ASMC Admin Panel - Comprehensive Documentation

A modern React.js-based admin panel for managing Anushaktinagar Sports Management Committee (ASMC) operations including members, bookings, events, payments, facilities, and comprehensive administrative functions.

Table of Contents

- Quick Start
- Architecture Overview
- Installation & Setup
- Environment Configuration
- Component Structure
- State Management
- API Integration
- UI Components
- Authentication & Authorization
- Dashboard Features
- Build & Deployment
- Development Guidelines
- Troubleshooting

Quick Start

Prerequisites

- **Node.js**: 18.0.0 or higher
- **npm**: 8.0.0 or higher
- **ASMC API**: Backend server running on port 7055

Local Development Setup

```
# Clone the repository
git clone <repository-url>
cd asmc-admin

# Install dependencies
npm install

# Set up environment variables
cp .env.example .env.local

# Configure environment variables (see Environment Configuration)
nano .env.local

# Start development server
npm start

# Visit the admin panel
open http://localhost:3000
```

Production Build

```
# Create production build
npm run build

# Serve the build (using serve package)
npx serve -s build -l 3000

# Or deploy to your web server
# Copy build/ folder to your web server root
```

Architecture Overview

Technology Stack

- **Frontend Framework**: React.js 18.2.0
- **State Management**: Redux Toolkit + RTK Query
- **UI Framework**: Material-UI (MUI) 5.13.7
- **Routing**: React Router DOM 6.11.2
- **HTTP Client**: Axios 1.4.0
- **Form Management**: Formik 2.4.1 + Yup 1.2.0
- **Charts**: Chart.js 4.4.4 + React-Chartjs-2
- **Rich Text Editor**: CKEditor 5
- **Date Handling**: date-fns 2.30.0
- **Build Tool**: Create React App

Project Structure

```
asmc-admin/
├── public/                 # Static assets
│   ├── index.html         # Main HTML template
│   ├── manifest.json      # PWA manifest
│   └── assets/           # Images and icons
├── src/
│   ├── components/        # Reusable UI components
│   │   ├── admin/        # Admin-specific components
│   │   ├── Common/       # Shared components
│   │   ├── layout/       # Layout components
│   │   └── public/       # Public components
│   ├── container/        # Container components (Redux connected)
│   │   ├── admin/        # Admin containers
│   │   ├── layout/       # Layout containers
│   │   └── public/       # Public containers
│   ├── pages/            # Page components
│   │   ├── admin/        # Admin pages
│   │   └── public/       # Public pages
│   ├── store/            # Redux store configuration
│   │   ├── members/      # Member management state
│   │   ├── booking/      # Booking management state
│   │   ├── events/       # Event management state
│   │   ├── facility/     # Facility management state
│   │   ├── staff/        # Staff management state
```

```
│   │   ├── common/       # Common state
│   │   └── documentation/ # Documentation state
│   ├── routes/           # Route configuration
│   ├── helpers/          # Utility functions
│   ├── hooks/            # Custom React hooks
│   └── index.js          # Application entry point
├── build/                # Production build output
├── docs/                 # Documentation files
└── package.json          # Dependencies and scripts
```

Component Architecture

The admin panel follows a **Container-Component pattern** with Redux integration:

1. **Pages**: Top-level route components
2. **Containers**: Redux-connected components with business logic
3. **Components**: Presentational components for UI rendering
4. **Store**: Redux store with RTK Query for API management

Installation & Setup

Environment Configuration

Create a `.env.local` file in the root directory:

```
# API Configuration
REACT_APP_API_BASE_URL=http://localhost:7055/api
REACT_APP_API_TIMEOUT=30000

# Application Configuration
REACT_APP_APP_NAME=ASMC Admin Panel
REACT_APP_VERSION=1.0.0

# Authentication
REACT_APP_JWT_STORAGE_KEY=asmc_admin_token
REACT_APP_REFRESH_TOKEN_KEY=asmc_admin_refresh

# Feature Flags
REACT_APP_ENABLE_ANALYTICS=true
REACT_APP_ENABLE_LOGGING=true

# Development
REACT_APP_DEBUG_MODE=true
```

Development Scripts

```
# Start development server
npm start

# Run tests
npm test

# Build for production
```

```
npm run build

# Eject from Create React App (not recommended)
npm run eject
```

🧩 Component Structure

Admin Components

The admin panel includes comprehensive management modules:

Core Management Modules

1. **Members Manager** (`/src/components/admin/members-manager/`)
    - Member registration and profiles
    - Family member management
    - Member status tracking
    - Payment history
2. **Booking Manager** (`/src/components/admin/booking-manager/`)
    - Hall booking management
    - Event booking system
    - Booking calendar view
    - Booking approval workflow
3. **Events Manager** (`/src/components/admin/events-manager/`)
    - Event creation and management
    - Event scheduling
    - Event registration tracking
    - Event reporting
4. **Facility Manager** (`/src/components/admin/facility-manager/`)
    - Facility information management
    - Facility availability tracking
    - Facility maintenance records
5. **Staff Manager** (`/src/components/admin/staff-manager/`)
    - Staff member management
    - Role and permission assignment
    - Staff activity tracking
6. **Payment Manager** (`/src/components/admin/payment-manager/`)
    - Payment processing
    - Payment verification
    - Financial reporting
    - Payment history

