ASMC Next - Frontend Web Application

A comprehensive Next.js frontend application for the Anushaktinagar Sports Management Committee (ASMC). This web application provides a public-facing interface for members to access services, make bookings, and manage their sports activities.

Table of Contents

- Overview
- Features
- Technology Stack
- Project Structure
- Quick Start
- Development
- API Integration
- State Management
- Routing
- Components
- Styling
- Build & Deployment
- Troubleshooting

Overview

The ASMC Next application is a modern, responsive web frontend built with Next.js 14. It serves as the primary public interface for ASMC members and visitors, providing seamless access to sports facilities, event bookings, and membership management.

Key Capabilities

- **Public Website**: Informational pages, about us, contact, gallery
- **Member Portal**: Authentication, profile management, booking history
- **Booking System**: Sports activities, events, and hall reservations
- **Payment Integration**: Secure payment processing via CCAvenue
- **Mobile Responsive**: Optimized for all device types
- **SEO Optimized**: Server-side rendering and meta tag management

Features

Core Features

1. Authentication & Authorization

- Member login/logout
- Password reset functionality
- Session management with cookies
- Protected routes and middleware

2. Member Management

- Profile viewing and editing
- Family member management
- Membership status tracking
- Renewal notifications

3. Booking System

- **Sports Activities**: Book various sports activities
- **Events**: Register for tournaments and events
- **Halls**: Reserve sports halls and facilities
- **Team Events**: Multi-player event registration

4. Payment Processing

- CCAvenue payment gateway integration
- Secure payment handling
- Payment status tracking
- Receipt generation

5. Content Management

- Dynamic content display
- Image galleries
- News and notices
- FAQ management

Advanced Features

1. Responsive Design

- Mobile-first approach
- Bootstrap 5 framework
- Custom SCSS styling
- Cross-browser compatibility

2. Performance Optimization

- Next.js Image optimization
- Code splitting and lazy loading
- Static generation where possible
- CDN integration for assets

3. SEO & Analytics

- Server-side rendering
- Dynamic meta tags
- Structured data markup
- Google Analytics integration

4. Accessibility

- WCAG 2.1 compliance
- Keyboard navigation
- Screen reader support
- High contrast mode support

Technology Stack

Core Technologies

Frontend Framework

- **Next.js 14**: React framework with SSR/SSG
- **React 18**: Component library
- **Pages Router**: File-based routing system

State Management

- **Redux Toolkit**: Global state management
- **RTK Query**: Data fetching and caching
- **React Redux**: React bindings

Styling & UI

- **Bootstrap 5**: CSS framework
- **SCSS**: CSS preprocessor
- **Font Awesome**: Icon library
- **AOS**: Animation on scroll

Forms & Validation

- **Formik**: Form management
- **Yup**: Schema validation
- **React Datepicker**: Date selection

HTTP Client

- **Axios**: HTTP requests
- **Cookies-next**: Cookie management
- **Token handling**: JWT authentication

Additional Libraries

- **React Slick**: Carousel/slider
- **React Big Calendar**: Calendar component
- **HTML2Canvas & jsPDF**: PDF generation
- **React QR Code**: QR code generation
- **React Toastify**: Notifications

