MERN-9: Project Part 5 - (Partner Page and Admin Approval)

Agenda
Creating the Theatre API
Creating the partners Page where they can List their theatres
Creating TheatresList for Admin where the Admin can approve and Deny the Theatres request
After Approval a Partner Theater can run shows

Now as we are done Adding Movies , The next step is to create the Partner's Page where a Partner can list their theatre

remember the roles that we provided Admin Partner and User in our userModel

Now add this code in your register page so your register page will now be able to handle to registration roles one for Partner one for user

```
<Form.Item className="d-block">
             <Button
               type="primary"
               block
               htmlType="submit"
               style={{ fontSize: "1rem", fontWeight: "600" }}
             >
               Register
```

```
          </Button>
        </Form.Item>
        <Form.Item
          label="Register as a Partner"
          htmlFor="role"
          name="role"
          className="d-block text-center"
          initialValue={false}
          rules={[{ required: true, message: "Please select an option!" }]}
        >
          <div className="d-flex justify-content-start">
            <Radio.Group name="radiogroup" className="flex-start">
              <Radio value={"partner"}>Yes</Radio>
              <Radio value={"user"}>No</Radio>
            </Radio.Group>
          </div>
        </Form.Item>
      </Form>
```

when the role is user we will provide user rights when the role is partner we will provide partner rights

Also have these routes added in the app.js

```
<Route
          path="/profile"
          element={
            <ProtectedRoute>
              <Profile />
            </ProtectedRoute>
          }
        />
        <Route
          path="/partner"
          element={
            <ProtectedRoute>
              <Partner />
            </ProtectedRoute>
          }
        />
```

Now when you will login as a Partner , The partner Profile should be able to list theatres

for that we will first create our theatre Model

title:Building the Theatre Model and Route

In your Server- Models - create a file with the theatreModel.js and define the schema and model for it

theatreModel.js code below

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

const theatreSchema = new mongoose.Schema(
 {
   name: {
     type: String,
     required: true,
   },
   address: {
     type: String,
     required: true,
   },
   phone: {
     type: Number,
     required: true,
   },
   email: {
    type: String,
    required: true,
   },
   owner: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "users",
   },
   isActive: {
    type: Boolean,
    default: false,
   },
 },
 { timestamps: true }
);

const Theatre = mongoose.model("theatres", theatreSchema);
module.exports = Theatre;
```

A reference to another document in the users collection, represented by an ObjectId.

title:Theatre CRUD Route

```
const router = require("express").Router();
const Theatre = require("../model/theatreModel");

router.post("/add-theatre", async (req, res) => {
 try {
   const newTheatre = new Theatre(req.body);
   await newTheatre.save();
   res.send({
    success: true,
    message: "New theatre has been added!",
   });
 } catch (err) {
   res.send({
    success: false,
     message: err.message,
   });
 }
});

// Update theatre
router.put("/update-theatre", async (req, res) => {
 try {
   await Theatre.findByIdAndUpdate(req.body.theatreId, req.body);
   // console.log(req.body.theatreId)
   res.send({
     success: true,
     message: "Theatre has been updated!",
   });
 } catch (err) {
   res.send({
     success: false,
     message: err.message,
   });
 }
});

// Delete theatre
router.delete("/delete-theatre/:theatreId", async (req, res) => {
 try {
   console.log("deleting theatre", req.params.theatreId);
   await Theatre.findByIdAndDelete(req.params.theatreId);
   res.send({
     success: true,
     message: "The theatre has been deleted!",
   });
 } catch (err) {
   res.send({
     success: false,
     message: err.message,
   });
 }
});


module.exports = router;
```

Update server.js

```
/** Routes */
app.use("/api/users", userRouter);
app.use("/api/movies", movieRouter);
app.use("/api/theatres", theatreRouter);
```

Getting Theatres

Now here As you see there can be different partners and every different partner will have their own set of theatres so for the partner we will have to fetch only the Theatres that they own

For the Admin because they should have all the theatres in the Admin table we will have have to fetch all the theatre and show who is the partner there

So we will need to have two get routes

