req.body;

if (!phone) {

return res.status(400).json({ error: 'Phone number is required' });

}

const updateQuery = `UPDATE user\_details SET phone\_no = ${phone} WHERE username = '${checkingSession}'`;

sql.connect(dbConfig)

.then((pool) => pool.request().query(updateQuery))

.then(() => {

sql.close();

return res.send('Phone number updated successfully');

})

.catch((err) => {

console.log(err);

return res.status(500).json({ error: 'Database error' });

});

});

//Update Name

app.put('/update-name', async (req, res) => {

const { name } = req.body;

if (!name) {

return res.status(400).json({ error: 'Name is required' });

}

try {

const pool = await sql.connect(dbConfig);

const updateQuery = `UPDATE user\_details SET name = '${name}' WHERE username = '${checkingSession}'`;

await pool.request().query(updateQuery);

sql.close();

return res.send('Name updated successfully');

} catch (err) {

console.log(err);

return res.status(500).json({ error: 'Database error' });

}

});

//Update Password

app.put('/update-password', async (req, res) => {

const { password } = req.body;

if (!password) {

return res.status(400).json({ error: 'Password is required' });

}

try {

const pool = await sql.connect(dbConfig);

const updateQuery = `UPDATE user\_details SET password = '${password}' WHERE username = '${checkingSession}'`;

await pool.request().query(updateQuery);

sql.close();

return res.send('Password updated successfully');

} catch (err) {

console.log(err);

return res.status(500).json({ error: 'Database error' });

}

});

//Update City

app.put('/update-city', async (req, res) => {

const { city } = req.body;

if (!city) {

return res.status(400).json({ error: 'City is required' });

}

try {

const pool = await sql.connect(dbConfig);

const getUserIdQuery = `SELECT address\_id FROM user\_details WHERE username = '${checkingSession}'`;

const result = await pool.request().query(getUserIdQuery);

const addressId = result.recordset[0].address\_id;

const updateQuery = `UPDATE address\_details SET city = '${city}' WHERE address\_id = ${addressId}`;

await pool.request().query(updateQuery);

sql.close();

return res.send('City updated successfully');

} catch (err) {

console.log(err);

return res.status(500).json({ error: 'Database error' });

}

});

//Update Address

app.put('/update-address', async (req, res) => {

const { address } = req.body;

if (!address) {

return res.status(400).json({ error: 'Address is required' });

}

try {

const pool = await sql.connect(dbConfig);

const getUserIdQuery = `SELECT address\_id FROM user\_details WHERE username = '${checkingSession}'`;

const result = await pool.request().query(getUserIdQuery);

const addressId = result.recordset[0].address\_id;

const updateQuery = `UPDATE address\_details SET home\_address = '${address}' WHERE address\_id = ${addressId}`;

await pool.request().query(updateQuery);

sql.close();

return res.send('Address updated successfully');

} catch (err) {

console.log(err);

return res.status(500).json({ error: 'Database error' });

}

});

//Update Nominee

app.put('/update-nominee', async (req, res) => {

try {

const username = checkingSession;

const nomineeName = req.body.nominee;

if (!nomineeName) {

return res.status(400).json({ error: 'Nominee name is required' });

}

const connection = new sql.ConnectionPool(dbConfig);

// Connect to the database

await connection.connect();

const getNomineeIdQuery = `SELECT nominee\_id FROM user\_details WHERE username = '${username}'`;

const nomineeIdResult = await connection.query(getNomineeIdQuery);

const nomineeId = nomineeIdResult.recordset[0]?.nominee\_id;

if (!nomineeId) {

return res.status(404).json({ error: 'Nominee not found' });

}

const updateQuery = `UPDATE Nominee\_Detail SET nominee\_name = '${nomineeName}' WHERE nominee\_id = ${nomineeId}`;

await connection.query(updateQuery);

connection.close();

res.json({ message: 'Nominee name updated successfully' });

} catch (err) {

console.error(err);

res.status(500).json({ error: 'Database error' });

}

});

//Premium page

app.get('/premium-profile', async (req, res) => {

try {

if (!req.session.username) {

return res.redirect('Login.aspx');

}

const uidQuery = `SELECT user\_id FROM user\_details WHERE username = '${req.session.username}'`;

const polNameQuery = `SELECT ref\_policy\_types.policy\_type\_name FROM user\_details INNER JOIN user\_policies ON user\_details.user\_id = user\_policies.user\_id INNER JOIN ref\_policy\_types ON ref\_policy\_types.policy\_type\_code = user\_policies.policy\_type\_id WHERE user\_details.username = '${req.session.username}'`;

const preAmountQuery = `SELECT premium\_amount FROM user\_details WHERE username = '${req.session.username}'`;

const sumQuery = `SELECT Sum\_Assured FROM user\_details WHERE username = '${req.session.username}'`;

const statQuery = `SELECT user\_policies.policy\_status FROM user\_policies INNER JOIN user\_details ON user\_details.user\_id = user\_policies.user\_id WHERE user\_details.username = '${req.session.username}'`;

