Group 1: Project Setup & UI Components --- TEAM KAMRANx

Objective: Set up the project and create reusable UI components for a cohesive interface.

1. **Set up the project**:
   - Run `npx create-react-app library-mis.`
   - Install required dependencies:

```
bash
Copy code
npm install @mui/material @mui/icons-material react-hook-form yup
```

2. **Create a Header Component** for app-wide navigation.

```
import React from 'react';
import PropTypes from 'prop-types';
import { AppBar, Toolbar, Typography, Button, Box } from '@mui/material';
import { useNavigate } from 'react-router-dom';

function Header({ title, routes }) {
    const navigate = useNavigate();

    return (
        <AppBar position="static">
            <Toolbar sx={{ display: 'flex', justifyContent: 'center', alignItems: 'center' }}>
                {/* Header Title */}
                <Typography variant="h6" sx={{ flexGrow: 1, textAlign: 'center' }}>
                    {title}
                </Typography>

                {/* Centered Buttons */}
                <Box sx={{ display: 'flex', gap: 2 }}>
                    {routes.map((route) => (
                        <Button
                            key={route.path}
                            color="inherit"
                            onClick={() => navigate(route.path)}
                            sx={{
                                textTransform: 'capitalize',
                                fontWeight: 'bold',
                                '&:hover': {
                                    backgroundColor: 'rgba(255, 255, 255, 0.1)',
                                },
                            }}
```

```
                        >
                            {route.label}
                        </Button>
                    ))}
                </Box>
            </Toolbar>
        </AppBar>
    );
}

export default Header;

```

45. **Set Up Routing and Main Layout**:
    - Use React Router to handle navigation between pages.

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import Header from './components/Header';
import Home from './pages/Home';
import Books from './pages/Books';
import Authors from './pages/Authors';
import Categories from './pages/Categories';
import Users from './pages/Users';
import Reservations from './pages/Reservations';

const routes=[
  { path:'/',label:"Home"},
  { path:'/books',label:"Books"},
  { path:'/authors',label:"Authors"},
  { path:'/categories',label:"Categories"},
  { path:'/users',label:"Users"},
  { path:'/reservations',label:"Reservations"},
]
function App() {
    return (
        <Router>
            <Header title="Library MIS" routes={routes}/>

            <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/books" element={<Books /> }  />
            <Route path="/authors" element={<Authors />} />
            <Route path="/categories" element={<Categories />}  />
            <Route path="/users" element={<Users />}  />
            <Route path="/reservations" element={<Reservations />}  />
          </Routes>
```

```
77.
78.
79.         </Router>
80.     );
81. }
82.
83. export default App;
```

Team Samiullah: Books & Categories Entities

Objective: Build forms and lists for managing books and categories, using `react-hook-form` and `yup` for validation.

1. **Books Entity**:
    - Create a **Book List** and **Book Form** component with form validation.

src/pages/Books.js

```
import React from 'react';

function Books() {
    // Static data to display books
    const books = [
        { id: 1, title: 'Book One', author: 'Author A', category: 'Fiction' },
        { id: 2, title: 'Book Two', author: 'Author B', category: 'Non-Fiction'
},
    ];

    return (
        <div>
            <h2>Books</h2>
            <ul>
                {books.map(book => (
                    <li key={book.id}>{book.title} - {book.author}</li>
                ))}
            </ul>
        </div>
    );
}

export default Books;
```

Book Form Component:

- Create a form to add or edit books, using `react-hook-form` and `yup` for validation.

```
import React from 'react';
import { useForm } from 'react-hook-form';
```

```jsx
import { TextField, Button } from '@mui/material';
import { yupResolver } from '@hookform/resolvers/yup';
import * as yup from 'yup';

const schema = yup.object().shape({
    title: yup.string().required("Title is required"),
    author: yup.string().required("Author is required"),
});

function BookForm({ onSubmit }) {
    const { register, handleSubmit, formState: { errors } } = useForm({
        resolver: yupResolver(schema)
    });

    return (
        <form onSubmit={handleSubmit(onSubmit)}>
            <TextField
                label="Title"
                {...register("title")}
                error={!!errors.title}
                helperText={errors.title?.message}
            />
            <TextField
                label="Author"
                {...register("author")}
                error={!!errors.author}
                helperText={errors.author?.message}
            />
            <Button type="submit" variant="contained">Submit</Button>
        </form>
    );
}

export default BookForm;
```

Group 3: Authors Entity -----TEAM MOHAMMAD ALI

Objective: Build components to manage authors, including a list and form with validation.

1. **Authors Page**:
    - List authors with sample data for now, similar to the Books page.

src/pages/Authors.js