```
// Get all theatres for Admin route
router.get("/get-all-theatres", async (req, res) => {
 try {
   const allTheatres = await Theatre.find().populate("owner");
   res.send({
     success: true,
     message: "All theatres fetched!",
     data: allTheatres,
   });
 } catch (err) {
   res.send({
     success: false,
     message: err.message,
   });
 }
});

// Get the theatres of a specific owner
router.get("/get-all-theatres-by-owner/:ownerId", async (req, res) => {
 try {
   const allTheatres = await Theatre.find({ owner: req.params.ownerId });
   res.send({
     success: true,
     message: "All theatres fetched successfully!",
     data: allTheatres,
   });
 } catch (err) {
   res.send({
     success: false,
     message: err.message,
   });
 }
});
```

Populating Owner Field: The .populate('owner') method is used to populate the owner field in the theatre documents with the related owner data from the Owner collection. This provides detailed information about the owner of each theatre.

title:Theatre Axios Instances

Now inside your api folder at the client side create AxiosInstaces for all the Theatre routes

```
import { axiosInstance } from ".";

export const addTheatre = async (payload) => {
 try {
   const response = await axiosInstance.post(
     "/api/theatres/add-theatre",
     payload
   );
   return response.data;
 } catch (err) {
   return err.response;
 }
};

// Get all theatres for the Admin route
export const getAllTheatresForAdmin = async () => {
 try {
   const response = await axiosInstance.get("/api/theatres/get-all-theatres");
   return response.data;
 } catch (err) {
   return err.response;
 }
};

// Get theatres of a specific owner
export const getAllTheatres = async (ownerId) => {
 try {
   const response = await axiosInstance.get(
     `/api/theatres/get-all-theatres-by-owner/${ownerId}`
   );
   return response.data;
 } catch (err) {
   return err.response;
 }
};

// Update Theatre
export const updateTheatre = async (payload) => {
 try {
   const response = await axiosInstance.put(
     "/api/theatres/update-theatre",
     payload
   );
   return response.data;
 } catch (err) {
   return err.resposne;
 }
```

```
};

// Delete Theatre
export const deleteTheatre = async (payload) => {
 try {
   const response = await axiosInstance.delete(
     `/api/theatres/delete-theatre/${payload}`
   );
   return response.data;
 } catch (err) {
   return err.response;
 }
};
```

title:Creating Partners Page

Now inside your Partner.js component Page , we will create a Tab which will show the theatre list and from there we can also add update delete theatres

In the Partners page will have a component by the name TheatreList

Create TheatreList Component

```
function TheatreList() {
 return <div>TheatreList</div>;
}

export default TheatreList;
```

add it in the tab of Partners page (index.jsx)

```
import { Tabs } from "antd";
import TheatreList from "./TheatreList";

const Partner = () => {
 const items = [
   {
     key: "1",
     label: "Theatres",
     children: <TheatreList />,
   },
 ];

 return (
   <>
     <h1>Partner Page</h1>
     <Tabs items={items} />
   </>
 );
};

export default Partner;
```

Inside the TheatreList we will use two more components by the name TheatreFormModel and DeleteTheatreModel

TheatreList.js

```
import React, { useEffect, useState } from "react";
import { Table, Button, message } from "antd";
import TheatreFormModal from "./TheatreFormModal";
import DeleteTheatreModal from "./DeleteTheatreModal";
import { EditOutlined, DeleteOutlined } from "@ant-design/icons";
import { getAllTheatres } from "../../api/theatres";
import { useSelector, useDispatch } from "react-redux";
import { ShowLoading, HideLoading } from "../../redux/loaderSlice";
```

State variables

```
const TheatreList = () => {
 const { user } = useSelector((state) => state.users);
 const [isModalOpen, setIsModalOpen] = useState(false);
 const [isDeleteModalOpen, setIsDeleteModalOpen] = useState(false);
 const [isShowModalOpen, setIsShowModalOpen] = useState(false);
 const [selectedTheatre, setSelectedTheatre] = useState(null);
 const [formType, setFormType] = useState("add");
 const [theatres, setTheatres] = useState(null);
 const dispatch = useDispatch();
};
```

Get data for all theatres for that partner

