Book Chair: Office Seat Booking System

Technical Documentation and Project Report

Document Control

- Project Name: Book Chair

- Document Type: Technical Documentation and Project Report

- Version: 1.0

---

Table of Contents

1. Executive Summary

2. Introduction

3. System Architecture

4.System Components

Database Design

5. Implementation Details

6. Features and Functionality

7. Future Enhancements

8. Conclusion

---

1. Executive Summary

Book Chair is a comprehensive office seat booking system designed to streamline workspace management in modern offices. The system facilitates flexible working arrangements by providing employees with tools to reserve office seats and manage their hybrid work schedule efficiently.

The system is built using modern web technologies, implementing a MERN stack variation with TypeScript integration, ensuring type safety and improved development experience.

---

2. Introduction

2.1 Project Overview

Book Chair addresses the growing need for flexible workspace management in contemporary office environments. The system provides a robust platform for employees to manage their workplace presence effectively.

2.2 Objectives

- Implement a user-friendly seat booking system

- Facilitate efficient workspace utilization

- Enable hybrid work schedule management

- Provide clock in/out functionality

- Ensure secure user authentication and profile management

3. System Architecture

3.1Technical Stack

The application follows a modern full-stack architecture utilizing the following technologies:

3.2 Frontend Technologies

- React: A JavaScript library for building user interfaces

- TypeScript: Adds static typing to JavaScript, enhancing code quality and

developer experience

- Vite: A modern frontend build tool that offers superior performance and improved developer experience

- Key Frontend Libraries:

* React Router for client-side routing
* React Query for state management and API calls
* TypeScript for type safety
* Vite for development and build optimization

3.3 Backend Technologies

- Node.js: JavaScript runtime for server-side development

- Express.js: Web application framework for Node.js

- MongoDB: NoSQL database for flexible data storage

- Key Backend Features:

* RESTful API architecture
* JWT-based authentication
* Express middleware for request processing
* MongoDB Atlas for database hosting

4. System Components

4.1 Frontend Architecture

1. Presentation Layer

- React components for UI rendering

- TypeScript interfaces for type definitions

- Responsive design implementation

- State management using React hooks and context

2. Application Layer

- API integration services

- Authentication handlers

- Form validation logic

- Date and time management utilities

4.2 Backend Architecture

1. API Layer

- Express routes for HTTP endpoints

- Authentication middleware

- Request validation

- Error handling middleware

2. Business Logic Layer

- User management services

- Booking logic implementation

- Notification services

- Authentication services

3. Data Access Layer

- MongoDB schemas

- Database queries

- Data validation

- Index optimization

5.Database Design - MongoDB Collections

1. Users Collection

```javascript

{

\_id: ObjectId,

first\_name: String,

last\_name: String,

team: String,

squad: String,

email: String,

password: String (hashed),

role: String,

avatar\_url: String,

createdAt: Date,

updatedAt: Date

}

```

2. Bookings Collection

```javascript

{

\_id: ObjectId,

user: ObjectId,

desk: ObjectId,

date: Date,

duration: Number,

referenceNumber: String,

status: String,

createdAt: Date,

updatedAt: Date

}

```

3. Desks Collection

```javascript

{

\_id: ObjectId,

name: String,

shortName: String,

description: String,

direction: String,

status: String,

createdAt: Date,

updatedAt: Date

}

```

4. Rooms Collection

```javascript

{

\_id: ObjectId,

name: String,

desks: ObjectId[],

type: String,

description: String,

images: String[],

status: String,

createdAt: Date,

updatedAt: Date

}

```

5. Attendances Collection

```javascript

{

\_id: ObjectId,

userId: ObjectId,

bookingId: ObjectId,

clockIn: Date,

clockOut: Date,

isClockIn: Boolean,

isClockOut: Boolean

}

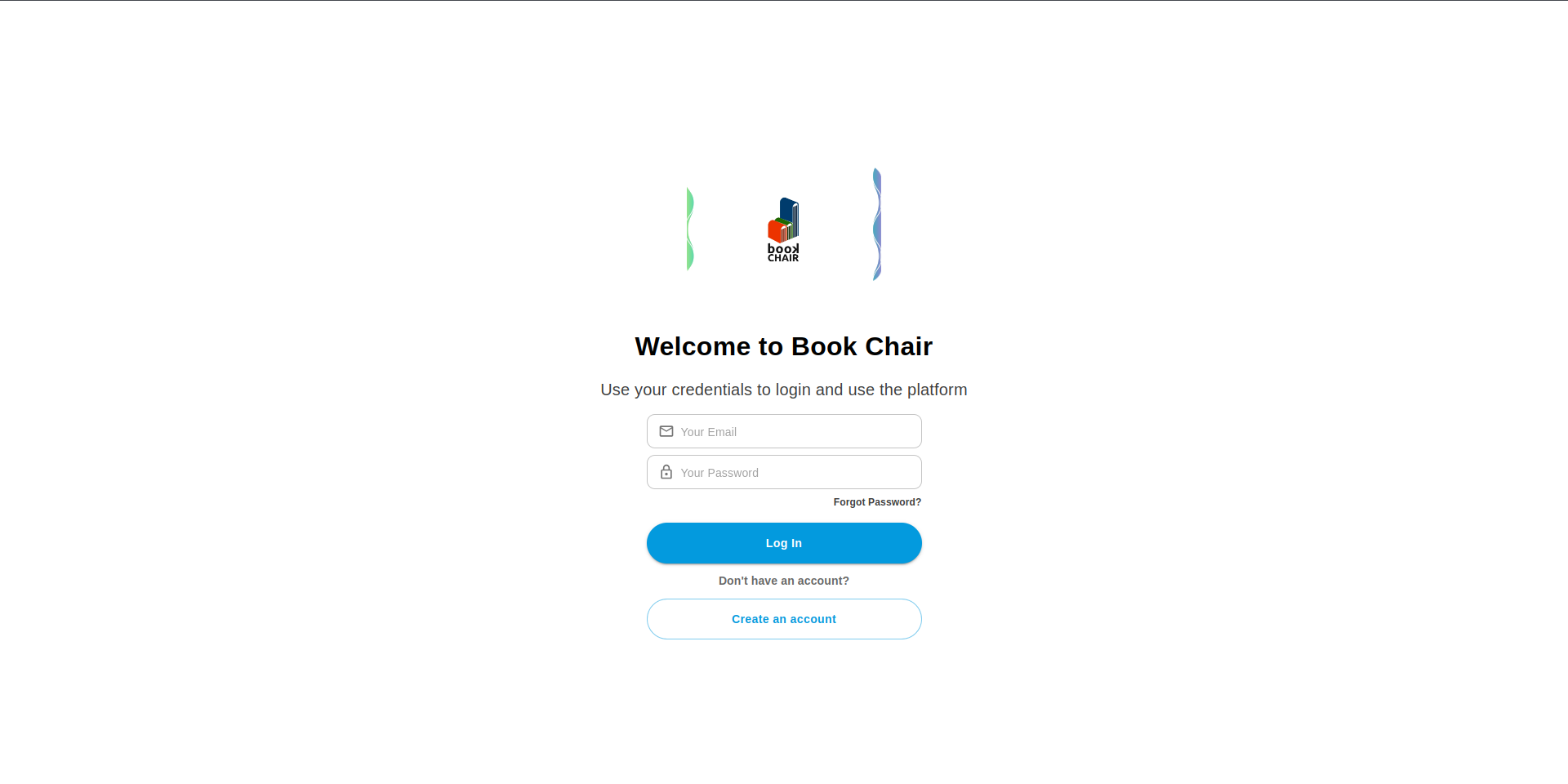
```

---

6.Implementation Details

* Front-end Implementation

1. Authentication System - Login Feature



The login system implementation follows a modern React architecture using Type-Script, Material-UI, and Redux Toolkit for state management. Here's a detailed breakdown of the implementation:

1.1 Component Structure

1. HomePage Component

- Serves as the container for the login page

- Implements responsive layout using Material-UI's Container and Box components

- Comprises two main components: LoginImages and LoginForm

2. LoginForm Component

- Implements the main login form functionality

- Uses Formik for form management and validation

- Integrates with Redux for state management

- Features responsive design for multiple screen sizes

1.2Key Technical Features

1. Form Management and Validation

first\_name: string;

last\_name: string;

team: string;

squad: string;

email: string;

password: string;

role: userRole;

avatar\_url: string;

}

const loginThunk = createAsyncThunk(

"auth/login",

async (userData: IUserLogin, { rejectWithValue }) => {

// Implementation details

}

);

```

- Uses Redux Toolkit for state management

- Implements async login handling with createAsyncThunk

- Manages user authentication state

- Handles token storage and management

4. UI Components and Styling

