MERN 5 - Project Intro

title: Project Setup and Adding Frontend Code for Registration

Before we move into authentication and authorization, it's essential to set up the project infrastructure. We'll start by configuring React, Express, Node.js, and MongoDB. These components must be installed and configured correctly to proceed.

To begin, let's create a folder named "bookMyShow-project." Inside this folder, we will create another one called "client," where we'll develop the frontend of our application. To kickstart the React app, use the following command:

npx create-react-app client

Additional commands that might be needed

```
npm install react@18 react-dom@18
rm -rf node_modules package-lock.json
npm cache clean --force
npm install
```

Additionally, create a folder named "server" in the main project directory. The "client" folder will house all the frontend code, while the "server" folder will be dedicated to setting up our Express backend.

With the React app initialized, let's navigate to the "App.js" file to continue.

With our React app successfully set up, it's time to create two fundamental pages: the login and registration pages. These pages are essential components of any application.

Our initial set of pages includes:

1. Home Page
2. Register Page
3. Login Page

To expedite the development process, we will leverage Ant Design, a library that provides pre-built components, eliminating the need for extensive custom CSS. Within the component section, you'll find various components, each with an example code.

Choose Ant Design if:

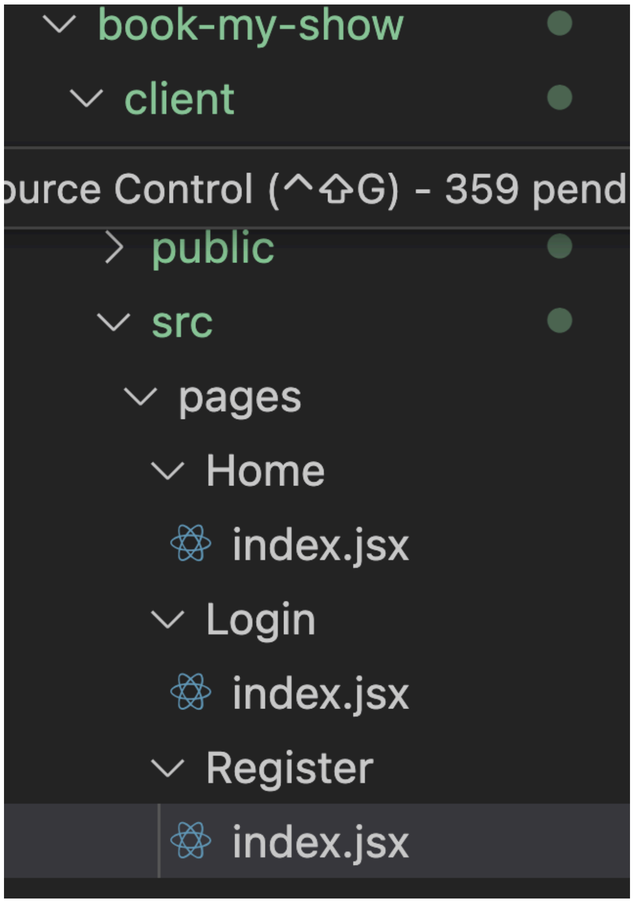
- You need a comprehensive set of pre-designed components to speed up development.
- You are building an enterprise application and want a consistent, professional look with minimal effort.
- You prefer following a well-defined design system.
- You need robust documentation and support for complex use cases

Since our focus is on building the login and registration pages, we'll primarily work with forms. So, let's navigate to the component section, search for "form," and explore multiple form examples. You can choose a form that suits your needs and copy the code from there.