Project Structure

```
asmc-next/
├── apis/                          # API layer
│   ├── auth.api.js               # Authentication endpoints
│   ├── bookings.api.js           # Booking endpoints
│   ├── common.api.js             # Common endpoints
│   ├── events.api.js             # Event endpoints
│   ├── members.api.js            # Member endpoints
│   └── axios-config.js           # Axios configuration
├── components/                    # Reusable components
│   ├── about/                    # About page components
│   ├── auth/                     # Authentication components
│   ├── common/                   # Shared components
│   ├── event/                    # Event-related components
│   ├── facility/                 # Facility components
│   ├── home/                     # Home page components
│   ├── includes/                 # Layout components
│   └── payment/                  # Payment components
├── container/                     # Container components (Redux)
│   ├── ActivityDetails/          # Activity details container
│   ├── BookedActivity/           # Booked activities container
│   ├── BookedEvents/             # Booked events container
│   ├── BookedHalls/              # Booked halls container
│   ├── BookingDetails/           # Booking details container
│   ├── ChangePassword/           # Password change container
```

```
│   ├── ContactUs/                # Contact form container
│   ├── Dashboard/                # Dashboard container
│   ├── EventBookingDetails/      # Event booking details
│   ├── EventDetails/             # Event details container
│   ├── Events/                   # Events listing container
│   ├── Facility/                 # Facility container
│   ├── FaqsContainer/            # FAQ container
│   ├── GalleryContainer/         # Gallery container
│   ├── GuestBookedEvents/        # Guest events container
│   ├── HallBookingDetails/       # Hall booking details
│   ├── HallDetails/              # Hall details container
│   ├── Halls/                    # Halls listing container
│   ├── HomePage/                 # Home page container
│   ├── Membership/               # Membership container
│   ├── NoticesContainer/         # Notices container
│   ├── PendingPayment/           # Pending payments container
│   └── SportsActivity/           # Sports activities container
├── docs/                         # Documentation
│   ├── API_INTEGRATION_GUIDE.md  # API integration guide
│   ├── API_LAYER.md              # API layer documentation
│   ├── ARCHITECTURE.md           # Architecture overview
│   ├── BOOKING_FLOWS.md          # Booking flow documentation
│   ├── COMPONENTS.md             # Component documentation
│   ├── DEPLOYMENT.md             # Deployment guide
│   ├── ENVIRONMENT.md            # Environment setup
│   ├── FRONTEND_USER_GUIDE.md    # User guide
│   ├── ROUTING.md                # Routing documentation
│   ├── STATE_MANAGEMENT.md       # State management guide
│   ├── STYLE_GUIDE.md            # Style guide
│   └── TROUBLESHOOTING.md        # Troubleshooting guide
├── pages/                        # Next.js pages (routing)
│   ├── _app.js                   # App wrapper
│   ├── _document.js              # Document wrapper
│   ├── about-us/                 # About us pages
│   ├── booked-activity.js        # Booked activities page
│   ├── booked-events/            # Booked events pages
│   ├── booked-halls.js           # Booked halls page
│   ├── booking/                  # Booking pages
│   ├── change-password.js        # Password change page
│   ├── coming-soon.js            # Coming soon page
│   ├── contact-us.js             # Contact page
│   ├── dashboard.js              # Dashboard page
│   ├── events/                   # Events pages
│   ├── facilities/               # Facilities pages
│   ├── faqs.js                   # FAQ page
│   ├── forgot-password.js        # Password reset page
│   ├── gallery.js                # Gallery page
│   ├── index.js                  # Home page
│   ├── members/                  # Member pages
│   ├── membership.js             # Membership page
│   ├── notices.js                # Notices page
│   ├── payment-status.js         # Payment status page
```

```
│   ├── pending-payment.js        # Pending payments page
│   ├── privacy-policy.js         # Privacy policy page
│   ├── sign-in.js                # Sign in page
│   └── sports-activity.js        # Sports activities page
├── public/                       # Static assets
│   ├── images/                   # Image assets
│   ├── *.png                     # Favicon and app icons
│   └── *.pdf                     # Static documents
├── redux/                        # Redux store
│   ├── auth/                     # Authentication slice
│   ├── common/                   # Common slice
│   ├── masters/                  # Masters data slice
│   └── store.js                  # Store configuration
├── styles/                       # Styling
│   ├── glyphter/                 # Custom font icons
│   ├── main-.css                 # Main CSS file
│   └── style.scss                # Main SCSS file
├── utils/                        # Utility functions
│   ├── axios.js                  # Axios configuration
│   ├── constants.js              # Application constants
│   ├── helper.js                 # Helper functions
│   ├── toast.js                  # Toast notifications
│   └── tokenHandler.js           # Token management
├── next.config.mjs               # Next.js configuration
├── package.json                  # Dependencies and scripts
├── eslintrc.json                 # ESLint configuration
├── jsconfig.json                 # JavaScript configuration
└── README.md                     # Project documentation
```