```typescript

<TextField

id="email"

type="email"

placeholder="Your Email"

variant="outlined"

sx={{

height: "42px",

width: {

sm: "100%",

lg: "337px",

},

maxWidth: "337px",

// Additional styling

}}

// Additional props

/>

```

- Uses Material-UI components for consistent design

- Implements custom styling using Material-UI's sx prop

- Features responsive input fields and buttons

- Includes icon integration with Material-UI icons

1.3 Authentication Flow

1. \*\*Login Process\*\*

- User enters email and password

- Form validation occurs in real-time

- On submission, credentials are sent to the backend

- Success/failure feedback provided via toast notifications

- Automatic redirection to /room on successful login

2. \*\*State Updates\*\*

- Redux store updates with user information

- Access token stored in localStorage

- User session maintained through token

3. \*\*Error Handling\*\*

- Form validation errors displayed inline

- API errors handled with user-friendly messages

- Network errors managed with appropriate feedback

1.4 Security Features

1. \*\*Token Management\*\*

```typescript

localStorage.setItem("token", accessToken);

```

- Secure token storage in localStorage

- Automatic token inclusion in subsequent requests

- Token validation and refresh mechanisms

2. \*\*Form Security\*\*

- Input sanitization

- Password field masking

- Protected routes implementation

- CSRF protection through withCredentials

1.5 User Experience Enhancements

1. \*\*Visual Feedback\*\*

- Loading states during form submission

- Error messages for invalid inputs

- Success notifications for successful login

- Smooth transitions and animations

2. \*\*Navigation\*\*

- "Forgot Password" link integration

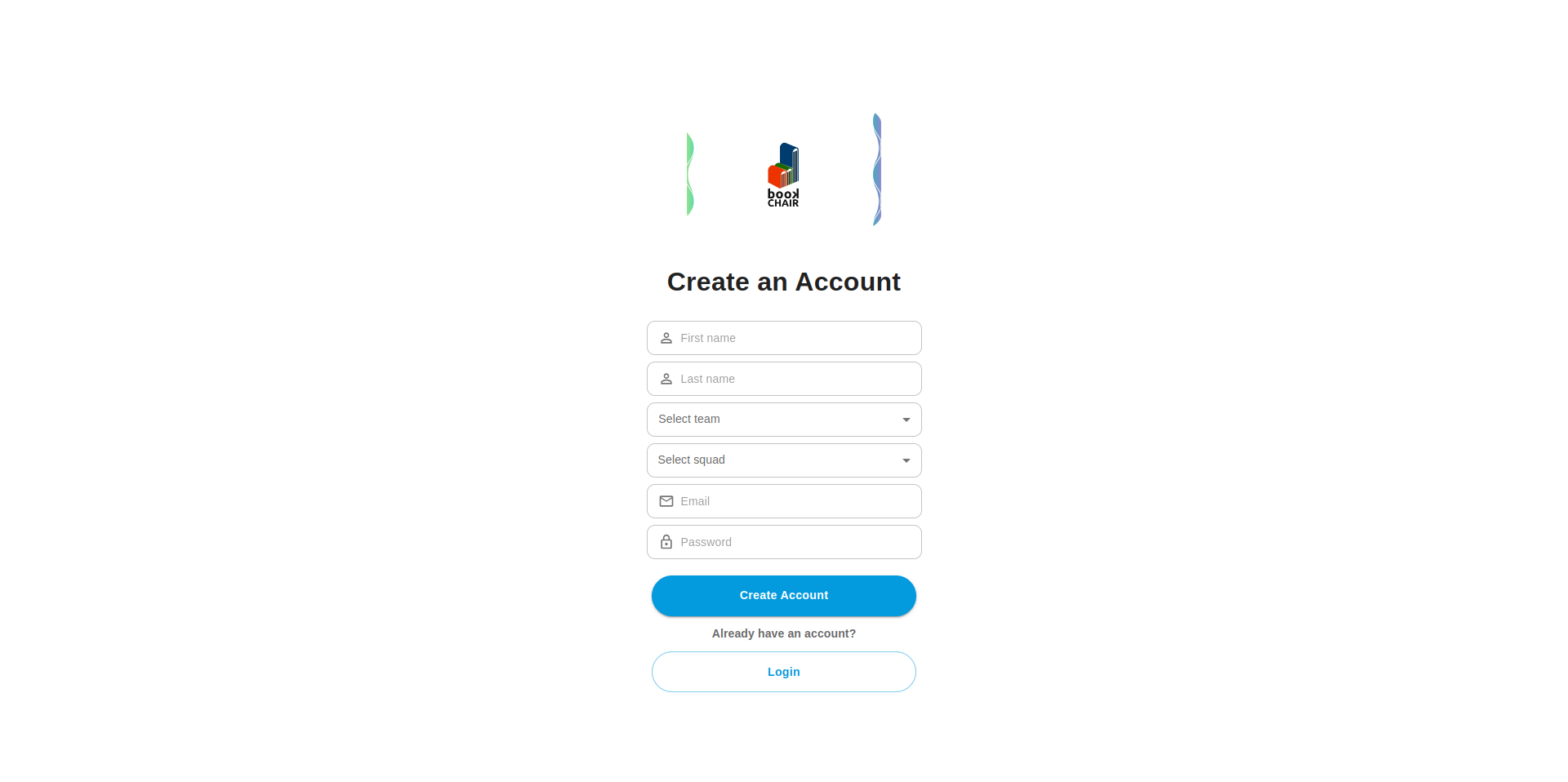
- Account creation option for new users

- Responsive navigation elements

- Intuitive form layout and design

2 Authentication System - Registration Feature

The registration system implements a comprehensive user onboarding process with team selection capabilities, form validation, and responsive design. Here's a detailed breakdown of the implementation:



2.1 Component Structure

1. \*\*RegisterPage Component\*\*

```typescript

export const RegisterPage = () => {

const theme = useTheme();

const isMobile = useMediaQuery(theme.breakpoints.down("sm"));

const isTablet = useMediaQuery(theme.breakpoints.between("sm", "md"));

return (

<Container

maxWidth={isMobile ? "xs" : isTablet ? "sm" : "lg"}

sx={{

display: "flex",

flexDirection: "column",

alignItems: "center",

justifyContent: "center",

}}

>

// Container implementation

</Container>

);

};

```

- Implements responsive container based on device size

- Integrates RegisterImages and RegisterForm components

- Handles layout adaptation for different screen sizes

2.2Form Implementation

1. \*\*Form Structure and State Management\*\*

```typescript

interface IUserRegister {

first\_name: string;

last\_name: string;

team: string;

squad: string;

email: string;

password: string;

}

const formik = useFormik({

initialValues: {

first\_name: "",

last\_name: "",

team: "",

squad: "",

email: "",

password: "",

},

onSubmit: (values) => {

dispatch(registerThunk(values));

},

validationSchema: validationSchemaRegister,

});

```

2. \*\*Form Fields and Validation\*\*

- Personal Information Fields:

- First Name

- Last Name

- Email

- Password

- Organization Fields:

- Team Selection (FND, BND, DESIGN, QA, HR, PR)

- Squad Selection (NETWATCH, GIZMO, NOVA)

3. \*\*UI Components\*\*

```typescript

<TextField

id="first\_name"

value={formik.values.first\_name}

onChange={formik.handleChange}

onBlur={formik.handleBlur}

sx={{

height: "42px",

width: "100%",

// Additional styling

}}

placeholder="First name"

variant="outlined"

InputProps={{

startAdornment: (

<InputAdornment position="start">

<PersonOutlineOutlinedIcon />

</InputAdornment>

),

}}

/>

```

2.3Team and Squad Selection Implementation

1. \*\*Team Selection Component\*\*

```typescript

<FormControl fullWidth>

<Select

id="team"

name="team"

value={formik.values.team}

onChange={formik.handleChange}

variant="outlined"

displayEmpty

>

<MenuItem value="FND">FrontEnd</MenuItem>

<MenuItem value="BND">BackEnd</MenuItem>

<MenuItem value="DESIGN">Design</MenuItem>

<MenuItem value="QA">Quality assurance</MenuItem>

<MenuItem value="HR">Human resources</MenuItem>

<MenuItem value="PR">Product team</MenuItem>

</Select>

</FormControl>

```

2. \*\*Squad Selection Component\*\*

```typescript

<FormControl fullWidth>

<Select

id="squad"

name="squad"

value={formik.values.squad}

onChange={formik.handleChange}

variant="outlined"

displayEmpty

>

<MenuItem value="NETWATCH">Net watch</MenuItem>

<MenuItem value="GIZMO">Gizmo</MenuItem>

<MenuItem value="NOVA">Nova</MenuItem>

</Select>

</FormControl>

```

2.4State Management and API Integration

1. \*\*Registration Thunk\*\*