```
import React, { useEffect, useState } from "react";
import { Table, Button, message } from "antd";
import TheatreFormModal from "./TheatreFormModal";
import DeleteTheatreModal from "./DeleteTheatreModal";
import { EditOutlined, DeleteOutlined } from "@ant-design/icons";
import { getAllTheatres } from "../../api/theatres";
import { useSelector, useDispatch } from "react-redux";
import { ShowLoading, HideLoading } from "../../redux/loaderSlice";
import ShowModal from "./ShowModal";

const TheatreList = () => {
 const { user } = useSelector((state) => state.user);
 const [isModalOpen, setIsModalOpen] = useState(false);
 const [isDeleteModalOpen, setIsDeleteModalOpen] = useState(false);
 const [isShowModalOpen, setIsShowModalOpen] = useState(false);
 const [selectedTheatre, setSelectedTheatre] = useState(null);
 const [formType, setFormType] = useState("add");
 const [theatres, setTheatres] = useState(null);
 const dispatch = useDispatch();

 const getData = async () => {
   try {
     dispatch(ShowLoading());
     const response = await getAllTheatres(user._id );
     if (response.success) {
       const allTheatres = response.data;
       // console.log(allTheatres);
       setTheatres(
         allTheatres.map(function (item) {
           return { ...item, key: `theatre${item._id}` };
         })
       );
     } else {
       message.error(response.message);
     }
     dispatch(HideLoading());
   } catch (err) {
     dispatch(HideLoading());
     message.error(err.message);
   }
 };
useEffect(() => {
   getData();
 }, []);
};
```

Adding column definitions

```
const columns = [
   {
     title: "Name",
     dataIndex: "name",
     key: "name",
   },
   {
     title: "Address",
     dataIndex: "address",
     key: "address",
   },
   {
     title: "Phone Number",
     dataIndex: "phone",
     key: "phone",
   },
   {
     title: "Email",
     dataIndex: "email",
     key: "email",
   },
   {
     title: "Status",
     dataIndex: "status",
     render: (status, data) => {
       if (data.isActive) {
         return `Approved`;
       } else {
         return `Pending/ Blocked`;
       }
     },
   },
   {
     title: "Action",
     dataIndex: "action",
     render: (text, data) => {
       return (
         <div className="d-flex align-items-center gap-10">
           <Button
             onClick={() => {
               setIsModalOpen(true);
               setFormType("edit");
               setSelectedTheatre(data);
             }}
           >
             <EditOutlined />
           </Button>
           <Button
             onClick={() => {
               setIsDeleteModalOpen(true);
               setSelectedTheatre(data);
             }}
           >
             <DeleteOutlined />
           </Button>
           {data.isActive && (
             <Button
               onClick={() => {
                 setIsShowModalOpen(true);
                 setSelectedTheatre(data);
               }}
             >
               + Shows
             </Button>
           )}
         </div>
       );
     },
   },
 ];
```

Markup

```
return (
   <>
     <div className="d-flex justify-content-end">
       <Button
         type="primary"
         onClick={() => {
           setIsModalOpen(true);
           setFormType("add");
         }}
       >
         Add Theatre
       </Button>
     </div>
     <Table dataSource={theatres} columns={columns} />
     {isModalOpen && (
       <TheatreFormModal
```

```
          isModalOpen={isModalOpen}
          selectedTheatre={selectedTheatre}
          setSelectedTheatre={setSelectedTheatre}
          setIsModalOpen={setIsModalOpen}
          formType={formType}
          getData={getData}
        />
      )}
      {isDeleteModalOpen && (
        <DeleteTheatreModal
          isDeleteModalOpen={isDeleteModalOpen}
          selectedTheatre={selectedTheatre}
          setIsDeleteModalOpen={setIsDeleteModalOpen}
          setSelectedTheatre={setSelectedTheatre}
          getData={getData}
        />
      )}
   </>
 );
export default TheatreList;
```

TheatreFormModal.js

Let us copy from MovieFormModal and make changes