const paymentsQuery = `SELECT SUM(policy\_payments.amount) FROM policy\_payments INNER JOIN user\_details ON policy\_payments.user\_id = user\_details.user\_id WHERE policy\_payments.user\_id = (SELECT user\_id FROM user\_details WHERE username = '${req.session.username}')`;

const pool = await sql.connect(dbConfig);

const uidResult = await pool.request().query(uidQuery);

const uid = uidResult.recordset[0].user\_id;

const polNameResult = await pool.request().query(polNameQuery);

const policy\_name = polNameResult.recordset[0].policy\_type\_name;

const preAmountResult = await pool.request().query(preAmountQuery);

const premium\_amount = preAmountResult.recordset[0].premium\_amount || '';

const sumResult = await pool.request().query(sumQuery);

const sum\_assured\_amount = sumResult.recordset[0].Sum\_Assured || '';

const statResult = await pool.request().query(statQuery);

const pstatus = statResult.recordset[0].policy\_status;

let payments = '';

const paymentsResult = await pool.request().query(paymentsQuery);

payments = paymentsResult.recordset[0][''] || '0';

if (premium\_amount === '') {

return res.render('Wait for policy Approval');

}

const checkUidQuery = `SELECT user\_id FROM Claim\_Detials WHERE user\_id = ${uid}`;

const checkUidResult = await pool.request().query(checkUidQuery);

const check\_uid = checkUidResult.recordset[0].user\_id || 0;

const renderData = {

policy\_name,

premium\_amount,

sum\_assured\_amount,

payments,

pstatus,

showButton1: true,

showButton2: false,

showLabel3: false,

};

if (sum\_assured\_amount === payments) {

renderData.showButton1 = false;

if (check\_uid === uid) {

renderData.showLabel3 = true;

renderData.showButton2 = false;

} else {

renderData.showLabel3 = false;

renderData.showButton2 = true;

}

}

sql.close();

return res.render('Profile.aspx', renderData);

} catch (err) {

console.log(err);

return res.status(500).json({ error: 'Database error' });

}

});

//Inactive user policy status detail

app.get('/user-policy-status-details', async (req, res) => {

try {

await sql.connect(dbConfig);

const result = await sql.query('EXEC User\_Policy\_Status\_Details;');

sql.close();

const data = result.recordset;

return res.json(data);

} catch (err) {

console.log(err);

return res.status(500).json({ error: 'Database error' });

}

});

//Active User policy status

app.get('/active-users-policies-details', async (req, res) => {

try {

await sql.connect(dbConfig);

const result = await sql.query('EXEC Active\_Users\_Policies\_Details;');

sql.close();

const data = result.recordset;

return res.json(data);

} catch (err) {

console.log(err);

return res.status(500).json({ error: 'Database error' });

}

});

//Claimed user policy

app.get('/claimed-user-policies', async (req, res) => {

try {

await sql.connect(dbConfig);

const result = await sql.query('EXEC Claimed\_User\_Policies;');

sql.close();

const data = result.recordset;

return res.json(data);

} catch (err) {

console.log(err);

return res.status(500).json({ error: 'Database error' });

}

});

//Approve policy

app.put('/approve-policy', async (req, res) => {

try {

const { username } = req.body;

if (!username) {

return res.status(400).json({ error: 'Username is required' });

}

const getUsernameQuery = `SELECT username FROM user\_details WHERE username = '${username}'`;

const pool = await sql.connect(dbConfig);

const result = await pool.request().query(getUsernameQuery);

const uname = result.recordset[0].username;

if (uname === username) {

const getUserIdQuery = `EXEC Fetching\_UserId\_For\_Activation '${username}'`;

const getPolicyNameQuery = `EXEC Fetching\_PolicyName\_For\_Activation '${username}'`;

const uidResult = await pool.request().query(getUserIdQuery);

const uid = uidResult.recordset[0][''];

const policyNameResult = await pool.request().query(getPolicyNameQuery);

const policyName = policyNameResult.recordset[0][''];

const costCalculationQuery = `EXEC CostCalculation '${policyName}', ${uid}`;

await pool.request().query(costCalculationQuery);

sql.close();

return res.send('Policy approved successfully');

} else {

return res.status(404).json({ error: 'Username not found' });

}

} catch (err) {

console.log(err);

return res.status(500).json({ error: 'Database error' });

}

});

//Check Details

app.post('/checkDetails', async (req, res) => {

const usertxt = req.body.usertxt;

try {

console.log(usertxt);

const pool = await sql.connect(dbConfig);

const getData = `select user\_id from user\_details where username='${usertxt}'`;

const resultUserId = await pool.request().query(getData);

if (resultUserId.recordset[0]!==undefined) {

console.log("User");

const uid = resultUserId.recordset[0].user\_id;

let pol\_name = `select ref\_policy\_types.policy\_type\_name from user\_details inner join user\_policies on user\_details.user\_id=user\_policies.user\_id inner join ref\_policy\_types on ref\_policy\_types.policy\_type\_code=user\_policies.policy\_type\_id where user\_details.username='${usertxt}'`;

let pre\_amount = `select premium\_amount from user\_details where username='${usertxt}'`;

let sum = `select Sum\_Assured from user\_details where username='${usertxt}'`;

let stat = `select user\_policies.policy\_status from user\_policies inner join user\_details on user\_details.user\_id=user\_policies.user\_id where user\_details.username='${usertxt}'`;

const res\_pol\_name = await pool.request().query(pol\_name);

const res\_pre\_amount = await pool.request().query(pre\_amount);

const res\_sum = await pool.request().query(sum);

const res\_stat = await pool.request().query(stat);

let res\_pay = '';

if (res\_stat.recordset[0].policy\_status === 'Inactive') {

res\_sum.recordset[0].Sum\_Assured = 'Waiting for Policy Approval';

res\_pre\_amount.recordset[0].premium\_amount = 'Waiting for Policy Approval';

res\_pay = 'Waiting for Policy Approval';