```typescript

export const registerThunk = createAsyncThunk(

"auth/register",

async (userData: IUserRegister, { rejectWithValue }) => {

try {

const response = await axiosInstance.post("/auth/register", userData);

toast.success("Successfully registered");

setTimeout(() => {

window.location.href = "/login";

}, 1000);

return response.data;

} catch (error) {

// Error handling

}

}

);

```

2. \*\*Redux State Updates\*\*

```typescript

builder.addCase(registerThunk.fulfilled, (state, action) => {

state = action.payload;

});

```

2.5User Experience Features

1. \*\*Responsive Design\*\*

- Adaptive layout for mobile, tablet, and desktop views

- Consistent styling across device sizes

- Optimized input field sizes and spacing

2. \*\*Visual Feedback\*\*

- Form validation indicators

- Success/error notifications

- Loading states during submission

- Clear input field labels and icons

3. \*\*Navigation Flow\*\*

- Automatic redirection to login after successful registration

- Clear path back to login for existing users

- Intuitive form progression

2.6Security Implementation

1. \*\*Form Validation\*\*

- Email format validation

- Password strength requirements

- Required field validation

- Team and squad selection validation

2. \*\*Data Security\*\*

- Secure password handling

- Protected API endpoints

- Input sanitization

- Form submission security

2.7 Error Handling

1. \*\*Client-side Validation\*\*

- Real-time input validation

- Field-specific error messages

- Form submission validation

2. \*\*Server Response Handling\*\*

- API error management

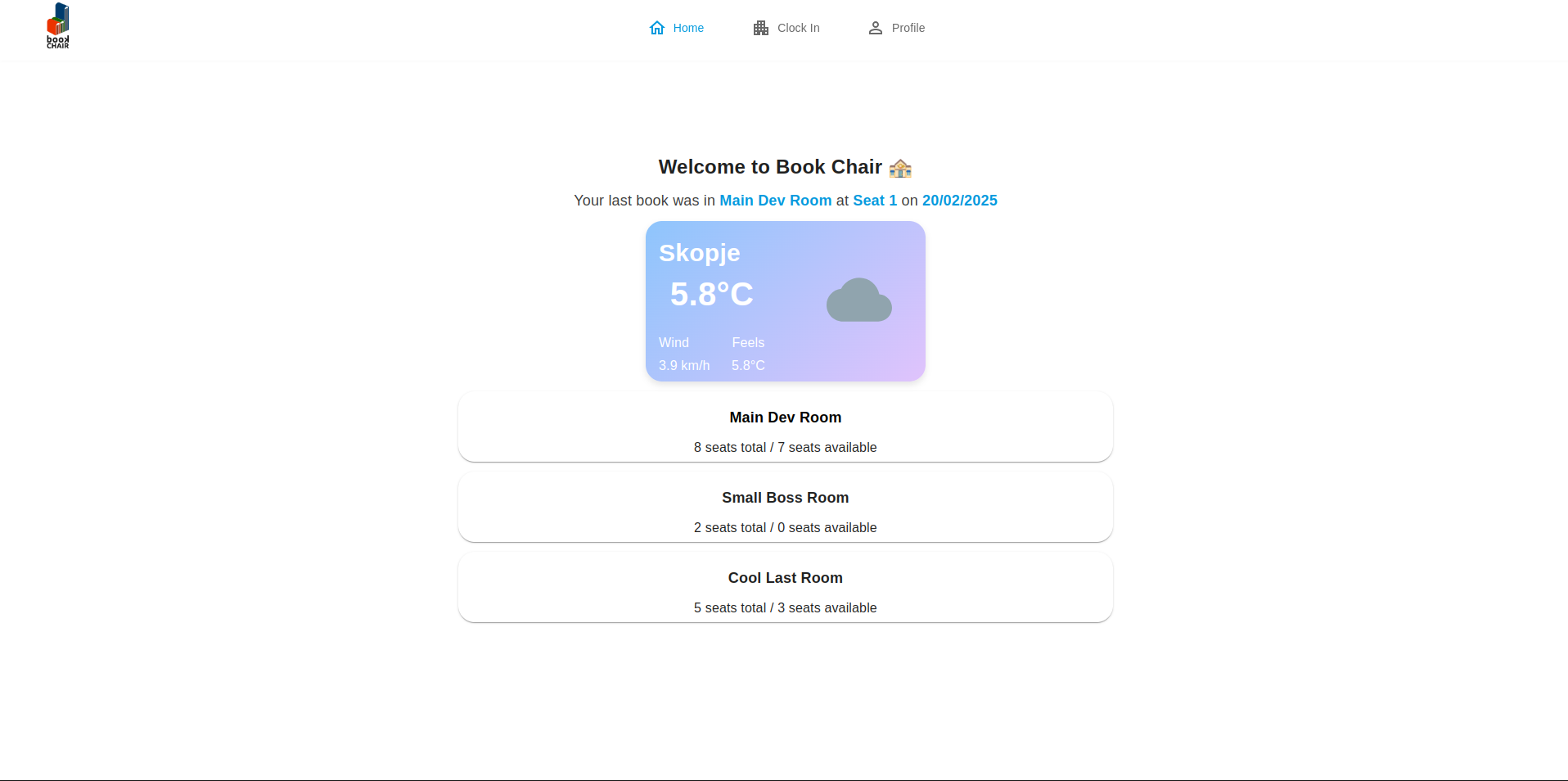
- User-friendly error messages

- Network error handling

- Duplicate email detection

3 Room Booking System Implementation

3.1 Component Structure



1. RoomsPage Component

```typescript

export const RoomsPage = () => {

const theme = useTheme();

return (

<>

<MediaQuery minHeight={781}>

<Container

sx={{

display: "flex",

flexDirection: "column",

alignItems: "center",

justifyContent: "center",

padding: theme.spacing(2),

}}

>

<Box

sx={{

display: "flex",

flexDirection: "column",

alignItems: "center",

width: "100%",

padding: theme.spacing(2),

borderRadius: 2,

backgroundColor: "white",

overflow: "hidden", // Prevent overflow

}}

>

<RoomsForm />

<MediaQuery maxWidth={1023}>

{" "}

<NavigationBar />

</MediaQuery>

<MediaQuery minWidth={1024}>

{" "}

<NavigationBarDesktopNoLogout />

</MediaQuery>

</Box>

</Container>

</MediaQuery>

<MediaQuery maxHeight={780}>

<Container

sx={{

display: "flex",

flexDirection: "column",

alignItems: "center",

justifyContent: "center",

padding: theme.spacing(2),

}}

>

<Box

sx={{

display: "flex",

flexDirection: "column",

alignItems: "center",

width: "100%",

padding: theme.spacing(2),

}}

>

<RoomsForm />

<MediaQuery maxWidth={1023}>

{" "}

<NavigationBar />

</MediaQuery>

<MediaQuery minWidth={1024}>

{" "}

<NavigationBarDesktopNoLogout />

</MediaQuery>

</Box>

</Container>

</MediaQuery>

</>

);

};

```

- Implements responsive container with height breakpoints (780px)

- Handles different layouts for desktop and mobile views

- Manages navigation bar variations based on screen width

3.2 Room Management Implementation

1. \*\*RoomsForm Component\*\*

```typescript

interface IRoom {

\_id: string;

name: string;

desks: string[];

}

interface IBooking {

\_id: string;

date: string;

desk: {

\_id: string;

name: string;

shortName: string;

};

}

```

2. \*\*State Management\*\*

- Room List Management:

- Room data fetching

- Availability tracking

- Booking status updates

- Booking Management:

- Current booking tracking

- Historical booking data

- Date normalization

3. \*\*Data Fetching Logic\*\*

```typescript

useEffect(() => {

const defaultQueryParams = {

search: "",

sortBy: "createdAt",

page: 1,

order: "asc",

};

dispatch(getRoomThunk(room.queryParameters || defaultQueryParams));