Quick Start

Prerequisites

- **Node.js**: 18.0.0 or higher
- **npm**: 8.0.0 or higher
- **Git**: Latest version

Installation

1. **Clone the repository**

```
git clone <repository-url>
cd asmc-next
```

2. **Install dependencies**

```
npm install
```

3. **Environment setup**

```
# Create environment file
cp .env.example .env.local
```

```
# Configure environment variables
NEXT_PUBLIC_API_URL=http://localhost:7055
```

4. **Start development server**

```
npm run dev
```

5. **Access the application**
   - Open http://localhost:3000
   - The application will hot-reload on changes

Production Build

```
# Build for production
npm run build

# Start production server
npm start

# Or use PM2 for process management
pm2 start npm --name "asmc-next" -- start
```

Development

Development Scripts

```
# Development server with hot reload
npm run dev

# Production build
npm run build

# Start production server
npm start

# Lint code
npm run lint

# Lint and fix
npm run lint:fix
```

Code Quality

ESLint Configuration

- **Config**: `eslintrc.json`
- **Rules**: React, Next.js, and Prettier rules
- **Auto-fix**: Available via npm scripts

Prettier Configuration

- **Formatting**: Consistent code formatting
- **Integration**: Works with ESLint

- **Auto-format**: On save (if configured)

Development Workflow

1. **Feature Development**
    - Create feature branch from `main`
    - Implement feature with tests
    - Create pull request
2. **Code Review**
    - Automated linting and formatting
    - Manual code review
    - Testing verification
3. **Deployment**
    - Merge to `main` branch
    - Automated build and deployment
    - Health checks and monitoring

API Integration

API Layer Structure

The application uses a centralized API layer with RTK Query for efficient data fetching and caching.

API Configuration

```
// utils/axios.js
import axios from 'axios';
import { BaseUrl } from './constants';

const axiosInstance = axios.create({
    baseURL: BaseUrl,
    timeout: 30000,
    headers: {
        'Content-Type': 'application/json',
    },
});

// Request interceptor for authentication
axiosInstance.interceptors.request.use(
    (config) => {
        const token = getCookieData('token');
        if (token) {
            config.headers.Authorization = `Bearer ${token}`;
        }
        return config;
    },
    (error) => Promise.reject(error),
);
```

```
// Response interceptor for error handling
axiosInstance.interceptors.response.use(
    (response) => response,
    (error) => {
        if (error.response?.status === 401) {
            // Handle unauthorized access
            removeCookieData('token');
            window.location.href = '/sign-in';
        }
        return Promise.reject(error);
    },
);
```

API Slices

```
// redux/auth/authApis.js
import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react';

export const authApis = createApi({
    reducerPath: 'authApis',
    baseQuery: fetchBaseQuery({
        baseUrl: `${process.env.NEXT_PUBLIC_API_URL}/api`,
        prepareHeaders: (headers, { getState }) => {
            const token = getCookieData('token');
            if (token) {
                headers.set('authorization', `Bearer ${token}`);
            }
            return headers;
        },
    }),
    tagTypes: ['Auth', 'User'],
    endpoints: (builder) => ({
        login: builder.mutation({
            query: (credentials) => ({
                url: '/auth/login',
                method: 'POST',
                body: credentials,
            }),
            invalidatesTags: ['Auth', 'User'],
        }),
        logout: builder.mutation({
            query: () => ({
                url: '/auth/logout',
                method: 'POST',
            }),
            invalidatesTags: ['Auth', 'User'],
        }),
        getProfile: builder.query({
            query: () => '/auth/profile',
            providesTags: ['User'],
        }),
```

```
    }),
});
```

API Endpoints

Authentication

- `POST /auth/login` - Member login
- `POST /auth/logout` - Member logout
- `GET /auth/profile` - Get user profile
- `POST /auth/change-password` - Change password