```
import React from 'react';

function Authors() {
    const authors = [
        { id: 1, name: 'Author A' },
        { id: 2, name: 'Author B' },
    ];

    return (
        <div>
            <h2>Authors</h2>
            <ul>
                {authors.map(author => (
                    <li key={author.id}>{author.name}</li>
                ))}
            </ul>
        </div>
    );
}

export default Authors;
```

Author Form Component:

- Set up a form similar to `BookForm`, validating the author's name.

```
import React from 'react';
import { useForm } from 'react-hook-form';
import { TextField, Button } from '@mui/material';
import { yupResolver } from '@hookform/resolvers/yup';
import * as yup from 'yup';

const schema = yup.object().shape({
    name: yup.string().required("Author name is required"),
});

function AuthorsForm({ onAddAuthor }) {
    const { register, handleSubmit, formState: { errors } } = useForm({
        resolver: yupResolver(schema)
    });

    const onSubmit = (data) => {
        onAddAuthor(data);
    };

    return (
        <form onSubmit={handleSubmit(onSubmit)}>
            <TextField
                label="Author Name"
                {...register("name")}
                error={!!errors.name}
                helperText={errors.name?.message}
            />
            <Button type="submit" variant="contained">Add Author</Button>
        </form>
    );
}

export default AuthorsForm;

```

New Group: Categories Management ----- MOHAMMAD REZA

This group will create:

1. **CategoriesForm** component to add new categories.
2. **CategoriesTable** component to display a list of categories.

Task 1: Categories Form

- **CategoriesForm.js**: Use `react-hook-form` and `yup` to validate the category name.

```
import React from 'react';
import { useForm } from 'react-hook-form';
import { TextField, Button } from '@mui/material';
import { yupResolver } from '@hookform/resolvers/yup';
import * as yup from 'yup';

const schema = yup.object().shape({
    name: yup.string().required("Category name is required"),
});

function CategoriesForm({ onAddCategory }) {
    const { register, handleSubmit, formState: { errors } } = useForm({
        resolver: yupResolver(schema)
    });

    const onSubmit = (data) => {
        onAddCategory(data);
    };

    return (
        <form onSubmit={handleSubmit(onSubmit)}>
            <TextField
                label="Category Name"
                {...register("name")}
                error={!!errors.name}
                helperText={errors.name?.message}
            />
            <Button type="submit" variant="contained">Add
Category</Button>
        </form>
    );
}

export default CategoriesForm;
```

Task 2: Categories Table

- **CategoriesTable.js**: Display the list of categories using an MUI table component.

```
import React from 'react';
import { Table, TableBody, TableCell, TableHead, TableRow } from
'@mui/material';

function CategoriesTable() {
  const [categories, setCategories] = useState([
    { id: 1, name: 'Fiction' },
    { id: 2, name: 'Non-fiction' }
]);
    return (
        <Table>
            <TableHead>
                <TableRow>
                    <TableCell>ID</TableCell>
                    <TableCell>Category Name</TableCell>
                </TableRow>
            </TableHead>
            <TableBody>
                {categories.map((category, index) => (
                    <TableRow key={index}>
                        <TableCell>{index + 1}</TableCell>
                        <TableCell>{category.name}</TableCell>
                    </TableRow>
                ))}
            </TableBody>
        </Table>
    );
}

export default CategoriesTable;

```

Group 4: Users & Reservations Entities ---------- TEAM SULIMAN

Objective: Build forms for users and book reservations.

1. **Users Page**:
    - Create a static display for users, and a form to add new users.

src/pages/Users.js

```
import React from 'react';

function Users() {
    const users = [
        { id: 1, name: 'User A' },
        { id: 2, name: 'User B' },
    ];

    return (
        <div>
            <h2>Users</h2>
            <ul>
                {users.map(user => (
                    <li key={user.id}>{user.name}</li>
                ))}
            </ul>
        </div>
    );
}

export default Users;
```

\

Group 5th

User Management: Create a user form and table to manage users.