}, [dispatch, room.queryParameters]);

```

##### Type Definitions and Interfaces

1. **Room State Interface**

typescript

Copy

export interface roomState {

currentRoom: IRoom;

roomList: IRoom[];

queryParameters: IRoomGetRoomQuery;

paginationMetadata: IRoomPaginationMetadata;

}

interface IRoom {

\_id: string;

name: string;

description: string;

images: IImage[];

desks: IDesk[];

type: roomType;

status: roomStatus;

}

interface IRoomPaginationMetadata {

totalRooms: number;

totalPages: number;

currentPage: number;

}

1. **Booking State Interface**

typescript

Copy

export interface bookingState {

currentBooking: IBooking;

bookingList: IBooking[];

queryParameters: IBookingGetBookingQuery;

paginationMetadata: IBookingPaginationMetadata;

}

interface IBooking {

\_id: string;

status: bookingStatus;

date: string;

user: {

\_id: string;

first\_name: string;

last\_name: string;

team: string;

squad: string;

email: string;

password: string;

role: userRole;

avatar\_url: string;

};

desk: {

\_id: string;

name: string;

shortName: string;

description: string;

status: deskStatus;

direction: deskDirection;

};

duration: number;

notes: string;

cancellationReason: string;

referenceNumber: string;

}

##### State Management Implementation

1. **Room Initial State**

typescript

Copy

export const initialState: roomState = {

currentRoom: {

\_id: "",

name: "",

description: "",

images: [] as IImage[],

desks: [] as IDesk[],

type: "" as roomType,

status: "" as roomStatus,

},

roomList: [],

queryParameters: {},

paginationMetadata: {

totalRooms: 0,

totalPages: 0,

currentPage: 1,

},

};

1. **Booking Initial State**

typescript

Copy

export const initialState: bookingState = {

currentBooking: {

\_id: "",

status: "" as bookingStatus,

date: "",

user: {

\_id: "",

first\_name: "",

last\_name: "",

team: "",

squad: "",

email: "",

password: "",

role: "" as userRole,

avatar\_url: "",

},

desk: {

\_id: "",

name: "",

shortName: "",

description: "",

status: "" as deskStatus,

direction: "" as deskDirection,

},

duration: 1,

notes: "",

cancellationReason: "",

referenceNumber: "",

},

bookingList: [],

queryParameters: {},

paginationMetadata: {

totalBookings: 0,

totalPages: 0,

currentPage: 1,

},

};

##### Redux Thunk Implementation

1. **Room Thunks**

typescript

Copy

export const getRoomThunk = createAsyncThunk(

"room/getAll",

async (queryParams: IRoomGetRoomQuery, { rejectWithValue }) => {

try {

const response = await axiosInstance.get("/room/", {

params: queryParams,

});

return response.data;

} catch (error) {

if (axios.isAxiosError(error) && error.response) {

return rejectWithValue(error.response.data);

} else {

return rejectWithValue("An unexpedted error occurred");

}

}

}

);

1. **Booking Thunks**

typescript

Copy

export const getLastBookingThunk = createAsyncThunk(

"booking/last",

async (\_, { rejectWithValue }) => {

try {

const response = await axiosInstance.get("/booking/last-booking");

return response.data;

} catch (error) {

if (axios.isAxiosError(error) && error.response) {

return rejectWithValue(error.response.data);

} else {

return rejectWithValue("An unexpected error occurred");

}

}

}

);

export const getBookingThunk = createAsyncThunk(

"booking/getAll",

async (queryParams: IBookingGetBookingQuery, { rejectWithValue }) => {

try {

const response = await axiosInstance.get("/booking/", {

params: queryParams,

});

return response.data;

} catch (error) {

if (axios.isAxiosError(error) && error.response) {

return rejectWithValue(error.response.data);

} else {

return rejectWithValue("An unexpedted error occurred");

}

}

}

);

##### Redux State Updates

1. **Room State Updates**

typescript

Copy

builder.addCase(getRoomThunk.fulfilled, (state, action) => {

state.roomList = action.payload.rooms;

state.paginationMetadata = {

totalRooms: action.payload.totalRooms,

totalPages: action.payload.totalPages,

currentPage: action.payload.currentPage,

};

});

1. **Booking State Updates**

typescript

Copy

builder.addCase(getLastBookingThunk.fulfilled, (state, action) => {

state.currentBooking = action.payload;

});

builder.addCase(getBookingThunk.fulfilled, (state, action) => {

state.bookingList = action.payload.bookings;

state.paginationMetadata = {

totalBookings: action.payload.totalBookings,

totalPages: action.payload.totalPages,

currentPage: action.payload.currentPage,

};

});

##### Error Handling

1. **Thunk Error Management**

* Axios error handling
* Type-safe error responses
* User-friendly error messages
* Error state management

1. **State Error Handling**

* Error boundary implementation
* Loading states
* Error recovery mechanisms
* Error feedback to users

##### Data Flow Architecture

1. **State Updates**

* Immutable state updates
* Type-safe actions
* Payload transformations
* State normalization

1. **Data Synchronization**

* Booking-room relationship management
* State consistency
* Cache invalidation
* Optimistic updates

Navigation Implementation

1. \*\*NavigationBarDesktopNoLogout Component\*\*

```typescript

export const NavigationBarDesktopNoLogout = () => {

const location = useLocation();

const isActive = (path: string) => location.pathname === path;

const { id } = useParams();

return (

<Box

sx={{

width: "100%",

display: "flex",

flexDirection: "row",

alignItems: "center",

justifyContent: "center",

height: "80px",

position: "fixed",

top: 0,

backgroundColor: "white",

boxShadow: "0 -2px 5px rgba(0, 0, 0, 0.1)",

}}

>

<Box

component="img"

src={logo}

sx={{

width: "100%",

maxWidth: "81px",

position: "absolute",

display: "flex",

left: 30,

}}

/>

<Link to="/room" style={{ textDecoration: "none", flex: 0.07 }}>

<Box

sx={{

flex: 1,

height: "47px",

display: "flex",

justifyContent: "center",

alignItems: "center",

flexDirection: "row",

}}

>

<HomeOutlinedIcon

sx={{

width: "24px",

height: "24px",

color:

isActive("/room") ||

isActive("/room/main-reserve-desk") ||

isActive("/room/small-reserve-desk") ||

isActive("/room/cool-reserve-desk")

? "#039ADE"

: "#686868",

}}

/>

<Typography

sx={{ pl: 1, pt: 0.3 }}

fontStyle="Roboto"

fontWeight="400"

textAlign="center"

color={

isActive("/room") ||

isActive("/room/main-reserve-desk") ||

isActive("/room/small-reserve-desk") ||

isActive("/room/cool-reserve-desk")

? "#039ADE"

: "#686868"

}

fontSize="14px"

lineHeight="19.6px"

>

Home

</Typography>

</Box>

</Link>

<Link to="/clock-in" style={{ textDecoration: "none", flex: 0.07 }}>

<Box

sx={{

flex: 1,

height: "47px",

display: "flex",

justifyContent: "center",

alignItems: "center",

flexDirection: "row",

}}

>

<ApartmentOutlinedIcon

sx={{

width: "24px",

height: "24px",

color:

isActive("/clock-in") || isActive("/clock-out")

? "#039ADE"

: "#686868",

}}

/>

<Typography

sx={{ pl: 1, pt: 0.3 }}

fontStyle="Roboto"

fontWeight="400"

textAlign="center"

color={

isActive("/clock-in") || isActive("/clock-out")

? "#039ADE"

: "#686868"

}

fontSize="14px"

lineHeight="19.6px"

>

Clock In

</Typography>

</Box>

</Link>

<Link to="/me" style={{ textDecoration: "none", flex: 0.07 }}>

<Box

sx={{

flex: 1,

height: "47px",

display: "flex",

justifyContent: "center",

alignItems: "center",

flexDirection: "row",

}}

>

<PersonOutlineOutlinedIcon

sx={{

width: "24px",

height: "24px",

color:

isActive("/me") || isActive(`/editProfile/${id}`)

? "#039ADE"

: "#686868",

}}

/>

<Typography

sx={{ pl: 1, pt: 0.3 }}

fontStyle="Roboto"

fontWeight="400"

textAlign="center"

color={

isActive("/me") || isActive(`/editProfile/${id}`)

? "#039ADE"

: "#686868"

}

fontSize="14px"

lineHeight="19.6px"

>

Profile

</Typography>

</Box>

</Link>

</Box>

);

}

```

2. \*\*Navigation Features\*\*

- Route Management:

- Home (/room)

- Clock In (/clock-in)

- Profile (/me)

- Active State Tracking:

- Dynamic color changes

- Route-based highlighting

- Nested route handling

##### UI Components and Styling

1. \*\*Room Cards\*\*

```typescript

<Card sx={{

width: "100%",

height: "86px",

borderRadius: "20px"

}}>

<CardContent>

<Typography>

// Room information display

</Typography>

</CardContent>

</Card>

```

2. \*\*Style Implementation\*\*

- Theme Integration:

- Responsive breakpoints

- Spacing system

- Typography hierarchy

- Layout Components:

- Container configurations

- Box layouts

- Card structures

Responsive Design Features

1. \*\*Media Queries\*\*

- Height-based breakpoints:

- Maximum height: 780px

- Minimum height: 781px

- Width-based breakpoints:

- Desktop: ≥1024px

- Mobile: <1024px

2. \*\*Layout Adaptations\*\*

- Container sizing

- Component spacing

- Navigation positioning

- Typography scaling

Room Availability Management

1. Availability Calculations