} else {

const pay = `select sum(policy\_payments.amount) as paid\_amount from policy\_payments inner join user\_details on policy\_payments.user\_id=user\_details.user\_id where policy\_payments.user\_id=${uid}`;

const res\_pay\_result = await pool.request().query(pay);

if(res\_pay==="")

res\_pay = "0";

}

const policyDetails = {

'Policy Name': res\_pol\_name.recordset[0].policy\_type\_name,

'Policy Status': res\_stat.recordset[0].policy\_status,

'Premium Amount': res\_pre\_amount.recordset[0].premium\_amount,

'Sum Assured': res\_sum.recordset[0].Sum\_Assured,

'Paid Amount': res\_pay

};

return res.json(policyDetails);

} else if (resultUserId.recordset[0]===undefined) {

console.log('No user');

return res.json({ message: 'No User exists' });

}

} catch (error) {

console.log(error);

return res.status(500).json({ error: 'Database error' });

}

});

//changing g=here

app.get('/check-session', (req, res) => {

console.log("Session: "+checkingSession);

console.log("Session Object: "+JSON.stringify(req.session));

if (checkingSession) {

res.json({ sessionValid: true});

} else {

res.json({ sessionValid: false });

}

});

//policy activation

app.post('/activate-policy', async (req, res) => {

const { usertxt } = req.body;

const getUsernameQuery = `select user\_details.username as username from user\_details inner join user\_policies on user\_details.user\_id=user\_policies.user\_id where user\_policies.policy\_status='Inactive' and user\_details.username='${usertxt}'`;

try {

await sql.connect(dbConfig);

const result = await sql.query(getUsernameQuery);

console.log(result);

if(result.recordset[0]===undefined){

return res.json({message:"no user found"})

}

const username = result.recordset[0]['username'];

console.log(username);

if (username) {

const getPolicyNameQuery = `select ref\_policy\_types.policy\_type\_name as policy\_name from user\_details inner join user\_policies on user\_details.user\_id=user\_policies.user\_id inner join ref\_policy\_types on user\_policies.policy\_type\_id=ref\_policy\_types.policy\_type\_code where user\_details.username='${usertxt}'`;

const policyNameResult = await sql.query(getPolicyNameQuery);

const policyName = policyNameResult.recordset[0]['policy\_name'];

console.log(policyName);

const getUserIdQuery = `SELECT user\_id FROM user\_details WHERE username = '${username}'`;

const result = await sql.query(getUserIdQuery);

const userId = result.recordset[0]['user\_id'];

if (policyName.toLowerCase() === 'life') {

const sumAssure = 500000;

const premiumAmount = 20000;

const policyActivationQuery = `UPDATE user\_details SET Sum\_Assured = ${sumAssure}, premium\_amount = ${premiumAmount} WHERE username = '${usertxt}';

UPDATE user\_policies SET policy\_status = 'Active' WHERE user\_id = ${userId}`;

await sql.query(policyActivationQuery);

console.log('Life policy activated');

return res.json({ message: 'Life policy activated' });

} else if (policyName.toLowerCase() === 'health') {

const sumAssure = 200000;

const premiumAmount = 10000;

console.log(premiumAmount);

const policyActivationQuery = `UPDATE user\_details SET Sum\_Assured = ${sumAssure}, premium\_amount = ${premiumAmount} WHERE username = '${usertxt}';

UPDATE user\_policies SET policy\_status = 'Active' WHERE user\_id = ${userId}`;

await sql.query(policyActivationQuery);

console.log('Health policy activated');

return res.json({ message: 'Health policy activated' });

} else {

return res.json({ message: 'Unknown policy type' });

}

} else {

return res.status(404).json({ message: 'User not found' });

}

} catch (error) {

console.error('Error:', error);

return res.status(500).json({ message: 'An error occurred' });

} finally {

// Close the SQL Server connection

sql.close();

}

});