In our client directory, let's begin by installing the required dependencies:

npm install react-router-dom antd axios

Now, let's structure our project by creating a "pages" folder. Inside this folder, create three subfolders: "login," "register," and "Home." Within each of these subfolders, create an "index.js" file.



```
book-my-show
  client
ource Control (^⇧G) - 359 pend
    public
    src
      pages
        Home
          index.jsx
        Login
          index.jsx
        Register
          index.jsx
```

Write some psuedocode in Home , Login and Register

```
const Home = () => {
   return (
     <div>
       This is my home page
     </div>
   )
 }
 export default Home;
```

```
const Login = () => {
   return (
     <div>
       This is my Login page
     </div>
   )
 }
 export default Login;
```

```
const Register = () => {
   return (
     <div>
       This is my Register page
     </div>
   )
 }
 export default Register;
```

We will navigate to the 'App.js' file import all of these components set up React Router DOM and create our first route

Update App.jsx

```
import Home from "./pages/Home";
import Login from "./pages/Login";
import Register from "./pages/Register";
import { BrowserRouter, Routes, Route } from "react-router-dom";

function App() {
 return (
   <div>
     <BrowserRouter>
       <Routes>
         <Route path="/" element={<Home />} />
         <Route path="/login" element={<Login />} />
         <Route path="/register" element={<Register />} />
       </Routes>
     </BrowserRouter>
   </div>
 );
}

export default App;
```