```
import { Col, Modal, Row, Form, Input, Select, Button, message } from "antd";
import TextArea from "antd/es/input/TextArea";
import { ShowLoading, HideLoading } from "../../redux/loaderSlice";
import { useDispatch, useSelector } from "react-redux";
import { addTheatre, updateTheatre } from "../../api/theatre";
import moment from "moment";

const TheatreForm = ({
 isModalOpen,
 setIsModalOpen,
 selectedTheatre,
 setSelectedTheatre,
 formType,
 getData,
}) => {
 const dispatch = useDispatch();
 const { user } = useSelector((state) => state.users);

 const onFinish = async (values) => {
   try {
     dispatch(ShowLoading());
     let response = null;
     if (formType === "add") {
       response = await addTheatre({ ...values, owner: user._id });
     } else {
       values.theatreId = selectedTheatre._id;
       response = await updateTheatre(values);
     }
     if (response.success) {
       getData();
       message.success(response.message);
       setIsModalOpen(false);
     } else {
       message.error(response.message);
     }
     setSelectedTheatre(null);
     dispatch(HideLoading());
   } catch (err) {
     dispatch(HideLoading());
     message.error(err.message);
   }
 };

 const handleCancel = () => {
   setIsModalOpen(false);
   setSelectedTheatre(null);
 };
```

```
return (
  <Modal
    centered
    title={formType === "add" ? "Add Theatre" : "Edit Theatre"}
    open={isModalOpen}
    onCancel={handleCancel}
    width={800}
    footer={null}
  >
    <Form
      layout="vertical"
      initialValues={selectedTheatre}
      onFinish={onFinish}
    >
      <Row gutter={{ xs: 6, sm: 10, md: 12, lg: 16 }}>
        <Col span={24}>
          <Form.Item
            label="Theatre Name"
            name="name"
            rules={[{ required: true, message: "Theatre name is required!" }]}
          >
            <Input placeholder="Enter the Theatre name" />
          </Form.Item>
        </Col>
        <Col span={24}>
          <Form.Item
            label="Theatre Address"
            name="address"
            rules={[{ required: true, message: "Theatre name is required!" }]}
          >
            <TextArea
              id="address"
              rows="3"
              placeholder="Enter the description"
            />
          </Form.Item>
        </Col>
        <Col span={24}>
          <Row gutter={{ xs: 6, sm: 10, md: 12, lg: 16 }}>
            <Col span={12}>
              <Form.Item
                label="Email"
                name="email"
                rules={[{ required: true, message: "Email is required!" }]}
              >
                <Input
                  type="email"
                  id="email"
                  placeholder="Enter the email"
                />
              </Form.Item>
            </Col>
            <Col span={12}>
              <Form.Item
                label="Phone number"
                name="phone"
                rules={[
                  { required: true, message: "Phone number is required!" },
                ]}
              >
                <Input
                  type="number"
                  id="number"
                  placeholder="Enter the contact number"
                />
              </Form.Item>
            </Col>
          </Row>
        </Col>
      </Row>
      <Form.Item>
        <Button
          block
          type="primary"
          htmlType="submit"
```

```
           style={{ fontSize: "1rem", fontWeight: "600" }}
         >
           Submit the Data
         </Button>
         <Button className="mt-3" block onClick={handleCancel}>
           Cancel
         </Button>
       </Form.Item>
     </Form>
   </Modal>
 );
};

export default TheatreForm;
```

DeleteTheatreModel

```
import { Modal, message } from "antd";
import { deleteTheatre } from "../../api/theatre";
import { ShowLoading, HideLoading } from "../../redux/loaderSlice";
import { useDispatch } from "react-redux";

