Experiment 1

Student Name: Harshit	**UID: 22BCS11117**
Branch: CSE	**Section/Group: NTPP_602(B)**
Semester: 6	**Date of Performance:**
Subject Name: AP LAB-II	**Subject Code: 22CSP-351**

1. Aim:

Full Stack Development (MERN). The primary aim of this experiment is to provide students or developers with an understanding of full-stack development involving MongoDB, Node.js, React, and Express.

1. Problem 1.1.1: Give understanding of MongoDB, Nodejs, React, Express.
2. Problem 1.1.2: Create a Frontend design of Login/Signup pages and create a backend of it.
3. Problem 1.1.3: Test the Backend API Using Postman

2. Objective:

- Understand the fundamentals of MongoDB, Node.js, React, and Express
- Create a functional frontend for Login/Signup pages
- Develop a backend using Express and MongoDB
- Test the backend API using Postman

3. Implementation/Code:

Backend:

- mkdir backend cd backend
- npm init -y npm install
- express mongoose cors bcryptjs jsonwebtoken **Server.js** const express = require('express');

```
const mongoose = require('mongoose'); const
cors = require('cors'); const dotenv =
require('dotenv'); const authRoutes =
require('./routes/authRoutes');

dotenv.config();

const app = express(); app.use(express.json());
// to parse JSON bodies app.use(cors());  // to
handle CORS

// Connect to MongoDB const
mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  email: { type: String, required: true, unique: true },
password: { type: String, required: true }
});

const User = mongoose.model('User', userSchema);

module.exports = User;



app.use('/api', authRoutes);
```

```
    // Start Server const port = 5000; app.listen(port, () => {
    console.log(`Server running on http://localhost:${port}`);
    });
```

Users.js const mongoose =

```
require('mongoose'); const UserSchema = new
mongoose.Schema({
  email: { type: String, required: true, unique: true },
password: { type: String, required: true },
});

module.exports = mongoose.model('User', UserSchema);
```

authRoutes.js // Signup Route

```
router.post('/signup', async (req, res) => {
const { email, password } = req.body;
try {
    const existingUser = await User.findOne({ email });      if
(existingUser) {      return res.status(400).json({ message: 'User
already exists' });
    }

    const hashedPassword = await bcrypt.hash(password, 10);
const newUser = new User({ email, password: hashedPassword });
await newUser.save();

    res.status(201).json({ message: 'User created successfully' });
  } catch (error) {      console.error('Signup error:', error); // Log error to console
res.status(500).json({ message: 'Server error', error: error.message }); // Include the
error message in the response
  }  });  router.post('/login', async (req,
res) => {    const { email, password } =
req.body;

     try {      const user = await User.findOne({
email });
```

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

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

    const token = jwt.sign({ userId: user._id }, process.env.JWT_SECRET, {
expiresIn: '1h' });      res.json({ message: 'Login successful', token });
  } catch (error) {      console.error('Login error:', error); //
Log error to console      res.status(500).json({ message:
'Server error', error: error.message }); // Include the error
message in the response
  }
});
```

my-auth-app:

- npm create vite@latest my-auth-app --template react

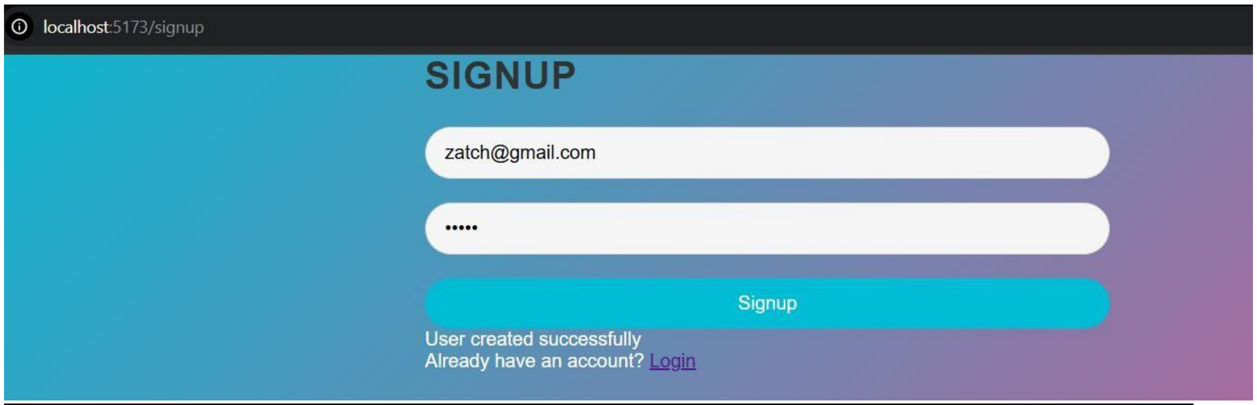
- cd my-auth-app
- npm install

App.jsx:

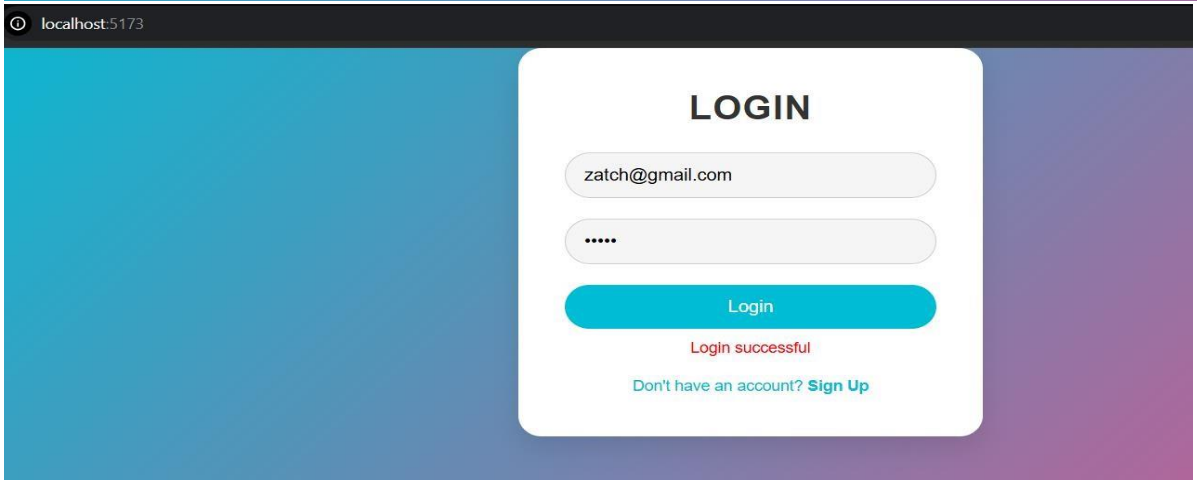
```
import React from 'react'; import { BrowserRouter as Router, Routes, Route

} from 'react-router-dom'; import Login from './Login'; import Signup from

'./Signup'; // In App.js or index.js import './index.css'; import './app.css';


const App = () => {

return (

    <Router>

      <div className="App">

        {/* <h1>Authentication App</h1> */}

        <Routes>

          <Route path="/" element={<Login />} />

          <Route path="/signup" element={<Signup />} />

        </Routes>

      </div>

    </Router>

  );
};



export default App;
```

4.

Output:



User created successfully
Already have an account? Login



Login successful

Don't have an account? Sign Up

5. Learning Outcome:

- Design user-friendly forms for user login and registration using React..
- Learn how each technology works individually and how they integrate to form a full-stack application.
- Set up a server with Express to handle HTTP requests for user registration and login

- Verify that the backend API functions as expected by testing the registration and login endpoints with Postman