```typescript

const [mainCounter, setMainCounter] = useState(8);

useEffect(() => {

setMainCounter(MainRoomAvailability(booking.bookingList, room.roomList, d));

}, [booking.bookingList, room.roomList, d]);

```

2. Date Handling

- UTC normalization

- Date formatting

- Booking date management

State and Data Flow

1. Redux Integration

- Room state management

- Booking state management

- Query parameter handling

- Thunk implementations

2. Data Processing

- Room availability calculation

- Booking history tracking

- Current booking identification

- Room name resolution

Navigation Features

1. Route Configuration

- Main room routes

- Booking paths

- Profile management

- Clock in/out system

2. Active State Management

- Route matching

- Visual feedback

- State persistence

User Experience Features

1. Visual Feedback

- Active route indicators

- Available seats display

- Last booking information

- Weather widget integration

2. Accessibility Features

- Semantic markup

- Screen reader support

- Keyboard navigation

- Focus management

Room Layout Components

1. MainRoomPage Component

```typescript

export const MainRoomPage = () => {

const theme = useTheme();

const isMobile = useMediaQuery(theme.breakpoints.down("sm"));

const isTablet = useMediaQuery(theme.breakpoints.between("sm", "md"));

const dispatch = useAppDispatch();

const room = useAppSelector((state) => state.room);

const booking = useAppSelector((state) => state.booking);

let day = dayjs().startOf("day").add(1, "day").toISOString();

day = normalizeDateToUTC(day);

day = day.split("T")[0] + "T00:00:00.000Z";

useEffect(() => {

const defaultQueryParams = {

search: "",

sortBy: "createdAt",

page: 1,

order: "asc",

};

dispatch(getRoomThunk(room.queryParameters || defaultQueryParams));

}, [dispatch, room.queryParameters]);

useEffect(() => {

const defaultQueryParams = {

search: "",

sortBy: "createdAt",

page: 1,

order: "asc",

};

dispatch(getBookingThunk(booking.queryParameters || defaultQueryParams));

}, [dispatch, booking.queryParameters]);

const [mainCounter, setMainCounter] = useState(8);

useEffect(() => {

setMainCounter(

MainRoomAvailability(booking.bookingList, room.roomList, day)

);

}, [booking.bookingList, room.roomList, day]);

return (

<Container

maxWidth={isMobile ? "xs" : isTablet ? "sm" : "md"} // Adjust maxWidth based on screen size

sx={{

display: "flex",

flexDirection: "column",

alignItems: "center",

justifyContent: "center",

padding: theme.spacing(2),

width: isMobile ? "100%" : "100%",

height: isMobile ? "90vh" : isTablet ? "100vh" : "80vh", // Adjust height based on screen size

maxHeight: "100%",

TOP: 20,

}}

>

<Box

sx={{

display: "flex",

flexDirection: "column",

alignItems: "center",

width: "100%",

height: "100%",

borderRadius: 2,

backgroundColor: "white",

}}

>

<MediaQuery maxWidth={1023}>

<ButtonBack />

</MediaQuery>

{room.roomList.length > 0 && (

<>

<Box

sx={{

width: "100%",

maxWidth: 400,

display: isMobile ? "none" : "flex",

flexDirection: "row",

position: "absolute",

left: "100px",

top: "130px",

gap: "5px",

}}

>

<Link to="/room" style={{ textDecoration: "none" }}>

<Typography color="#686868">Home / </Typography>

</Link>

<Typography color=" #039ADE">

{" "}

{room.roomList[0]?.name} 🏣

</Typography>

</Box>

<Typography

sx={{

width: "100%",

maxWidth: 361,

display: "flex",

flexDirection: "column",

position: "absolute",

left: isMobile ? 16 : "",

top: isMobile ? 112 : 185,

}}

textAlign={isMobile || isTablet ? "left" : "center"}

fontStyle="Roboto"

fontSize="24px"

fontWeight="700"

lineHeight="33.6px"

>

{room.roomList[0]?.name} 🏣

<Typography

sx={{

width: "100%",

maxWidth: 361,

display: "flex",

flexDirection: "column",

position: "absolute",

left: isMobile ? 0 : "",

top: isMobile ? 25 : 30,

}}

textAlign={isMobile || isTablet ? "left" : "center"}

fontStyle="Roboto"

fontSize="18px"

fontWeight="400"

lineHeight="25.5px"

color="#686868"

>

{" "}

{room.roomList[0]?.desks.length} seats / {mainCounter} seats

available

</Typography>

</Typography>

<MainRoomForm />

</>

)}

<MediaQuery maxWidth={1023}>

{" "}

<NavigationBar />

</MediaQuery>

<MediaQuery minWidth={1024}>

{" "}

<NavigationBarDesktopNoLogout />

</MediaQuery>

</Box>

</Container>

);

}

```

- Implements responsive container based on device size

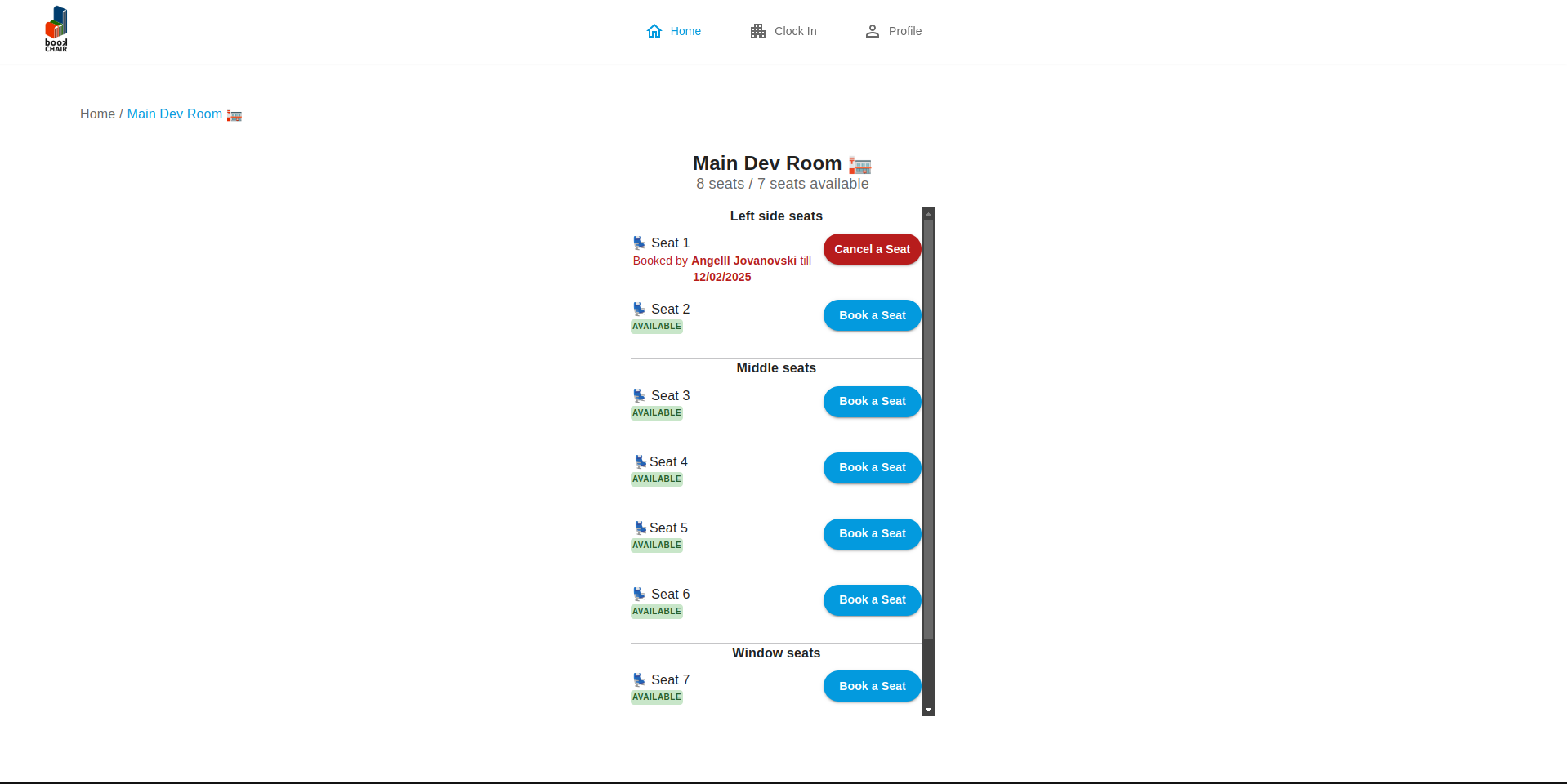
- Manages room and booking data fetching

- Tracks seat availability counters

- Handles breadcrumb navigation

- Integrates with navigation components

2. MainRoomForm Component