const DeleteTheatreModal = ({
 isDeleteModalOpen,
 setIsDeleteModalOpen,
 selectedTheatre,
 setSelectedTheatre,
 getData,
}) => {
 const dispatch = useDispatch();
 const handleOK = async () => {
   try {
     dispatch(ShowLoading());
     const theatreId = selectedTheatre._id;
     const response = await deleteTheatre(theatreId);
     if (response.success) {
       message.success(response.message);
       getData();
     } else {
       message.error(response.message);
     }
     setSelectedTheatre(null);
     setIsDeleteModalOpen(false);
     dispatch(HideLoading());
   } catch (err) {
     message.error(err.message);
     dispatch(HideLoading());
     setIsDeleteModalOpen(false);
   }
 };

 const handleCancel = () => {
   setIsDeleteModalOpen(false);
   setSelectedTheatre(null);
 };
 return (
   <Modal
     centered
     title="Delete Theatre"
     open={isDeleteModalOpen}
     onCancel={handleCancel}
     onOk={handleOK}
   >
     <p className="pt-3 fs-18">
       Are you sure you want to delete this theatre?
     </p>
     <p className="pb-3 fs-18">
       This action can't be undone and you'll lose this theatre data.
     </p>{" "}
   </Modal>
 );
};
export default DeleteTheatreModal;
```

title:Admin Approval

Now go the admin page and there we have a TheatreTable.js component where we can recieve theatre request and can approve and deny it , let's start buidling that

TheatresTable.js in admin Directory

```
import { useState, useEffect } from "react";
import { getAllTheatresForAdmin, updateTheatre } from "../../api/theatre";
import { ShowLoading, HideLoading } from "../../redux/loaderSlice";
import { useDispatch } from "react-redux";
import { message, Button, Table } from "antd";

const TheatresTable = () => {
 const [theatres, setTheatres] = useState([]);
 const dispatch = useDispatch();

 const getData = async () => {
   try {
     dispatch(ShowLoading());
     const response = await getAllTheatresForAdmin();
     if (response.success) {
       const allTheatres = response.data;
       setTheatres(
         allTheatres.map(function (item) {
           return { ...item, key: `theatre${item._id}` };
         })
       );
     } else {
       message.error(response.message);
     }
     dispatch(HideLoading());
   } catch (err) {
     dispatch(HideLoading());
     message.error(err.message);
   }
 };

 const handleStatusChange = async (theatre) => {
   try {
     dispatch(ShowLoading);
     let values = {
       ...theatres,
       theatreId: theatre._id,
       isActive: !theatre.isActive,
     };
     const response = await updateTheatre(values);
     console.log(response, theatre);
     if (response.success) {
       message.success(response.message);
       getData();
     }
     dispatch(HideLoading);
   } catch (err) {
     dispatch(HideLoading);
     message.error(err.message);
   }
 };

 const columns = [
   {
     title: "Name",
     dataIndex: "name",
     key: "name",
   },
   {
     title: "Address",
     dataIndex: "address",
     key: "address",
   },
   {
     title: "Owner",
     dataIndex: "owner",
     render: (text, data) => {
       return data.owner && data.owner.name;
     },
   },
   {
     title: "Phone Number",
     dataIndex: "phone",
     key: "phone",
   },
   {
     title: "Email",
     dataIndex: "email",
     key: "email",
   },
   {
     title: "Status",
     dataIndex: "status",
     render: (status, data) => {
       if (data.isActive) {
         return "Approved";
       } else {
         return "Pending/ Blocked";
       }
     },
   },
   {
     title: "Action",
     dataIndex: "action",
     render: (text, data) => {
       return (
         <div className="d-flex align-items-center gap-10">
           {data.isActive ? (
             <Button onClick={() => handleStatusChange(data)}>Block</Button>
           ) : (
             <Button onClick={() => handleStatusChange(data)}>Approve</Button>
           )}
         </div>
       );
     },
   },
 ];

 useEffect(() => {
   getData();
 }, []);

 // console.log(theatres.length > 0 && theatres);

 return (
   <>
     {theatres && theatres.length > 0 && (
       <Table dataSource={theatres} columns={columns} />
     )}
   </>
 );
};

export default TheatresTable;
```