Experiment-01

Student Name: Vikash Upadhyay **UID:** 22BCS16124
Branch: BE-CSE **Section/Group:** NTPP-602_A
Semester: 6th **Date of Performance:** 14-01-2025
Subject Name: AP-2 **Subject Code:** 22CSH-359

1. **Aim:** To develop an understanding and implementation of full-stack development using the MERN stack (MongoDB, Express.js, React, and Node.js).

2. **Objective:**
   a. To understand the components of the MERN stack and their integration.
   b. To design a frontend interface for login/signup pages.
   c. To create backend APIs for handling user authentication.
   d. To test backend APIs using tools like Postman.
   e. To build a full-stack application that integrates the frontend and backend for user authentication.

3. **Implementation/Code:**

Backend Implementation

1. **Setup the Backend**
   Clone the backend repository:

```
bash
CopyEdit
git clone https://github.com/Roshk7021
cd backend
npm install
```

2. **Database Connection Code:**

```
javascript
CopyEdit
const { MongoClient, ServerApiVersion } = require('mongodb');
const uri =
"mongodb+srv://rosh63441:<password>@rosh.yhbuk.mongodb.net/?retryWrites
=true&w=majority&appName=Rosh";
const client = new MongoClient(uri, {
    serverApi: {
        version: ServerApiVersion.v1,
        strict: true,
        deprecationErrors: true,
    },
});
```

```
const connectDB = async () => {
    try {
        await client.connect();
        await client.db("admin").command({ ping: 1 });
        console.log("Successfully connected to MongoDB!");
    } catch (err) {
        console.error(err.message);
        process.exit(1);
    }
};

module.exports = { connectDB, client };
```

1. **Authentication Routes:**
    - **Register User:**

```
javascript
CopyEdit
exports.register = async (req, res) => {
    const { name, email, password } = req.body;
    try {
        const db = client.db('auth');
        const users = db.collection('users');

        let user = await users.findOne({ email });
        if (user) return res.status(400).json({ msg: 'User
already exists' });

        const salt = await bcrypt.genSalt(10);
        const hashedPassword = await bcrypt.hash(password, salt);

        user = { name, email, password: hashedPassword };
        await users.insertOne(user);

        const payload = { user: { id: user._id } };
        jwt.sign(payload, 'secret', { expiresIn: 360000 }, (err,
token) => {
            if (err) throw err;
            res.json({ token });
        });
    } catch (err) {
        console.error(err.message);
        res.status(500).send('Server error');
    }
};
```

    - **Login User:**

```
javascript
CopyEdit
exports.login = async (req, res) => {
    const { email, password } = req.body;
    try {
        const db = client.db('auth');
        const users = db.collection('users');
        let user = await users.findOne({ email });
        if (!user) return res.status(400).json({ msg: 'Invalid
credentials' });

        const isMatch = await bcrypt.compare(password,
user.password);
        if (!isMatch) return res.status(400).json({ msg: 'Invalid
credentials' });

        const payload = { user: { id: user._id } };
        jwt.sign(payload, 'secret', { expiresIn: 360000 }, (err,
token) => {
            if (err) throw err;
            res.json({ token });
        });
    } catch (err) {
        console.error(err.message);
        res.status(500).send('Server error');
    }
};
```

2. **User Model (Mongoose Schema):**

```
javascript
CopyEdit
const mongoose = require('mongoose');
const UserSchema = new mongoose.Schema({
    name: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    password: { type: String, required: true },
});
module.exports = mongoose.model('User', UserSchema);
```

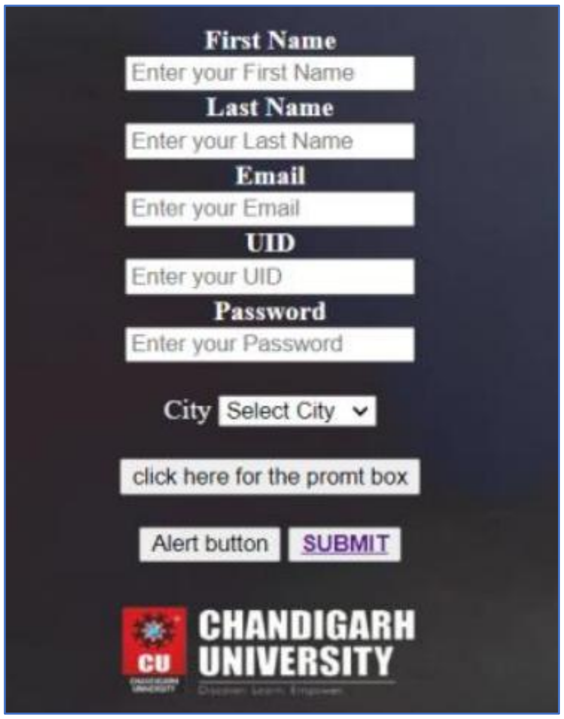
Frontend Implementation

1. **Setup the Frontend**
   Navigate to the frontend directory:

```
bash
CopyEdit
cd frontend
npm install
npm start
```

2. **Frontend Components:**
   - Design login/signup pages using React components.
   - Use Axios to send HTTP requests to the backend for authentication.

4. Output:



5. Learning Outcome:

a. MongoDB: Learned to use MongoDB as a NoSQL database for data storage and retrieval.

b. Node.js: Gained knowledge of using Node.js for backend development.

c. Express.js: Understood how to create RESTful APIs and manage routes.

d. React: Built dynamic user interfaces for frontend development.

e. API Testing: Used Postman to test and debug backend APIs.

f. Full-Stack Integration: Successfully integrated frontend and backend for a full-stack web application.