Experiment - 1

Student Name: **Ayush Pathania** UID: **22BCS16023**

Branch: **BE - CSE** Section/Group: **NTPP 602 - A**

Semester: **6** Sub Code: **22CSP-351**

Subject Name: **Advanced Programming Lab - 2** Date: **14/01/2025**

Problem - 1

Aim - 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. Give understanding of MongoDB, Nodejs, React, Express.
2. Create a Frontend design of Login/Signup pages and create a backend of it.
3. Test the Backend API Using Postman.

Objective -

1. Gain knowledge of MongoDB, Node.js, React, and Express to build a functional web application.
2. Design a Login/Signup frontend, create a backend with Express and MongoDB, and test APIs using Postman.

Implementation/Code -

1. *Index.js*

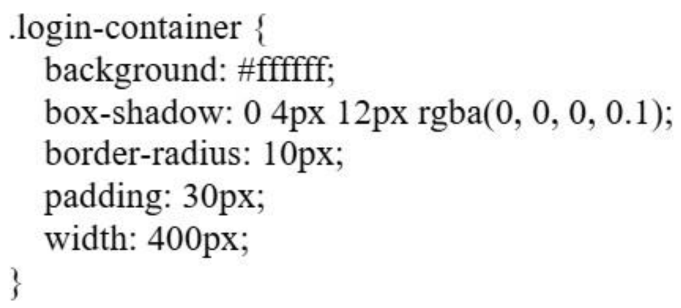
```
const express = require("express") const app=express()
const path=require("path") const hbs=require("hbs")
const collection=require("./mongodb")
const templatePath=path.join(   dirname,'../templates') app.use(express.json())
app.set("view engine","hbs") app.use(express.static(path.join(   dirname, '../public')));
app.set("views",templatePath) app.use(express.urlencoded({extended:false}))
app.get("/",(req,res)=>{res.render("home");})
app.get("/signup",(req,res)=>{ res.render("signup");})
app.get("/login",(req,res)=>{ res.render("login");})
app.post("/signup",async (req,res)=>{ const data={
name:req.body.name, password:req.body.password,}
await collection.insertMany([data])
res.render("home")})
app.listen(3000,()=>{ console.log("port connected");})
```

2. *Mongodb.js*

```
const mongoose=require("mongoose");
mongoose.connect('mongodb://localhost:27017/Yash',
{useNewUrlParser: true, useUnifiedTopology: true, }).then(() =>
console.log('MongoDB connected')).catch((err) => console.error('MongoDB
connection error:', err));
const LogInSchema=new mongoose.Schema({ name:{ type:String, required:true},
password:{ type:String, required:true}})
const collection=new mongoose.model("Collection1",LogInSchema)
module.exports=collection
```

3. *Index.html*

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Page</title>
  <style>
    body {
      margin: 0;
      font-family: 'Arial', sans-serif;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background: #f4f4f4;
      color: #333;
    }
```



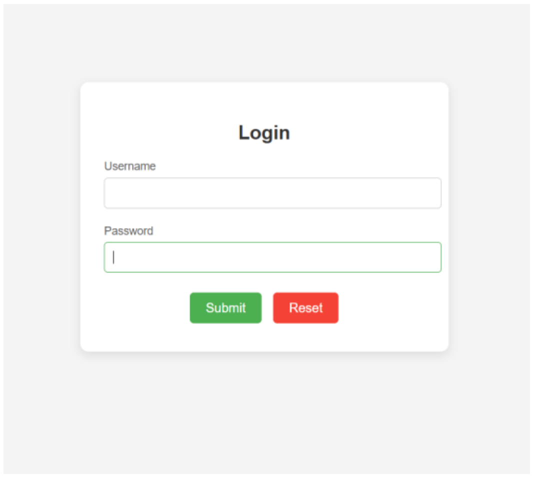
```
.login-container {
  background: #ffffff;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
  border-radius: 10px;
  padding: 30px;
  width: 400px;
}
```

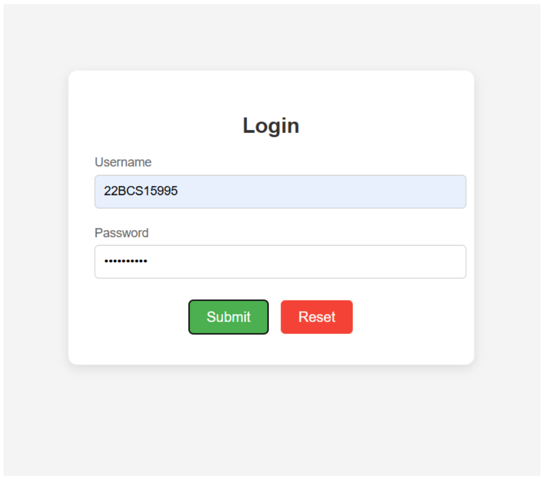
```
        .login-container h2 {
            text-align: center;
            margin-bottom: 20px;
            font-size: 24px;
            color: #333;
        }