```typescript

export const MainRoomForm = () => {

const room = useAppSelector((state) => state.room);

const dispatch = useAppDispatch();

const theme = useTheme();

const isMobile = useMediaQuery(theme.breakpoints.down("sm"));

const isTablet = useMediaQuery(theme.breakpoints.between("sm", "md"));

useEffect(() => {

const defaultQueryParams = {

search: "",

sortBy: "createdAt",

page: 1,

order: "asc",

};

dispatch(getRoomThunk(room.queryParameters || defaultQueryParams));

}, [dispatch, room.queryParameters]);

return (

<Box

sx={{

width: isMobile ? "100%" : "100%",

height: isMobile ? "70vh" : isTablet ? "70vh" : "65vh", // Adjust height based on screen size

maxHeight: "100%",

overflowY: "auto",

overflowX: "hidden",

maxWidth: isMobile || isTablet ? "361px" : "372px",

display: "flex",

flexDirection: "column",

position: "absolute",

alignItems: "center",

top: isMobile ? "200px" : "256px",

}}

>

<Box

sx={{

display: "flex",

flexDirection: "column",

gap: "10px",

width: "100%",

maxWidth: "361px",

left: 20,

}}

>

<Typography

sx={{

width: "100%",

maxWidth: "361px",

}}

fontStyle="Roboto"

fontSize="16px"

fontWeight="700"

lineHeight="22.4px"

>

Left side seats

</Typography>

<Seat1 seat={room.roomList[0].desks[0]} roomId={room.roomList[0].\_id} />

<Seat2 seat={room.roomList[0].desks[1]} roomId={room.roomList[0].\_id} />

</Box>

<Box

sx={{ width: "100%", maxWidth: 361, border: 1, color: "#C5C5C7 " }}

></Box>

<Box

sx={{

display: "flex",

flexDirection: "column",

gap: "10px",

maxWidth: "361px",

width: "100%",

top: 204,

}}

>

<Typography

sx={{

maxWidth: "361px",

width: "100%",

}}

fontStyle="Roboto"

fontSize="16px"

fontWeight="700"

lineHeight="22.4px"

>

Middle seats

</Typography>

<Seat3 seat={room.roomList[0].desks[2]} roomId={room.roomList[0].\_id} />

<Seat4 seat={room.roomList[0].desks[3]} roomId={room.roomList[0].\_id} />

<Seat5 seat={room.roomList[0].desks[4]} roomId={room.roomList[0].\_id} />

<Seat6 seat={room.roomList[0].desks[5]} roomId={room.roomList[0].\_id} />

</Box>

<Box

sx={{ width: "100%", maxWidth: 361, border: 1, color: "#C5C5C7 " }}

></Box>

<MediaQuery maxHeight={700}>

<Box

sx={{

display: "flex",

flexDirection: "column",

gap: "10px",

maxWidth: "361px",

minHeight: "40vh",

width: "100%",

}}

>

<Typography

sx={{

maxWidth: "361px",

width: "100%",

}}

fontStyle="Roboto"

fontSize="16px"

fontWeight="700"

lineHeight="22.4px"

>

Window seats

</Typography>

<Seat7

seat={room.roomList[0].desks[6]}

roomId={room.roomList[0].\_id}

/>

<Seat8

seat={room.roomList[0].desks[7]}

roomId={room.roomList[0].\_id}

/>

</Box>

</MediaQuery>

<MediaQuery minHeight={701}>

<Box

sx={{

display: "flex",

flexDirection: "column",

gap: "10px",

maxWidth: "361px",

width: "100%",

}}

>

<Typography

sx={{

maxWidth: "361px",

width: "100%",

}}

fontStyle="Roboto"

fontSize="16px"

fontWeight="700"

lineHeight="22.4px"

>

Window seats

</Typography>

<Seat7

seat={room.roomList[0].desks[6]}

roomId={room.roomList[0].\_id}

/>

<Seat8

seat={room.roomList[0].desks[7]}

roomId={room.roomList[0].\_id}

/>

</Box>

</MediaQuery>

</Box>

);

}

```

- Organizes seats into sections:

- Left side seats (2 seats)

- Middle seats (4 seats)

- Window seats (2 seats)

- Implements responsive height adjustments

- Manages seat component rendering

Seat Management

1. Seat Component Structure

```typescript

interface SeatProps {

seat: string | IDesk;

roomId: string;

}

export const Seat3 = ({ seat, roomId }: SeatProps) => {

const dispatch = useAppDispatch();

const desk = useAppSelector((state) => state.desk);

useEffect(() => {

const defaultQueryParams = {

search: "",

sortBy: "createdAt",

page: 1,

order: "asc",

};

dispatch(getDeskThunk(desk.queryParameters || defaultQueryParams));

}, [dispatch, desk.queryParameters]);

const currentDesk = desk.deskList.find((value) => value.\_id === seat);

const convertedDesk = String(currentDesk?.\_id);

return (

<Box

sx={{

display: "flex",

flexDirection: "row",

width: 361,

height: 71,

}}

>

<Box

sx={{

width: 73,

height: 47,

top: 44,

left: 20,

borderRadius: "4px 0px 0px 0px",

}}

>

<Typography>

{desk.deskList.map((value) =>

value.\_id === seat

? value.name.split(" ").pop() + " " + value.shortName

: ""

)}

</Typography>

<Available deskId={convertedDesk} roomId={roomId} />

</Box>

<Box

sx={{

width: 113,

top: 49,

padding: " 9px, 16px, 9px, 16px",

display: "flex",

flexDirection: "column",

}}

>

<ButtonBookSeat deskId={convertedDesk} roomId={roomId} />

</Box>

</Box>

);

}

```

2. Seat Features

- Individual seat status tracking

- Booking button integration

- Availability indicator

- Seat identification display

- Room association

Layout Organization

1. Section Management

- Left side seats section:

- Two seats with unique positioning

- Individual availability tracking

- Middle seats section:

- Four seats with central positioning

- Flexible layout adaptation

- Window seats section:

- Two seats with window-adjacent positioning

- Height-based rendering conditions

2. Responsive Adaptations

```typescript

<MediaQuery maxHeight={700}>

// Compact layout for smaller screens

</MediaQuery>

<MediaQuery minHeight={701}>

// Extended layout for larger screens

</MediaQuery>

```

Visual Components

1. Seat Visualization

- Seat icon display

- Seat number and shortname

- Availability status indicator

- Booking action button

- User booking information

2. Layout Elements

```typescript

<Box

sx={{

display: "flex",

flexDirection: "column",

gap: "10px",

width: "100%",

maxWidth: "361px",

}}

>

// Section content

</Box>

```

Seat Status Management

1. Availability Tracking

```typescript

const Available = ({ deskId, roomId }: SeatProps) => {

const dispatch = useAppDispatch();

const booking = useAppSelector((state) => state.booking);

useEffect(() => {

const defaultQueryParams = {

search: "",

sortBy: "createdAt",

page: 1,

order: "asc",

};

dispatch(getBookingThunk(booking.queryParameters || defaultQueryParams));

}, [dispatch, booking.queryParameters]);

const newList = booking.bookingList.map((value) => ({

...value,

date: normalizeDateToUTC(value.date),

}));

let day = dayjs().startOf("day").add(1, "day").toISOString();

day = normalizeDateToUTC(day);

const bookingId = newList.find(

(value) =>

value.date === day &&

value.status === "booked" &&

value.desk.\_id === deskId

);

let object;

let flag = 0;

let formattedDate;

if (newList.length > 0) {

for (let i = 0; i < newList.length; i++) {

if (

newList[i].user.\_id === bookingId?.user.\_id &&

newList[i].status === "booked" &&

newList[i].date === bookingId.date &&

newList[i].desk.\_id === deskId

) {

if (newList[i].duration > 1) {

object = newList[i + newList[i].duration - 1];

flag = 1;

const splitDate = object.date.split("-");

formattedDate =

splitDate[2] + "/" + splitDate[1] + "/" + splitDate[0];

break;

} else {

object = newList[i];

flag = 1;

const splitDate = object.date.split("-");

formattedDate =

splitDate[2] + "/" + splitDate[1] + "/" + splitDate[0];

break;