title: Design the Login and Register Page

Copy this CSS code in the app.css file , this the css for alignment and basic styling , this is not be explained in the class, learners can make sense of css this is just to be copied and used.

```css
.App-logo {
 height: 40vmin;
 pointer-events: none;
}
.App-header {
 background-color: #f3f8ff;
 min-height: 100vh;
 display: flex;
 flex-direction: column;
 align-items: center;
 justify-content: center;
}
.flex-1{
 flex: 1;
}
.ms-3{
 margin-left: 1rem;
}
.text-center{
 text-align: center;
}
.cursor-pointer{
 cursor: pointer;
}
.inner-container{
 max-width: 1000px;
 margin-inline: auto;
}
.max-width-300{
 max-width: 300px;
}
.movie-data{
 color: #999;
 font-size: 15px;
 margin-top: 4px;
 margin-bottom: 8px;
}
.movie-data span{
 color: #212121;
 font-weight: 600;
}
.flex-shrink-0{
 flex-shrink: 0;
}
.App-link {
 color: #61dafb;
}
.d-block{
 display: block;
}
input, select{
 min-height: 45px;
}
button{
 min-height: 45px;
 font-size: 1rem;
}
.w-100{
 width: 100%;
}
.text-white{
 color: #fff;
}
.m-0{
 margin: 0
}
.mt-0{
 margin-top: 0
}
.mt-3{
 margin-top: 1rem;
}
.me-3{
 margin-right: 1rem;
}
```

```css
.gap-10{
 gap: 10px;
}
.justify-content-between{
 justify-content: space-between;
}
.justify-content-end{
 justify-content: end;
}
.fs-12{
 font-size: 12px;
}
.ant-select-selector{
 height: 45px !important;
}
.ant-select-selection-item{
 padding-top: 5px !important;
}
table th{
 white-space: nowrap;
}
.ant-modal .ant-modal-title{
 font-size: 2rem;
}
.py-3{
 padding-block: 1rem;
}
.pb-3{
 padding-bottom: 1rem;
}
.pt-3{
 padding-top: 1rem;
}
.fs-18{
 font-size: 18px;
}
.d-none{
 display: none;
}
.mb-3{
 margin-bottom: 1rem;
}
```

```css
.mb-5{
 margin-bottom: 50px;
}
.mb-10px{
 margin-bottom: 10px;
}
.mb-25px{
 margin-bottom: 25px;
}
.ant-menu{
 flex: 1;
 justify-content: flex-end;
}
.mt-8{
 margin-top: 8px;
}
hr{
 background-color: #eee !important;
 border: 1px solid #ddd;
}
.show-ul{
 list-style: none;
 display: flex;
 flex-wrap: wrap;
 gap: 0.5rem;
 padding-left: 0;
}
.blue-clr{
 color: #1890ff
}
.show-ul li{
 padding: 0.75rem 1rem;
 border: 1px solid #bbb;
 border-radius: 5px;
 cursor: pointer;
 color: orange;
 transition: all 0.1s ease-in;
}
.show-ul li:hover{
 color: #fff;
 background-color: orange;
 border-color: transparent;
}
.movie-title-details *{
 margin-bottom: 6px;
 margin-top: 0;
}
.show-name *{
 margin-bottom: 4px;
 margin-top: 0;
}
.show-name span{
 font-weight: 500;
 color: #555;
}
.max-width-600{
 max-width: 600px;
}
.mx-auto{
 margin-inline: auto;
}
.seat-ul{
 list-style-type: none;
 padding: 0;
 margin: 0;
 display: flex;
 gap: 0.5rem;
 flex-wrap: wrap;
 max-width: 600px;
}
.seat-ul .seat-btn{
 background-color: #f6f6f6;
 border: 1px solid #a0a0a0;
 width: 40px;
 height: 35px;
 text-align: center;
 line-height: 1;
 cursor: pointer;
 font-size: 13px;
 min-height: unset;
}
.seat-ul .seat-btn:hover{
 background-color: #f1f1f1;
}
```

```css
.seat-ul .seat-btn.selected{
 border-color: #02a802;
 background-color: #02a802;
 color: #fff;
}
.seat-ul .seat-btn.booked{
 background-color: #ddd;
 color:#999;
 border-color: #ddd;
 cursor:not-allowed;
}
.screen-div{
 max-width: 75%;
 margin-inline: auto;
 height: 10px;
 background-color: #eee;
}
.bottom-card{
 padding: 1rem;
 border: 1px solid #ccc;
 border-radius: 5px;
}
.bottom-card div span{
 font-weight: 600;
 font-size: 1.25rem;
}
.show-details h3{
 border-bottom: 1px solid #e9e9e9;
 padding-bottom: 4px;
}
.show-details *{
 margin-bottom: 2px;
 margin-top: 0;
}
.px-3{
 padding-inline: 1rem;
}

/* Loader */
.loader-container{
 position: fixed;
 top: 0;
 left: 0;
 width: 100%;
 height: 100%;
 background-color: rgba(0,0,0,0.35);
 z-index: 9;
 display: flex;
 justify-content: center;
 align-items: center;
}
.loader {
 border: 16px solid #f3f3f3;
 border-radius: 50%;
 border-top: 16px solid #3498db;
 width: 120px;
 height: 120px;
 -webkit-animation: spin 2s linear infinite; /* Safari */
 animation: spin 2s linear infinite;
}
.ant-table-content{
 overflow-y: auto;
 width: 100%;
}
@-webkit-keyframes spin {
 0% { -webkit-transform: rotate(0deg); }
 100% { -webkit-transform: rotate(360deg); }
}

@keyframes spin {
 0% { transform: rotate(0deg); }
 100% { transform: rotate(360deg); }
}
/* Loader Ends here */

@media screen and (min-width: 768px){
 .site-layout{
   padding: 0 50px;
 }
 .show-details{
   margin-left: 1rem;
 }
 .mw-500{
   width: 500px;
 }
}
@media screen and (max-width: 767px){
 .mt-3-mob{
   margin-top: 1rem;
 }
 .mt-8px-mob{
   margin-top: 8px;
 }
 .ant-layout-header{
   padding-inline: 10px;
 }
 .flex-column-mob{
   flex-direction: column;
 }
 .single-movie-div{
   flex-direction: column;
 }
 .single-movie-img{
   margin-right: 0;
   margin-bottom: 1rem;
 }
}
```