```jsx
import React from 'react';
import { useForm } from 'react-hook-form';
import { TextField, Button } from '@mui/material';
import { yupResolver } from '@hookform/resolvers/yup';
import * as yup from 'yup';

// Define validation schema with Yup
const userSchema = yup.object({
  name: yup.string().required("Name is required"),
  email: yup.string().email("Invalid email").required("Email is required"),
  phone: yup.string().required("Phone number is required"),
});

const UserForm = ({ onSubmit }) => {
  const { register, handleSubmit, formState: { errors }, reset } = useForm({
    resolver: yupResolver(userSchema),
  });

  const submitForm = (data) => {
    onSubmit(data);
    reset();  // Clear the form after submission
  };

  return (
    <form onSubmit={handleSubmit(submitForm)}>
      <TextField
        label="Name"
        {...register("name")}
        variant="outlined"
        error={!!errors.name}
        helperText={errors.name?.message}
        fullWidth
        margin="normal"
      />
      <TextField
        label="Email"
        {...register("email")}
        variant="outlined"
        error={!!errors.email}
```

```
        helperText={errors.email?.message}
        fullWidth
        margin="normal"
      />
      <TextField
        label="Phone"
        {...register("phone")}
        variant="outlined"
        error={!!errors.phone}
        helperText={errors.phone?.message}
        fullWidth
        margin="normal"
      />
      <Button type="submit" variant="contained" color="primary">
        Add User
      </Button>
    </form>
  );
};

export default UserForm;
```

Task 2: ----------TEAM KHALID

The `UserTable` component will display the list of users in a table format.

File: `src/components/UserTable.js`

```
import React from 'react';
import { Table, TableBody, TableCell, TableContainer, TableHead, TableRow, Paper
} from '@mui/material';

const UserTable = ({ users }) => {
export const users = [
  { id: 1, name: "Alice Johnson", email: "alice@example.com", phone: "123-456-
7890" },
  { id: 2, name: "Bob Smith", email: "bob@example.com", phone: "987-654-3210" },
];

  return (
    <TableContainer component={Paper}>
      <Table>
        <TableHead>
          <TableRow>
            <TableCell>Name</TableCell>
            <TableCell>Email</TableCell>
            <TableCell>Phone</TableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {users.map((user, index) => (
            <TableRow key={index}>
              <TableCell>{user.name}</TableCell>
              <TableCell>{user.email}</TableCell>
              <TableCell>{user.phone}</TableCell>
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </TableContainer>
  );
};

export default UserTable;
```

Group 6:

Task 2: Reservation Management ---------- TEAM USMAN SULTAN

Goal: Create a form to add reservations and display them in a table.

Step 2.1: Create the Reservation Form Component

The `ReservationForm` will allow users to add reservation details.

File: `src/components/ReservationForm.js`

```
import React from 'react';
import { useForm } from 'react-hook-form';
import { TextField, Button } from '@mui/material';
import { yupResolver } from '@hookform/resolvers/yup';
import * as yup from 'yup';

// Define validation schema with Yup
const reservationSchema = yup.object({
  date: yup.string().required("Date is required"),
  time: yup.string().required("Time is required"),
  partySize: yup.number().min(1, "At least 1 person required").required("Party
size is required"),
});

const ReservationForm = ({ onSubmit }) => {
  const { register, handleSubmit, formState: { errors }, reset } = useForm({
    resolver: yupResolver(reservationSchema),
  });

  const submitForm = (data) => {
    onSubmit(data);
    reset();  // Clear the form after submission
  };

  return (
    <form onSubmit={handleSubmit(submitForm)}>
      <TextField
        label="Date"
        type="date"
        {...register("date")}
        InputLabelProps={{ shrink: true }}
        error={!!errors.date}
        helperText={errors.date?.message}
```