}

}

if (flag === 1) {

break;

}

}

}

return (

<Box

sx={{

display: "flex",

width: 224,

height: 18,

gap: "10px",

padding: "4px, 6px, 4px, 6px",

}}

>

{bookingId?.\_id === undefined ? (

<Typography

sx={{

width: "64px",

paddingTop: "2px",

background: "#C8E6C9",

borderRadius: "4px",

letterSpacing: "0.5px",

}}

fontStyle="Roboto"

fontSize="10px"

fontWeight="600"

lineHeight="14px"

color="#2A602C "

textAlign="center"

>

AVAILABLE

</Typography>

) : (

<Typography

color="#B71C1C"

fontStyle="Roboto"

fontSize="14px"

fontWeight="500"

lineHeight="19.6px"

>

Booked by{" "}

<b>

{bookingId?.user.first\_name} {bookingId?.user.last\_name}

</b>{" "}

till <b>{formattedDate}</b>

</Typography>

)}

</Box>

);

}

```

2. Booking Integration

```typescript

const ButtonBookSeat = ({ deskId, roomId }: SeatProps) => {

// Booking action handling

// Status updates

// User interaction management

}

```

Responsive Design Implementation

1. Screen Size Adaptations

```typescript

const isMobile = useMediaQuery(theme.breakpoints.down("sm"));

const isTablet = useMediaQuery(theme.breakpoints.between("sm", "md"));

```

- Mobile layout adjustments

- Tablet-specific styling

- Desktop optimizations

2. \*\*Layout Adjustments\*\*

- Dynamic container sizing

- Flexible height management

- Scroll behavior control

- Component positioning

Navigation Integration

1. Breadcrumb Navigation

```typescript

<Link to="/room" style={{ textDecoration: "none" }}>

<Typography color="#686868">Home / </Typography>

</Link>

```

2. Responsive Navigation

- Mobile navigation bar

- Desktop navigation integration

- Route management

- Active state tracking

Room Information Display

1. Room Details

```typescript

<Typography

textAlign={isMobile || isTablet ? "left" : "center"}

fontStyle="Roboto"

fontSize="24px"

fontWeight="700"