title: Designing the Login page

```
import React from "react";
import { Button, Form, Input } from "antd";
import { Link } from "react-router-dom";

function Login() {
 return (
   <>
     <main className="App-header">
       <h1>Login to BookMyShow</h1>
       <section className="mw-500 text-center px-3">
         <Form layout="vertical">
```

```
          <Form.Item
            label="Email"
            htmlFor="email"
            name="email"
            className="d-block"
            rules={[{ required: true, message: "Email is required" }]}
          >
            <Input
              id="email"
              type="text"
              placeholder="Enter your Email"
            ></Input>
          </Form.Item>

          <Form.Item
            label="Password"
            htmlFor="password"
            name="password"
            className="d-block"
            rules={[{ required: true, message: "Password is required" }]}
          >
            <Input
              id="password"
              type="password"
              placeholder="Enter your Password"
            ></Input>
          </Form.Item>

          <Form.Item className="d-block">
            <Button
              type="primary"
              block
              htmlType="submit"
              style={{ fontSize: "1rem", fontWeight: "600" }}
            >
              Login
            </Button>
          </Form.Item>
        </Form>
        <div>
          <p>
            New User? <Link to="/register">Register Here</Link>
          </p>
        </div>
      </section>
    </main>
  </>
 );
}

export default Login;
```

title: Design The Resgiter Page

```
import React from "react";
import { Button, Form, Input, message } from "antd";
import { Link } from "react-router-dom";

function Register() {
 return (
  <>
    <main className="App-header">
      <h1>Register to BookMyShow</h1>
      <section className="main-area mw-500 text-center px-3">
        <Form layout="vertical">
          <Form.Item
            label="Name"
            htmlFor="name"
            name="name"
            className="d-block"
            rules={[{ required: true, message: "Name is required" }]}
          >
            <Input
              id="name"
              type="text"
              placeholder="Enter your name"
            ></Input>
          </Form.Item>

          <Form.Item
            label="Email"
            htmlFor="email"
            name="email"
            className="d-block"
            rules={[{ required: true, message: "Email is required" }]}
          >
            <Input
              id="email"
              type="text"
              placeholder="Enter your Email"
            ></Input>
          </Form.Item>

          <Form.Item
            label="Password"
            htmlFor="password"
            name="password"
            className="d-block"
            rules={[{ required: true, message: "Password is required" }]}
          >
            <Input
              id="password"
              type="password"
              placeholder="Enter your Password"
            ></Input>
          </Form.Item>

          <Form.Item className="d-block">
            <Button
              type="primary"
              block
              htmlType="submit"
              style={{ fontSize: "1rem", fontWeight: "600" }}
            >
              Register
            </Button>
          </Form.Item>
        </Form>
        <div>
          <p>
            Already a user? <Link to="/login">Login now</Link>
          </p>
```

```
         </div>
       </section>
     </main>
   </>
 );
}

export default Register;
```

Adding email input type validation as well

Login

```
 <Form.Item
             label="Email"
             htmlFor="email"
             name="email"
             className="d-block"
             rules={[
               { required: true, message: "Email is required" },
               { type: "email", message: "Please enter a valid email" },
             ]}
           >
```

Register

```
           <Form.Item
             label="Email"
             htmlFor="email"
             name="email"
             className="d-block"
             rules={[
               { required: true, message: "Email is required" },
               { type: "email", message: "Please enter a valid email" },
             ]}
           >
```

title: Establish Server-Side Architecture

Next, we'll move into the server-side operations. Within our server directory, we'll create a file named 'server.js' and proceed to install the Express framework.

Initialise the node project with **npm init -y**

To enhance security, we'll make use of 'bcryptjs', a package specialized in password hashing. The necessary packages can be installed with the following command:

npm install express mongoose bcryptjs jsonwebtoken

In 'server.js', we'll initiate the Express server. This step forms the foundation for our backend operations.

ENV file

As part of our server setup, we'll create an environment configuration ('.env') file where you can store essential information such as the MongoDB URL and JSON Web Token (JWT) secrets. This allows us to keep sensitive data secure.