        .input-group label {
            font-size: 14px;
            color: #666;
            margin-bottom: 5px;
            display: block;
        }


        .input-group input:focus {
            border-color: #4caf50;
        }

        .btn-container {
            text-align: center;
        }

        .btn:hover {
            background: #45a049;
        }

        .reset-btn {
            background: #f44336;
        }

        .reset-btn:hover {
            background: #e53935;
        }

    </style>
</head>
<body>
    <div class="login-container">
```

```
.input-group {
    margin-bottom: 20px;
    position: relative;
}


.input-group input {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 14px;
    outline: none;
    transition: border-color 0.3s;
}


.btn {
    padding: 10px 20px;
    background: #4caf50;
    border: none;
    border-radius: 5px;
    color: #fff;
    font-size: 16px;
    cursor: pointer;
    transition: background 0.3s;
    margin: 5px;
}


.message {
    margin-top: 15px;
    font-size: 14px;
    text-align: center;
    color: #4caf50;
    display: none;
}

.message.show {
    display: block;
}
```

```
    <h2>Login</h2>
    <form id="login-form">
        <div class="input-group">
            <label for="username">Username</label>
            <input type="text" id="username" name="username" required>
        </div>
        <div class="input-group">
            <label for="password">Password</label>
            <input type="password" id="password" name="password" required>
        </div>
        <div class="btn-container">
            <button type="submit" class="btn">Submit</button>
            <button type="reset" class="btn reset-btn">Reset</button>
        </div>
    </form>
    <div class="message" id="success-message">Login Successful!</div></div>
<script>const form = document.getElementById('login-form');
    const message = document.getElementById('success-message');
    form.addEventListener('submit', (e) => {
        e.preventDefault();
        message.classList.add('show');
        setTimeout(() => {
            message.classList.remove('show');
        }, 3000);
    }); </script></body></html>
```

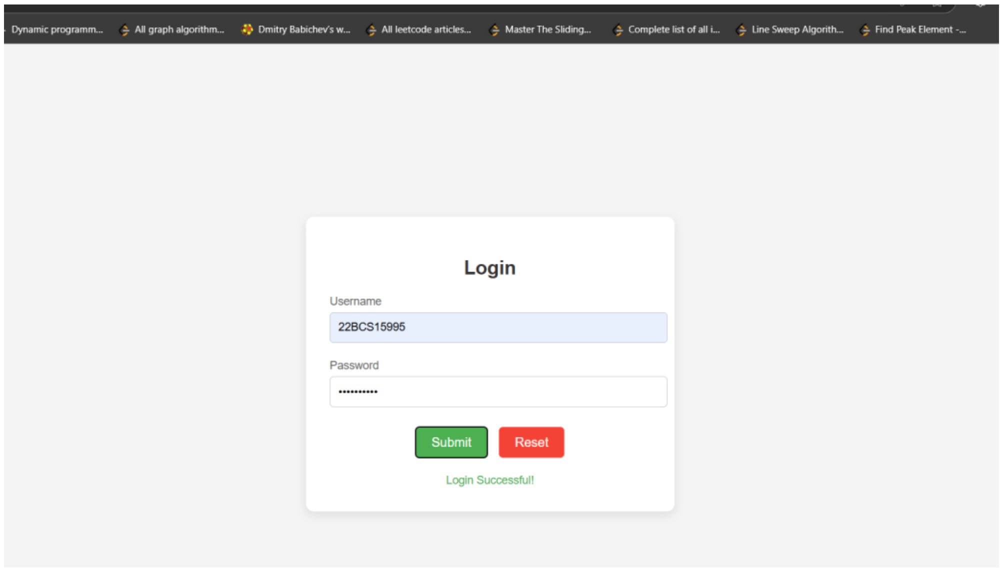
Output -





22BCS15995

••••••••••



Login

Username: 22BCS15995

Password: ••••••••••

Login Successful!

Learning Outcomes -

1. Learn about MongoDB: Understand how to use MongoDB as a NoSQL database for storing and retrieving user data.
2. Learn about Node.js: Understand how to set up and use Node.js as a backend server and handle API requests.
3. Learn about Express.js: Understand how to use Express.js to create routes and handle HTTP requests in the Node.js server.
4. Learn about React: Learn how to create a simple frontend interface with React to handle user interactions (login/signup).
5. Backend API Testing: Use tools like Postman to test backend APIs and ensure the server is responding correctly.
6. Integration: Integrate the frontend (React) with the backend API to create a full-stack authentication system.