```
        fullWidth
        margin="normal"
      />
      <TextField
        label="Time"
        type="time"
        {...register("time")}
        InputLabelProps={{ shrink: true }}
        error={!!errors.time}
        helperText={errors.time?.message}
        fullWidth
        margin="normal"
      />
      <TextField
        label="Party Size"
        type="number"
        {...register("partySize")}
        error={!!errors.partySize}
        helperText={errors.partySize?.message}
        fullWidth
        margin="normal"
      />
      <Button type="submit" variant="contained" color="primary">
        Add Reservation
      </Button>
    </form>
  );
};

export default ReservationForm;
```

Task 2 : Create a reservation table

```jsx
import React from 'react';
import { Table, TableBody, TableCell, TableContainer, TableHead, TableRow, Paper
} from '@mui/material';

const ReservationTable = () => {
  export const reservations = [
    { id: 1, date: "2024-11-10", time: "18:30", partySize: 4 },
    { id: 2, date: "2024-11-12", time: "19:00", partySize: 2 },
  ];

  return (
    <TableContainer component={Paper}>
      <Table>
        <TableHead>
          <TableRow>
            <TableCell>Date</TableCell>
            <TableCell>Time</TableCell>
            <TableCell>Party Size</TableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {reservations.map((reservation, index) => (
            <TableRow key={index}>
              <TableCell>{reservation.date}</TableCell>
              <TableCell>{reservation.time}</TableCell>
              <TableCell>{reservation.partySize}</TableCell>
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </TableContainer>
  );
};

export default ReservationTable;
```

Team ABDUL HAI

Create an Authentication Context

This context will help us manage the authenticated state across the app.

AuthContext.js

```
import { createContext, useContext, useState } from 'react';

const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
  const [isAuthenticated, setIsAuthenticated] = useState(false);

  const login = () => setIsAuthenticated(true);
  const logout = () => setIsAuthenticated(false);

  return (
    <AuthContext.Provider value={{ isAuthenticated, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};

export const useAuth = () => useContext(AuthContext);
```

```jsx
import React from 'react';
import { useForm } from 'react-hook-form';
import { yupResolver } from '@hookform/resolvers/yup';
import * as yup from 'yup';
import { TextField, Button, Box, Typography, Container } from '@mui/material';
import { useAuth } from './AuthContext';
import { useNavigate } from 'react-router-dom';

// Yup schema for validation
const schema = yup.object().shape({
  email: yup.string().email('Invalid email').required('Email is required'),
  password: yup.string().min(6, 'Password must be at least 6
characters').required('Password is required'),
});

const SignIn = () => {
  const { login } = useAuth();
  const navigate = useNavigate();
  const { register, handleSubmit, formState: { errors } } = useForm({
    resolver: yupResolver(schema),
  });

  const onSubmit = (data) => {
    console.log(data);
    login(); // Set authentication to true
    navigate('/'); // Redirect to home or protected route
  };

  return (
    <Container component="main" maxWidth="xs">
      <Box
        sx={{
          marginTop: 8,
          display: 'flex',
          flexDirection: 'column',
          alignItems: 'center',
        }}
      >
        <Typography component="h1" variant="h5">Sign In</Typography>
        <Box component="form" onSubmit={handleSubmit(onSubmit)} sx={{ mt: 1 }}>
          <TextField
            label="Email"
            fullWidth
            margin="normal"
            {...register('email')}
```

```
            error={!!errors.email}
            helperText={errors.email?.message}
          />
          <TextField
            label="Password"
            fullWidth
            margin="normal"
            type="password"
            {...register('password')}
            error={!!errors.password}
            helperText={errors.password?.message}
          />
          <Button type="submit" fullWidth variant="contained" sx={{ mt: 3, mb: 2
}}>
            Sign In
          </Button>
        </Box>
      </Box>
    </Container>
  );
};

export default SignIn;
```

TEAM SAMIULLAH SHIRANI

Set Up Protected Routes and Conditional Navbar Rendering

Create a wrapper for protected routes and conditionally render the navbar if the user is authenticated.

ProtectedRoute.js

```
import React from 'react';
import { Navigate, Outlet } from 'react-router-dom';
import { useAuth } from './AuthContext';
import NavComponent from './NavComponent';

const ProtectedRoute = () => {
  const { isAuthenticated } = useAuth();

  return (
    <>
      {isAuthenticated ? (
        <>
          <NavComponent />
          <Outlet />
        </>
      ) : (
        <Navigate to="/signin" />
      )}
    </>
  );
};

export default ProtectedRoute;
```