// Start the server

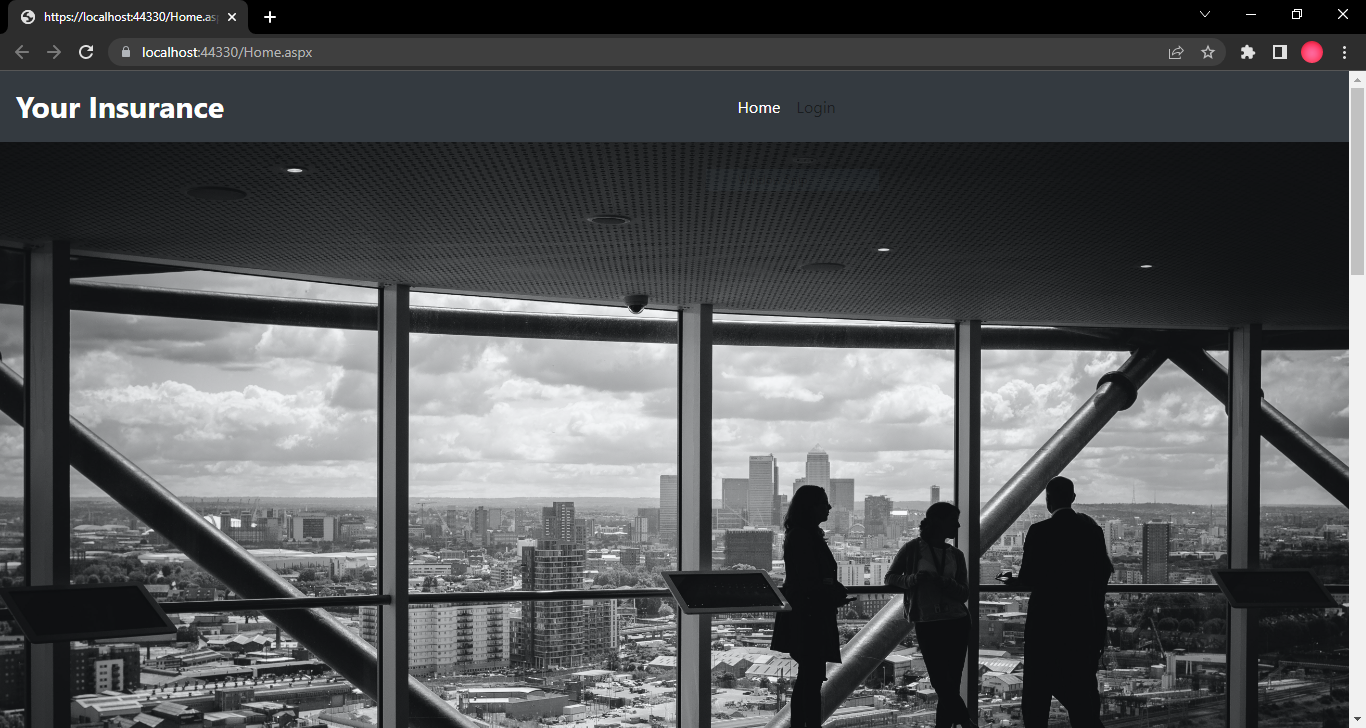
const port = 3000;