CREATE .env file

```
DB_URL='mongodb+srv://ayushrajsd:28Wca5DmXC6Q9BDW@cluster0.vg5saeo.mongodb.net/'
```

Furthermore, we'll establish a 'dbconfig.js' file in the 'config' folder. This file will contain the configuration settings for connecting to our MongoDB databas

In order to load the environment variables, we need to install dotenv package

And then in the server.js file, do this

```
const express = require('express');
const app = express();
require('dotenv').config(); // Load environment variables
const connectDB = require('./config/db'); // Import database configuration
console.log("server",process.env.DB_URL);
connectDB(); // Connect to database

app.listen(8082, () => {
   console.log('Server is Running');
});
```

Create config/db.js

```
const mongoose = require("mongoose");


// driver
```

```
const dbURL = process.env.DB_URL;
console.log(dbURL);
// once

const connectDB = async () => {
 try {
   await mongoose.connect(dbURL);
   console.log("connected to db");
 } catch (err) {
   console.log(err);
 }
};

module.exports = connectDB;
```

To bring everything together, the revised 'server.js' code snippet includes the necessary 'require' statements, environment variable loading, and the inclusion of the 'dbConfig' module. This ensures that our server is properly configured and ready to run on port 8082."

title: Exploring Authentication

Authentication is a fundamental aspect of our application. We'll define the schema and create a model based on that schema. For user registration, our schema will encompass essential fields like name, password, and email.

To structure our project effectively, we'll establish a 'model' folder. Our initial model, 'userModel.js,' will commence with defining a schema. With this schema in place, we'll proceed to create a model, enabling the creation of various user documents

models/userModel.js

```
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,
 },
 isAdmin: {
   type: Boolean,
   required: true,
   default: false,
 },
});

const UserModel = mongoose.model("users", userSchema);
module.exports = UserModel;
```

User Routes

Routing is another crucial aspect. We'll create user routes within our 'routes' folder.

```
const express = require("express");
const User = require("../models/userModel");

const userRouter = express.Router();
// Register a user
userRouter.post("/register", async (req, res) => {
 try {
   const userExists = await User.findOne({ email: req.body.email });

   if (userExists) {
     return res.send({
       success: false,
       message: "User Already Exists",
     });
   }

   const newUser = new User(req.body);
   await newUser.save();

   res.send({
     success: true,
     message: "Registration Successful, Please login",
   });
 } catch (error) {
   console.log(error);
 }
});

module.exports = userRouter;
```

In the 'server.js' file, we will establish a route for user-related operations:

```
const express = require('express');
const app = express();
require('dotenv').config(); // Load environment variables
const connectDB = require('./config/db'); // Import database configuration
const userRouter = require('./routes/userRoute'); // Import user routes


connectDB(); // Connect to database

/** Routes */
app.use(express.json());
app.use('/api/users', userRouter);

app.listen(8082, () => {
   console.log('Server is Running');
});
```

title: Login Route

```

userRouter.post("/login", async (req, res) => {
 try {
   const user = await User.findOne({ email: req.body.email });

   if (!user) {
     return res.send({
       success: false,
       message: "User does not exist. Please register.",
     });
   }

   // Simplified password validation (assuming passwords are stored in plain text,
which is not recommended)
   if (req.body.password !== user.password) {
     return res.send({
       success: false,
       message: "Sorry, invalid password entered!",
     });
   }

   res.send({
     success: true,
     message: "You've successfully logged in!",
   });
 } catch (error) {
   console.error(error);
   res.status(500).send({
     success: false,
     message: "An error occurred. Please try again later.",
   });
 }
});
```

Directly comparing plaintext passwords as shown above is not secure. Always use a secure method such as hashing (e.g., bcrypt) to store and compare passwords to protect user data We will be covering hashing in our future classes

Now as we have established Register and Login routes we will be connecting these routes to our Client so when we send data from the client side basically our login and resgister page these two Routes should work in correspondence