Additional Management Modules

- **Activity Manager**: Sports activity management
- **Halls Manager**: Hall and venue management
- **Plans Manager**: Membership plan management
- **Biometric Manager**: Attendance tracking

- **Reports Manager**: Comprehensive reporting
- **Documentation Manager**: System documentation access

Common Components

Reusable components located in `/src/components/Common/`:

- **Input Components**: Text inputs, selects, date pickers
- **Table Components**: Data tables with pagination and filtering
- **Modal Components**: Reusable modal dialogs
- **Upload Components**: File upload with image cropping
- **Chart Components**: Data visualization components
- **Permission Components**: Role-based access control

State Management

Redux Store Architecture

The application uses Redux Toolkit with RTK Query for state management:

Store Structure

```
// Store configuration
export const store = configureStore({
    reducer: {
        // Redux Slices
        common: commonSlice,
        members: membersSlice,
        booking: bookingSlice,
        events: eventsSlice,
        facility: facilitySlice,
        staff: staffSlice,
        documentation: documentationSlice,

        // RTK Query APIs
        membersApi: membersApis,
        commonApi: commonApis,
        bookingApi: bookingApis,
        eventsApi: eventsApis,
        facilityApi: facilityApis,
        staffApi: staffApis,
        documentationApi: documentationApi,
    },
    middleware: (getDefaultMiddleware) =>
        getDefaultMiddleware().concat([
            membersApis.middleware,
            commonApis.middleware,
            bookingApis.middleware,
            eventsApis.middleware,
            facilityApis.middleware,
            staffApis.middleware,
            documentationApi.middleware,
        ]),
});
```

RTK Query Integration

Each module has its own API slice for data fetching:

```
// Example: Members API
export const membersApi = createApi({
    reducerPath: 'membersApi',
    baseQuery: fetchBaseQuery({
        baseUrl: '/api/members',
        prepareHeaders: (headers, { getState }) => {
            const token = getState().auth.token;
            if (token) {
                headers.set('authorization', `Bearer ${token}`);
            }
            return headers;
        },
    }),
    tagTypes: ['Members'],
    endpoints: (builder) => ({
        getMembers: builder.query({
            query: (params) => ({
                url: '/',
                params,
            }),
            providesTags: ['Members'],
        }),
        createMember: builder.mutation({
            query: (memberData) => ({
                url: '/',
                method: 'POST',
                body: memberData,
            }),
            invalidatesTags: ['Members'],
        }),
    }),
});
```

API Integration

HTTP Client Configuration

The application uses Axios with interceptors for API communication:

```
// Axios configuration
const apiClient = axios.create({
    baseURL: process.env.REACT_APP_API_BASE_URL,
    timeout: process.env.REACT_APP_API_TIMEOUT,
    headers: {
        'Content-Type': 'application/json',
    },
});

// Request interceptor for authentication
```

```
apiClient.interceptors.request.use(
    (config) => {
        const token = getAuthToken();
        if (token) {
            config.headers.Authorization = `Bearer ${token}`;
        }
        return config;
    },
    (error) => Promise.reject(error),
);

// Response interceptor for error handling
apiClient.interceptors.response.use(
    (response) => response,
    (error) => {
        if (error.response?.status === 401) {
            // Handle token expiration
            logout();
        }
        return Promise.reject(error);
    },
);
```

API Endpoints

The admin panel integrates with the following API endpoints:

- **Members API**: `/api/members/*`
- **Booking API**: `/api/booking/*`
- **Events API**: `/api/events/*`
- **Facility API**: `/api/facility/*`
- **Staff API**: `/api/staff/*`
- **Authentication API**: `/api/auth/*`
- **Common API**: `/api/common/*`
- **Documentation API**: `/api/documentation/*`

UI Components

Material-UI Integration

The application uses Material-UI components with custom theming:

```
// Theme configuration
const theme = createTheme({
    palette: {
        primary: {
            main: '#1976d2',
        },
        secondary: {
            main: '#dc004e',
        },
    },
    typography: {
```

```
        fontFamily: '"Roboto", "Helvetica", "Arial", sans-serif',
    },
    components: {
        MuiButton: {
            styleOverrides: {
                root: {
                    textTransform: 'none',
                },
            },
        },
    },
});
```

Custom Components

Data Table Component

```
// Reusable data table with pagination and filtering
<Table
    data={members}
    columns={memberColumns}
    onEdit={handleEdit}
    onDelete={handleDelete}
    onPageChange={handlePageChange}
    onFilterChange={handleFilterChange}
    loading={isLoading}
    pagination={pagination}
/>
```