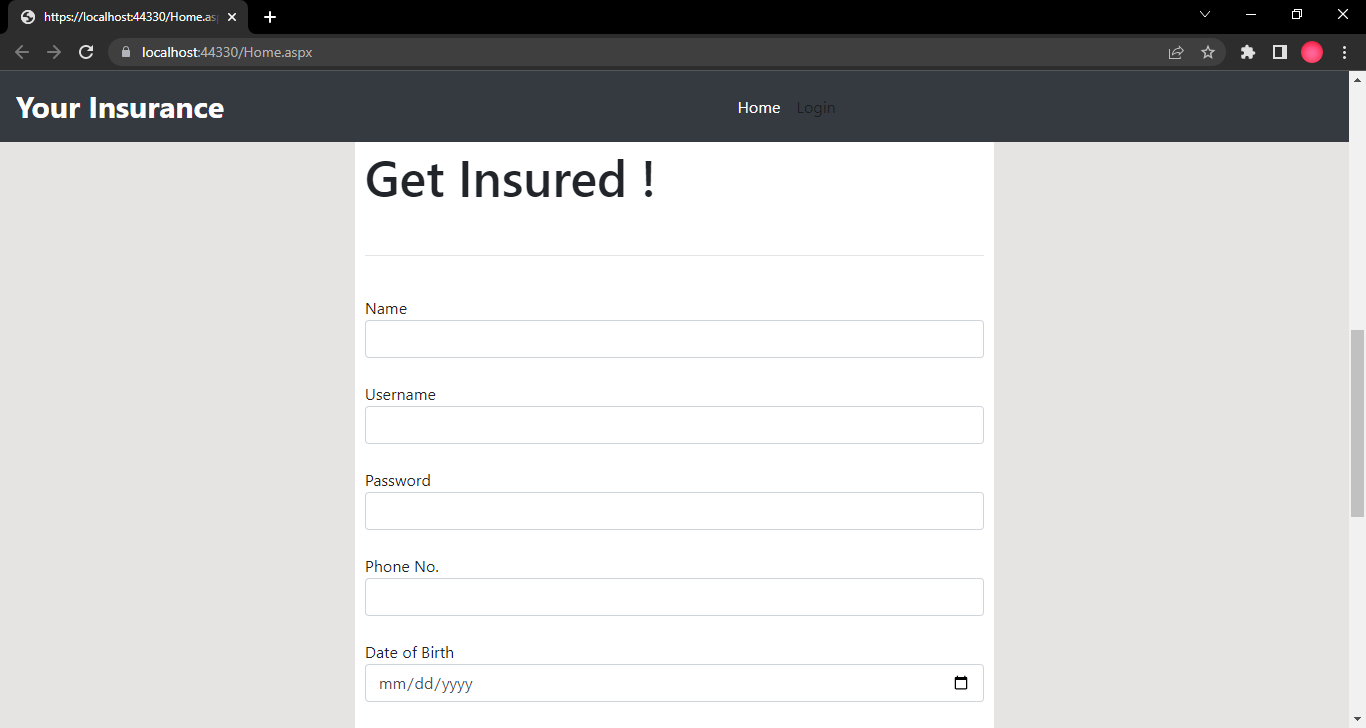
app.listen(port, () => {

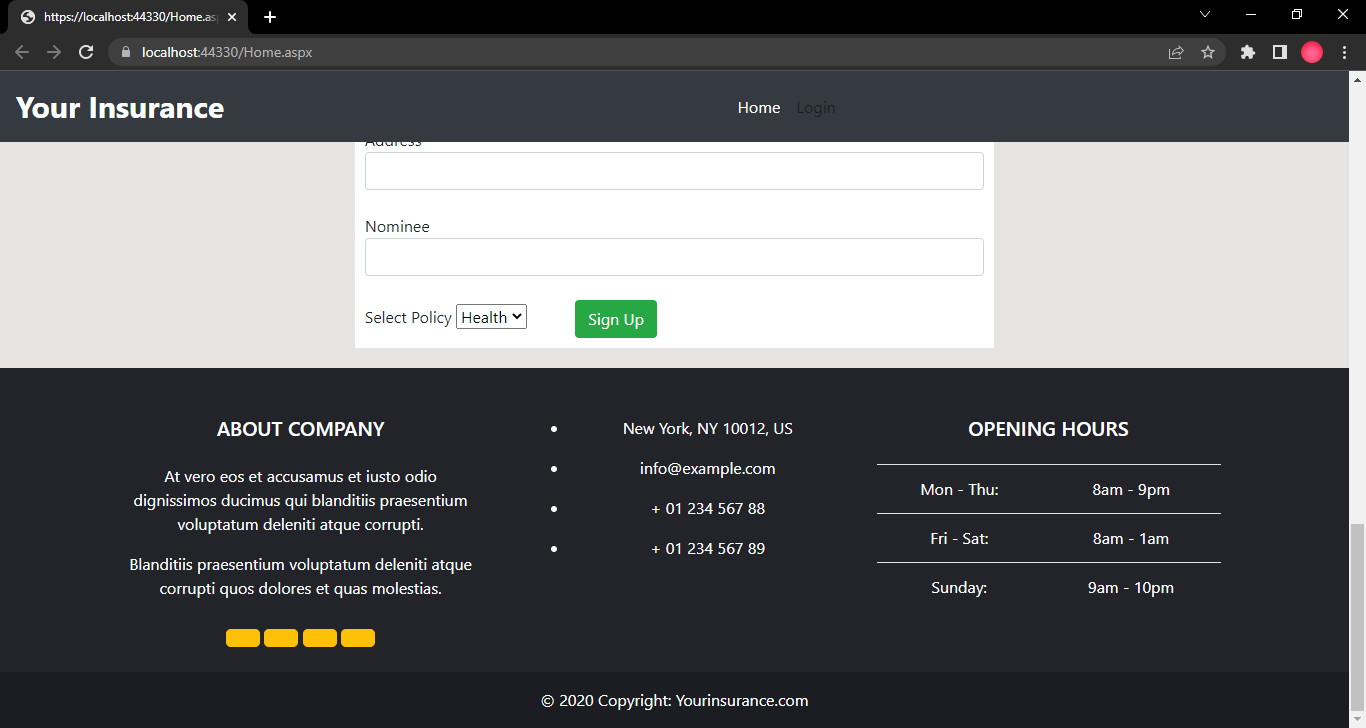
console.log(`Server is running on port ${port}`);

