



WORKSHEET 1

Student Name: Avreet Singh

UID: 22BCS16488

Branch: CSE

Section/Group: NTPP-602-A

Semester: 6

Date of Performance: 15/01/2025

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 how to use MongoDB as a NoSQL database for managing user data.
- Learn to set up and use Node.js as a backend server to handle API requests.
- Explore Express.js for creating routes and managing HTTP requests in Node.js.
- Build a simple frontend interface with React for user interactions like login and signup.
- Test backend APIs using tools like Postman to verify server responses.
- Integrate the React frontend with the backend API to develop a full-stack authentication system.
- Implement password hashing with bcrypt.js to enhance security.



3. Source code/Implementation:

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());
app.use(cors());

mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => console.log('MongoDB connected'))
  .catch((err) => console.error('MongoDB connection
error:', err));

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

const port = process.env.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

```
const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const User = require('../models/User');

const router = express.Router();

// Signup
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);
    res.status(500).json({ message: 'Server error' });
  }
});

// Login
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);
    res.status(500).json({ message: 'Server error' });
  }
});

module.exports = router;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

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';  
  
const App = () => {  
  return (  
    <Router>  
      <div className="App">  
        <Routes>  
          <Route path="/" element={<Login />} />  
          <Route path="/signup" element={<Signup />} />  
        </Routes>  
      </div>  
    </Router>  
  );  
};  
  
export default App;
```

4. Screenshots of outputs:

Username

Password

Login



5. Learning Outcomes:

- Understand and implement MongoDB as a NoSQL database for managing user data.
- Build and configure a Node.js backend server using Express.js to handle API requests and routes.
- Design and develop user-friendly interfaces for login and signup using React.
- Securely authenticate users by implementing password hashing and JSON Web Tokens (JWT).
- Test and validate backend APIs using Postman to ensure proper functionality and error handling.
- Integrate frontend and backend components to create a complete and secure full-stack authentication system.