Form Components

```
// Form with validation
<Formik
    initialValues={initialValues}
    validationSchema={validationSchema}
    onSubmit={handleSubmit}
>
    {({ values, errors, touched, handleChange, handleBlur }) => (
        <Form>
            <Input
                name="name"
                label="Member Name"
                value={values.name}
                onChange={handleChange}
                onBlur={handleBlur}
                error={errors.name && touched.name}
                helperText={errors.name}
            />
        </Form>
    )}
</Formik>
```

Authentication & Authorization

JWT Authentication

The admin panel implements JWT-based authentication:

```
// Authentication flow
const login = async (credentials) => {
    const response = await authApi.login(credentials);
    const { token, refreshToken, user } = response.data;

    // Store tokens
    localStorage.setItem('authToken', token);
    localStorage.setItem('refreshToken', refreshToken);

    // Update Redux state
    dispatch(setAuthUser(user));
    dispatch(setAuthToken(token));
};
```

Role-Based Access Control

Permission-based component rendering:

```
// Permission wrapper component
const RequirePermission = ({ permission, children }) => {
    const userPermissions = useSelector(selectUserPermissions);

    if (!userPermissions.includes(permission)) {
        return <AccessDenied />;
    }

    return children;
};

// Usage
<RequirePermission permission="members:write">
    <CreateMemberButton />
</RequirePermission>;
```

Dashboard Features

Main Dashboard

The admin dashboard provides:

1. **Overview Cards**: Key metrics and statistics
2. **Recent Activities**: Latest system activities
3. **Quick Actions**: Fast access to common tasks
4. **Charts & Graphs**: Visual data representation
5. **Notifications**: System alerts and updates

Management Modules

Each management module includes:

- **List View**: Paginated data tables with filtering
- **Create/Edit Forms**: Comprehensive form handling
- **Detail Views**: Detailed information display
- **Bulk Operations**: Mass actions on selected items
- **Export Functionality**: Data export in multiple formats
- **Search & Filter**: Advanced search capabilities

Build & Deployment

Production Build

```
# Create optimized production build
npm run build

# Build output will be in the 'build' directory
# Contains optimized JS, CSS, and static assets
```

Deployment Options

1. Static File Hosting

```
# Copy build files to web server
cp -r build/* /var/www/html/

# Or use serve package for testing
npx serve -s build -l 3000
```

2. Nginx Configuration

```
server {
    listen 80;
    server_name admin.asmc.com;
    root /var/www/html/build;
    index index.html;

    # Handle client-side routing
    location / {
        try_files $uri $uri/ /index.html;
    }

    # API proxy
    location /api {
        proxy_pass http://localhost:7055;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

3. Docker Deployment

```dockerfile
# Dockerfile
FROM node:18-alpine as build
WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production
COPY . .
RUN npm run build

FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
COPY nginx.conf /etc/nginx/nginx.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Development Guidelines

Code Standards

1. **Component Structure**: Use functional components with hooks
2. **State Management**: Use Redux Toolkit for global state
3. **API Calls**: Use RTK Query for data fetching
4. **Styling**: Use Material-UI components and theming
5. **Form Handling**: Use Formik with Yup validation
6. **Error Handling**: Implement comprehensive error boundaries

File Naming Conventions

- **Components**: PascalCase (e.g., `MemberManager.jsx`)
- **Containers**: PascalCase with "Container" suffix
- **Pages**: PascalCase with "Page" suffix
- **Hooks**: camelCase with "use" prefix
- **Utilities**: camelCase
- **Constants**: UPPER_SNAKE_CASE

Component Development

```javascript
// Component template
import React from 'react';
import { Box, Typography } from '@mui/material';

const ComponentName = ({ prop1, prop2, onAction }) => {
    // Component logic

    return (
        <Box>
            <Typography variant="h6">Component Title</Typography>
            {/* Component JSX */}
        </Box>
    );
};

export default ComponentName;
```

Troubleshooting

Common Issues

1. Build Errors

```
# Clear node modules and reinstall
rm -rf node_modules package-lock.json
npm install

# Clear npm cache
npm cache clean --force
```

2. API Connection Issues

- Verify `REACT_APP_API_BASE_URL` in environment variables
- Check if backend server is running on correct port
- Verify CORS configuration in backend

3. Authentication Issues

- Check JWT token storage in localStorage
- Verify token expiration handling
- Check refresh token implementation

4. Performance Issues

- Use React DevTools Profiler to identify bottlenecks
- Implement React.memo for expensive components
- Optimize bundle size with code splitting

Debug Mode

Enable debug mode in development:

```
// Enable Redux DevTools
const store = configureStore({
    reducer: rootReducer,
    devTools: process.env.NODE_ENV !== 'production',
});

// Enable API debugging
if (process.env.REACT_APP_DEBUG_MODE === 'true') {
    console.log('API Response:', response);
}
```

Support

For technical support and questions:

- **Documentation**: Check this comprehensive guide
- **Issues**: Report bugs through the issue tracker
- **Development**: Follow the development guidelines
- **API Reference**: Refer to the backend API documentation

Version: 1.0.0
Last Updated: January 2025
Maintainer: ASMC Development Team