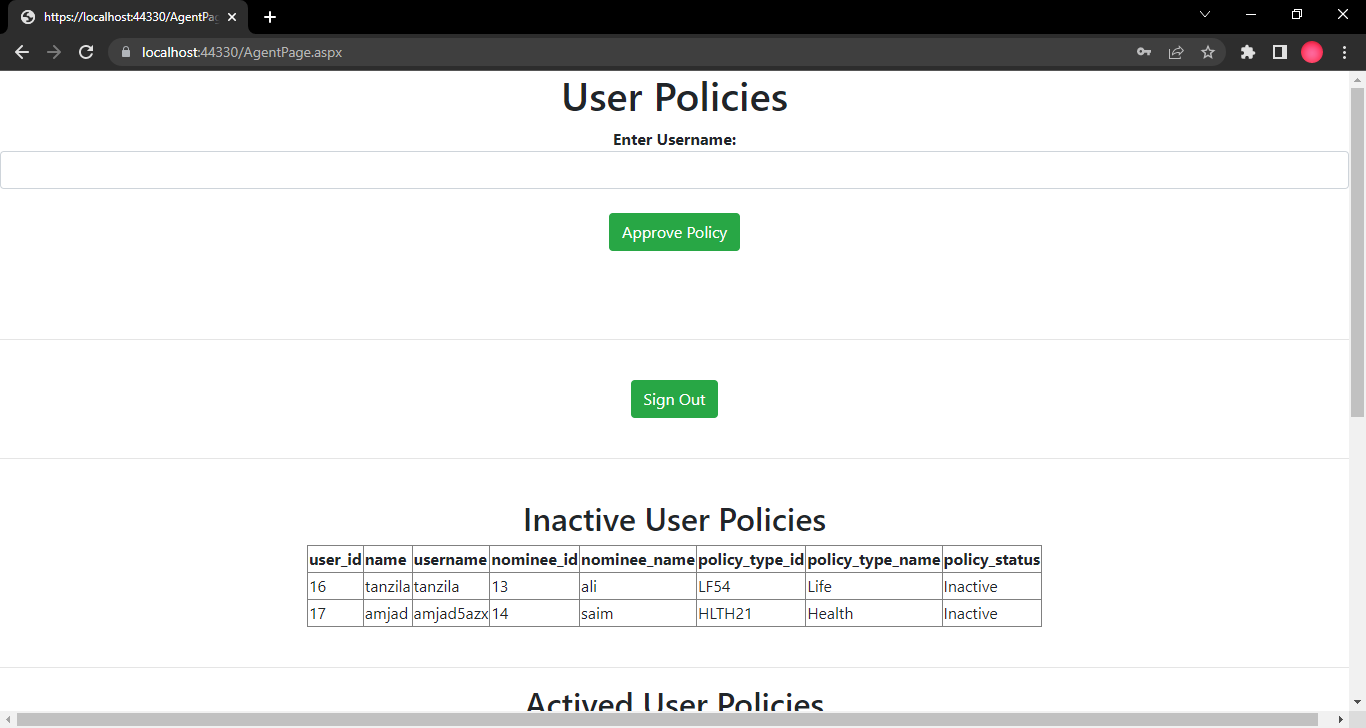
});