>

{room.roomList[0]?.name} 🏣

</Typography>

```

2. Availability Summary

- Total seat count

- Available seat counter

- Section-wise availability

- Real-time updates

**Breakdown of My-Profile-Page component**

MyProfilePage is a React functional component that displays a user's profile information and booking details. It integrates with Redux for state management, handles logout functionality, and adapts to different screen sizes using Material-UI (@mui/material).

**Code**

export const MyProfilePage = () => {

const [open, setOpen] = useState(false);

const dispatch = useAppDispatch();

const user = useAppSelector((state) => state.auth);

const booking = useAppSelector((state) => state.booking);

const theme = useTheme();

const isMobile = useMediaQuery(theme.breakpoints.down("sm"));

const isTablet = useMediaQuery(theme.breakpoints.between("sm", "md"));

const isDesktop = useMediaQuery(theme.breakpoints.up("lg"));

const isMobileHeight = useMediaQuery("(max-height: 750px)");

useEffect(() => {

dispatch(meThunk());

dispatch(getBookingThunk({ order: "asc" }));

}, [dispatch]);

const handleClickOpen = () => {

setOpen(true);

};

const handleClickClose = () => {

setOpen(false);

};

const logoutLogic = () => {

dispatch(logoutThunk());

};

const filteredBookingsOfTheUser: IBooking[] = booking.bookingList.filter(

(value) => value.user.\_id === user.\_id

);

const filteredBookingsOfTheUserNotCancelled: IBooking[] =

booking.bookingList.filter(

(value) => value.user.\_id === user.\_id && value.status === "booked"

);

const filteredBookingsOfTheUserCancelled: IBooking[] =

booking.bookingList.filter(

(value) => value.user.\_id === user.\_id && value.status === "cancelled"

);

return (

<Container

sx={{

display: "flex",

flexDirection: "column",

alignItems: "center",

justifyContent: "center",

width: "100%",

}}

>

{/\* Logout Icon \*/}

<MediaQuery maxWidth={1024}>

<LogoutIcon

sx={{

width: 32,

height: 32,

color: "#B71C1C",

position: "absolute",

display: "flex",

flexDirection: "column",

top: 70,

right: 16,

}}

onClick={handleClickOpen}

/>

<Dialog

open={open}

onClose={handleClickClose}

fullWidth

PaperProps={{

sx: {

borderRadius: "20px",

width: "369px",

height: "164px",

},

}}

>

<CloseIcon

sx={{

width: 20,

height: 20,

display: "flex",

position: "absolute",

flexDirection: "column",

top: 10,

right: 10,

}}

onClick={handleClickClose}

/>

<DialogTitle id="logout-dialog-title">

<Typography fontSize="18px" fontStyle="Roboto" fontWeight="700">

Log out

</Typography>

</DialogTitle>

<DialogContent>

<DialogContentText id="logout-dialog-description">

<Typography fontWeight="400" fontSize="14px" fontStyle="Roboto">

Are you sure you want to logout ?

</Typography>

</DialogContentText>

</DialogContent>

<DialogActions>

<Button

size="medium"

sx={{

width: "78px",

height: "38px",

borderRadius: "100px",

backgroundColor: "#DBDBDD",

textTransform: "none",

color: "gray",

top: "-40%",

fontWeight: 700,

fontSize: "14px",

fontStyle: "Roboto",

right: "1%",

}}

onClick={handleClickClose}

>

Cancel

</Button>

<Button

variant="contained"

size="medium"

sx={{

width: "78px",

height: "38px",

padding: "9px 16px 9px 16px",

borderRadius: "100px",

textTransform: "none",

top: "-40%",

fontWeight: 700,

fontSize: "14px",

fontStyle: "Roboto",

right: "1%",

}}

onClick={logoutLogic}

autoFocus

>

Okay

</Button>

</DialogActions>

</Dialog>

</MediaQuery>

{/\* Profile Section \*/}

<Box

sx={{

display: "flex",

flexDirection: "column",

alignItems: "center",

width: "100%",

padding: theme.spacing(2),

borderRadius: 2,

}}

>

<Box

sx={{

display: "flex",

alignItems: "center",

width: "100%",

}}

>

<Typography

sx={{ ml: isDesktop ? 10 : 1 }}

variant="h6"

fontWeight="700"

fontStyle="Roboto"

fontSize="24px"

>

{user.first\_name}’s Profile

</Typography>

<Link to={`/editProfile/${user.\_id}`}>

<SettingsOutlinedIcon

sx={{

ml: isDesktop ? 2 : 1,

mt: 0.5,

width: 32,

height: 32,

color: "#039ADE",

}}

/>

</Link>

</Box>

<Box

sx={{

display: "flex",

alignItems: "center",

maxWidth: "369px",

width: "100%",

borderRadius: 2,

padding: 2,

boxShadow: "0px 3px 6px rgba(0, 0, 0, 0.1)",

}}

>

<Box

component="img"

src={user.avatar\_url !== "" ? user.avatar\_url : icon}

sx={{

width: 154,

height: 154,

mr: 2,

borderRadius: 2,

}}

/>

<Box sx={{ maxWidth: "181px", width: "50%" }}>

<Typography

sx={{

width: "100%",

maxWidth: "181px",

}}

variant="h4"

fontWeight="700"

fontSize={

isDesktop

? "28px"

: isTablet

? "28px"

: isMobileHeight

? "18px"

: "24px"

}

>

{user.first\_name}

</Typography>

<Typography

variant="h4"

fontWeight="700"

fontSize={

isDesktop

? "28px"

: isTablet

? "28px"

: isMobileHeight

? "18px"

: "24px"

}

paddingBottom="10px"

>

{user.last\_name}

</Typography>

<Typography variant="body2" color="#686868" fontSize="12px">

Email

</Typography>

<Typography

variant="body2"

color="black"

fontSize={

isDesktop

? "12px"

: isTablet

? "12px"

: isMobileHeight

? "8px"

: "12px"

}

paddingBottom="10px"

>

{user.email}

</Typography>

<Typography variant="body2" color="#686868" fontSize="12px">

Squad

</Typography>

<Typography

variant="body2"

color="black"

fontSize={

isDesktop

? "12px"

: isTablet

? "12px"

: isMobileHeight

? "8px"

: "12px"

}

paddingBottom="10px"

>

{user.squad === "NOVA"

? "Nova"

: user.squad === "NETWATCH"

? "Net Watch"

: "Gizmo"}

</Typography>

<Typography variant="body2" color="#686868" fontSize="12px">

Team

</Typography>

<Typography

variant="body2"

color="black"

fontSize={

isDesktop

? "12px"

: isTablet

? "12px"

: isMobileHeight

? "8px"

: "12px"

}

>

{user.team === "BND"

? "Backend Team"

: user.team === "FND"

? "Frontend Team"

: user.team === "DESIGN"

? "Design Team"

: user.team === "QA"

? "Quality assurance Team"

: user.team === "HR"

? "Human resources Team"

: "Product Team"}

</Typography>

</Box>

</Box>

</Box>

{/\* Info Section \*/}

<Box

sx={{

maxWidth: "361px", // Adjust maxWidth based on screen size

width: "100%",

height: "auto", // Allow height to adjust based on content

left: "16px",

gap: "12px",

display: "flex",

flexDirection: "column",

alignItems: "center",

top: "425px",

}}

>

<Card

sx={{

maxWidth: isMobile || isTablet ? "361px" : "800px", // Adjust maxWidth based on screen size

width: "100%",

height: "86px",

borderRadius: "20px", // Added borderRadius here

}}

>

<CardContent>

<Typography

sx={{

width: "100%",

height: "25px",

paddingTop: "3px",

}}

fontStyle="Roboto"

fontWeight="700"

fontSize="18px"

component="div"

color="#000000"

>

Total bookings

</Typography>

<Typography

sx={{

width: "100%",

height: "25px",

paddingTop: "16px",

}}

component="div"

>

{filteredBookingsOfTheUser.length}

</Typography>

</CardContent>

</Card>

<Card

sx={{

maxWidth: isMobile || isTablet ? "361px" : "800px", // Adjust maxWidth based on screen size

width: "100%",

height: "86px",

borderRadius: "20px", // Added borderRadius here

}}

>

<CardContent>

<Typography

sx={{

width: "100%",

height: "25px",

paddingTop: "3px",

}}

fontStyle="Roboto"

fontWeight="700"

fontSize="18px"

component="div"

color="#000000"

>

Not cancelled bookings

</Typography>

<Typography

sx={{

width: "100%",

height: "25px",

paddingTop: "16px",

}}

component="div"

>

{filteredBookingsOfTheUserNotCancelled.length}

</Typography>

</CardContent>

</Card>

<Card

sx={{

maxWidth: isMobile || isTablet ? "361px" : "800px", // Adjust maxWidth based on screen size

width: "100%",

height: "86px",

borderRadius: "20px", // Added borderRadius here

}}

>

<CardContent>

<Typography

sx={{

width: "100%",

height: "25px",

paddingTop: "3px",

}}

fontStyle="Roboto"

fontWeight="700"

fontSize="18px"

component="div"

color="#000000"

>

Cancelled bookings

</Typography>

<Typography

sx={{

width: "100%",

height: "25px",

paddingTop: "16px",

}}

component="div"

>

{filteredBookingsOfTheUserCancelled.length}

</Typography>

</CardContent>

</Card>

</Box>

<MediaQuery maxWidth={1024}>

<NavigationBar />

</MediaQuery>

<MediaQuery minWidth={1025}>

<NavigationBarDesktop

handleClickOpen={handleClickOpen}

handleClickClose={handleClickClose}

logoutLogic={logoutLogic}

open={open}

/>

</MediaQuery>

</Container>

);

};

**State and Redux Setup**

const [open, setOpen] = useState(false);

const dispatch = useAppDispatch();

const user = useAppSelector((state) => state.auth);

const booking = useAppSelector((state) => state.booking);

open: A boolean state used to control the logout confirmation dialog.

dispatch: A Redux dispatcher for executing actions.

user: Retrieves authenticated user data from Redux.

booking: Retrieves booking data from Redux.

### ****Responsive Design with Material-UI Hooks****

const theme = useTheme(); const isMobile = useMediaQuery(theme.breakpoints.down("sm")); const isTablet = useMediaQuery(theme.breakpoints.between("sm", "md")); const isDesktop = useMediaQuery(theme.breakpoints.up("lg")); const isMobileHeight = useMediaQuery("(max-height: 750px)");

Uses Material-UI's useMediaQuery to detect screen size and adjust styling dynamically.

### ****Fetching User & Booking Data on Mount****

useEffect(() => {

dispatch(meThunk());

dispatch(getBookingThunk({ order: "asc" }));

}, [dispatch]);

meThunk(): Fetches the logged-in user's data.

getBookingThunk({ order: "asc" }): Fetches the user's bookings in ascending order.

### ****Event Handlers****

**const handleClickOpen = () => setOpen(true);**

**const handleClickClose = () => setOpen(false);**

**const logoutLogic = () => dispatch(logoutThunk());**

handleClickOpen: Opens the logout confirmation dialog.

handleClickClose: Closes the logout confirmation dialog.

logoutLogic: Dispatches the logout action.

### ****Filtering Bookings by User & Status****

**const filteredBookingsOfTheUser: IBooking[] = booking.bookingList.filter(**

**(value) => value.user.\_id === user.\_id**

**);**

**const filteredBookingsOfTheUserNotCancelled: IBooking[] =**

**booking.bookingList.filter(**

**(value) => value.user.\_id === user.\_id && value.status === "booked"**

**);**

**const filteredBookingsOfTheUserCancelled: IBooking[] =**

**booking.bookingList.filter(**

**(value) => value.user.\_id === user.\_id && value.status === "cancelled"**

**);**

filteredBookingsOfTheUser**: Retrieves all bookings belonging to the user.**

filteredBookingsOfTheUserNotCancelled: Filters only active (non-cancelled) bookings.

filteredBookingsOfTheUserCancelled: Filters cancelled bookings.

## ****UI Structure****

### ****1. Logout Icon & Confirmation Dialog****

**<LogoutIcon onClick={handleClickOpen} />**

**<Dialog open={open} onClose={handleClickClose}>**

**<DialogTitle>Log out</DialogTitle>**

**<DialogContent>**

**<DialogContentText>Are you sure you want to logout?</DialogContentText>**

**</DialogContent>**

**<DialogActions>**

**<Button onClick={handleClickClose}>Cancel</Button>**

**<Button onClick={logoutLogic}>Okay</Button>**

**</DialogActions>**

**</Dialog>**

**Shows a logout icon that, when clicked, opens a confirmation dialog.**

Provides buttons to either cancel or proceed with logging out.

### ****2. Profile Section****

<Typography>{user.first\_name}’s Profile</Typography>

<Link to={`/editProfile/${user.\_id}`}>

<SettingsOutlinedIcon />

</Link>

<Box component="img" src={user.avatar\_url !== "" ? user.avatar\_url : icon} />

Displays the user’s name, avatar, and an edit profile button.

### ****3. User Details****

<Typography>Email: {user.email}</Typography>

<Typography>

Squad: {user.squad === "NOVA" ? "Nova" : user.squad === "NETWATCH" ? "Net Watch" : "Gizmo"}

</Typography>

<Typography>

Team: {user.team === "BND" ? "Backend Team" : user.team === "FND" ? "Frontend Team" : "Product Team"}

</Typography>

Displays user information such as email, squad, and team.

### ****4. Booking Statistics****

**<Typography>Total bookings: {filteredBookingsOfTheUser.length}</Typography>**

**<Typography>Not cancelled bookings: {filteredBookingsOfTheUserNotCancelled.length}</Typography>**

**<Typography>Cancelled bookings: {filteredBookingsOfTheUserCancelled.length}</Typography>**

**Displays the count of total, non-cancelled, and cancelled bookings.**

## ****Summary****

* **Manages state** using Redux (useAppSelector, useAppDispatch).
* **Fetches user & booking data** on component mount.
* **Implements responsive design** with useMediaQuery.
* **Provides user information** (name, email, squad, team).
* **Displays booking statistics** categorized by status.
* **Handles logout functionality** through a confirmation dialog.

## Conclusion

The Book Chair Office Seat Booking System represents a comprehensive solution for modern workplace management challenges. Through its implementation of the MERN stack with TypeScript, the system delivers a robust, scalable, and user-friendly platform for office space management.

### Key Achievements

* Successfully implemented a type-safe, full-stack application
* Created an intuitive user interface for seat booking and management
* Established a secure authentication and authorization system
* Developed a flexible and scalable database architecture
* Implemented real-time attendance tracking capabilities

### Technical Success Factors

* Strong type safety through TypeScript implementation
* Efficient state management using modern React patterns
* Scalable MongoDB database design
* Robust API architecture with Express.js
* Comprehensive testing coverage

The system's architecture and implementation provide a solid foundation for future enhancements while meeting current business requirements for flexible workspace management. The modular design ensures that new features can be added with minimal impact on existing functionality, making Book Chair a sustainable solution for long-term office space management needs.