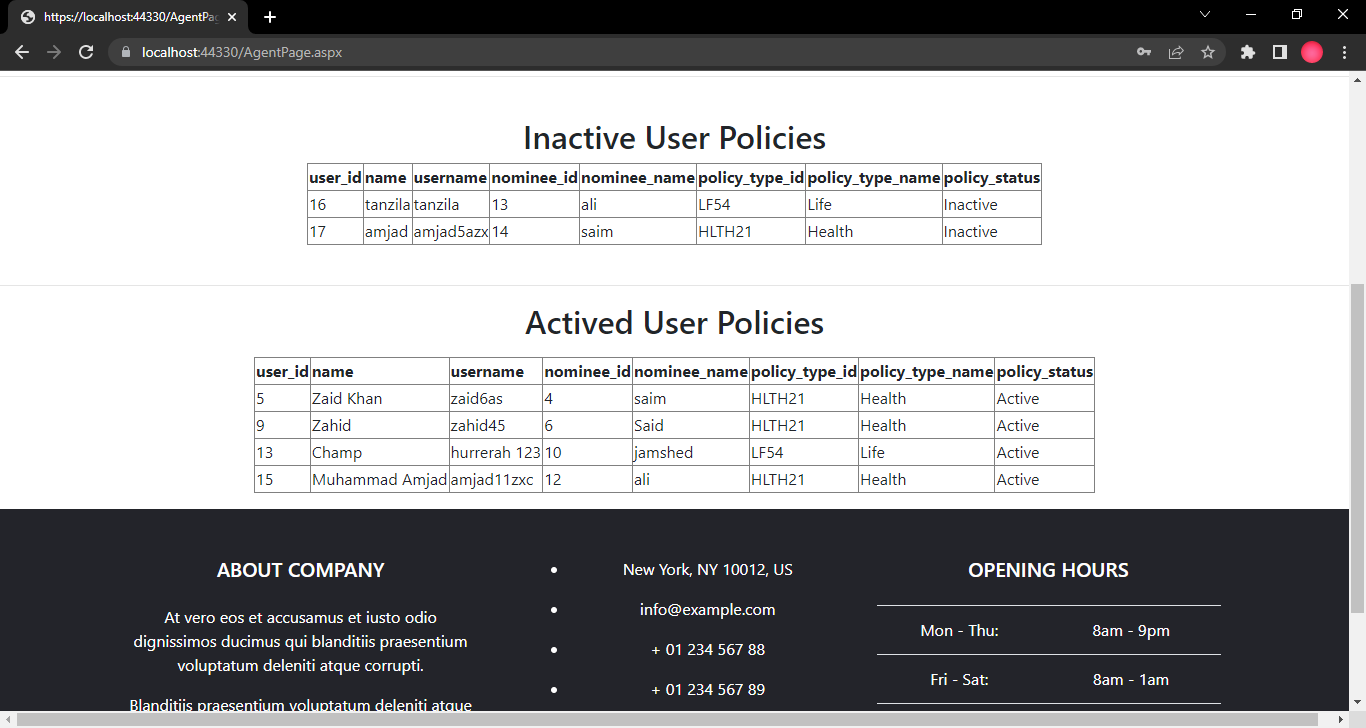
**INTERFACES:**

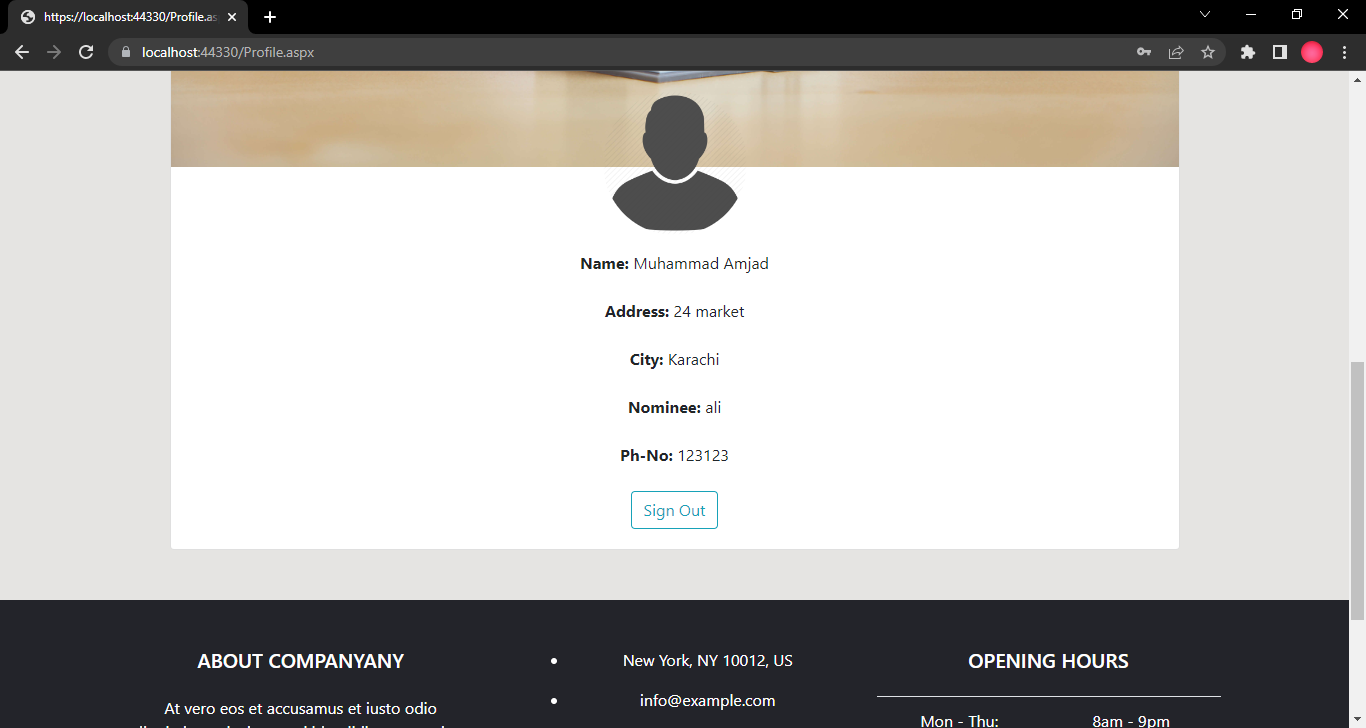


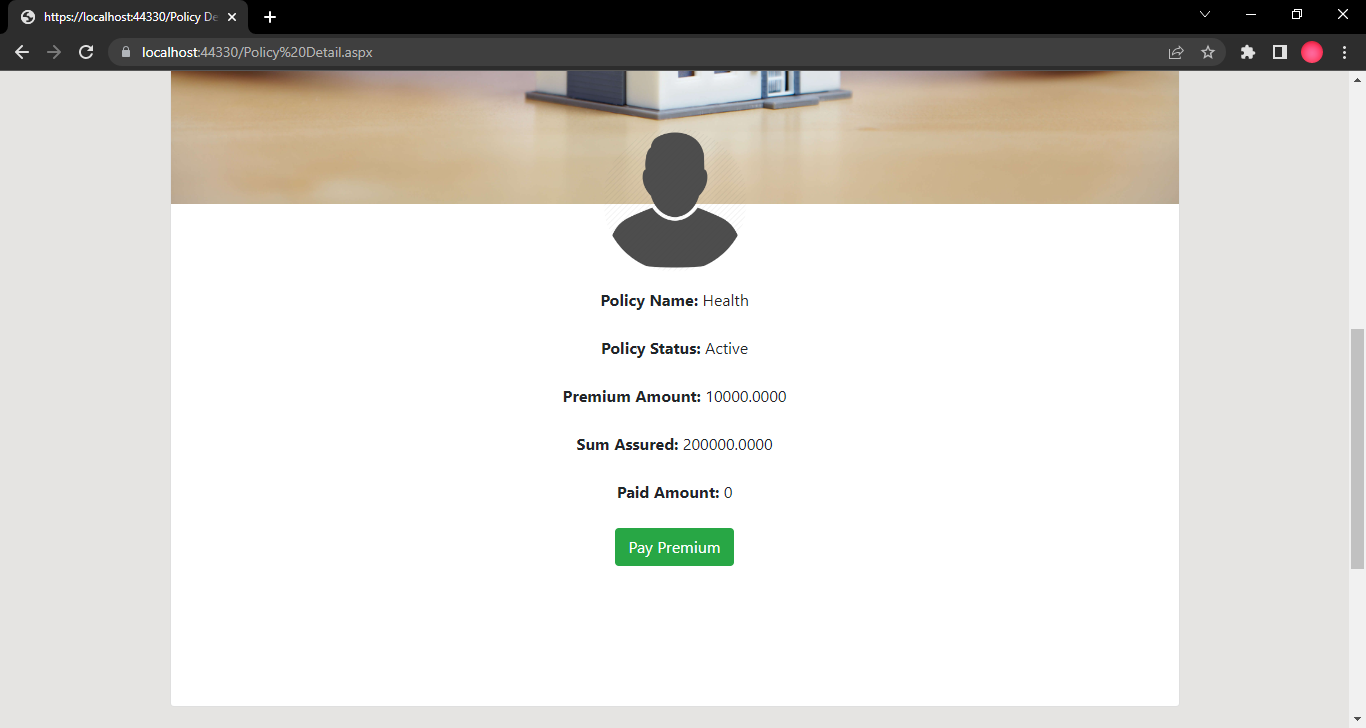


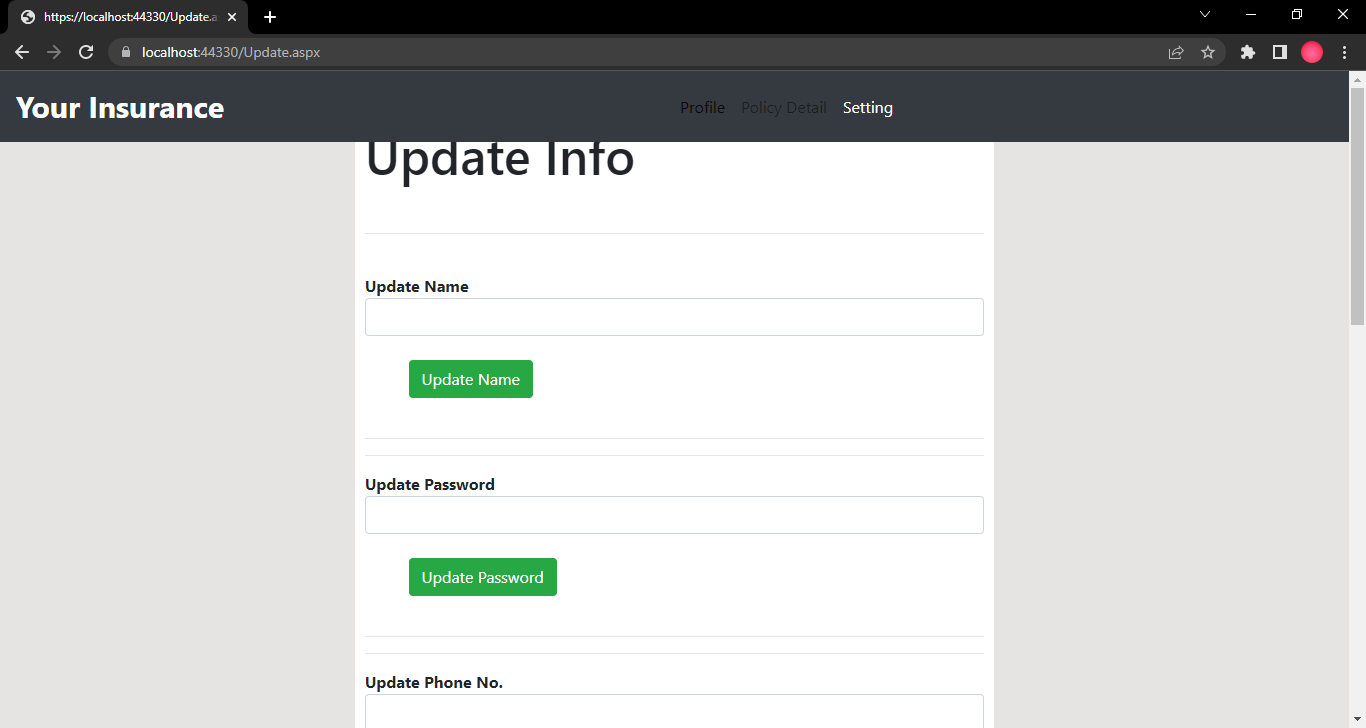












**CONCLUSION:**

In conclusion, our Insurance Management System leverages cloud technologies for efficient hosting, API management, and database storage. It provides functionalities for admins to view client details and manage policies, agents to monitor policies and activate them, and clients to sign up, pay premiums, and file claims. By streamlining processes and promoting transparency, the system aims to enhance customer satisfaction and optimize operations in the insurance industry.