Members

- `GET /members/profile` - Get member profile
- `PUT /members/profile` - Update member profile
- `GET /members/family` - Get family members
- `POST /members/family` - Add family member

Activities

- `GET /activities` - List activities
- `GET /activities/:id` - Get activity details
- `POST /activities/enroll` - Enroll in activity
- `GET /activities/enrolled` - Get enrolled activities

Events

- `GET /events` - List events
- `GET /events/:id` - Get event details
- `POST /events/register` - Register for event
- `GET /events/registered` - Get registered events

Halls

- `GET /halls` - List halls
- `GET /halls/:id` - Get hall details
- `POST /halls/book` - Book hall
- `GET /halls/bookings` - Get hall bookings

Payments

- `GET /payments/history` - Payment history
- `POST /payments/process` - Process payment
- `GET /payments/methods` - Available payment methods

State Management

Redux Store Architecture

The application uses Redux Toolkit with RTK Query for efficient state management and data fetching.

Store Configuration

```
// redux/store.js
import { combineReducers, configureStore } from '@reduxjs/toolkit';
import commonApis from './common/commonApis';
import mastersApis from './masters/mastersApis';
import authApis from './auth/authApis';
```

```
import commonSlice from './common/commonSlice';
import authSlice from './auth/authSlice';
import mastersSlice from './masters/mastersSlice';

const reducers = {
    [commonSlice.name]: commonSlice.reducer,
    [authSlice.name]: authSlice.reducer,
    [mastersSlice.name]: mastersSlice.reducer,

    [commonApis.reducerPath]: commonApis.reducer,
    [mastersApis.reducerPath]: mastersApis.reducer,
    [authApis.reducerPath]: authApis.reducer,
};

const rootReducer = combineReducers(reducers);

export const store = configureStore({
    reducer: rootReducer,
    middleware: (getDefaultMiddleware) => {
        return getDefaultMiddleware({
            serializableCheck: false,
        }).concat([commonApis.middleware, mastersApis.middleware,
authApis.middleware]);
    },
});
```

State Slices

Authentication Slice

```
// redux/auth/authSlice.js
import { createSlice } from '@reduxjs/toolkit';

const initialState = {
    isAuthenticated: false,
    user: null,
    token: null,
    loading: false,
    error: null,
};

const authSlice = createSlice({
    name: 'auth',
    initialState,
    reducers: {
        setCredentials: (state, action) => {
            state.isAuthenticated = true;
            state.user = action.payload.user;
            state.token = action.payload.token;
        },
        clearCredentials: (state) => {
            state.isAuthenticated = false;
```

```
            state.user = null;
            state.token = null;
        },
        setLoading: (state, action) => {
            state.loading = action.payload;
        },
        setError: (state, action) => {
            state.error = action.payload;
        },
    },
});

export const { setCredentials, clearCredentials, setLoading, setError } =
    authSlice.actions;
export default authSlice.reducer;
```

Common Slice

```
// redux/common/commonSlice.js
import { createSlice } from '@reduxjs/toolkit';

const initialState = {
    theme: 'light',
    language: 'en',
    notifications: [],
    loading: false,
    modals: {
        login: false,
        register: false,
        payment: false,
    },
};

const commonSlice = createSlice({
    name: 'common',
    initialState,
    reducers: {
        setTheme: (state, action) => {
            state.theme = action.payload;
        },
        setLanguage: (state, action) => {
            state.language = action.payload;
        },
        addNotification: (state, action) => {
            state.notifications.push(action.payload);
        },
        removeNotification: (state, action) => {
            state.notifications = state.notifications.filter(
                (notification) => notification.id !== action.payload,
            );
        },
        toggleModal: (state, action) => {
            const { modal, isOpen } = action.payload;
```

```
            state.modals[modal] = isOpen;
        },
    },
});

export const { setTheme, setLanguage, addNotification, removeNotification, toggleModal
} =
    commonSlice.actions;
export default commonSlice.reducer;
```

Data Flow

1. **Component Dispatch**: Components dispatch actions to Redux store
2. **Reducer Processing**: Reducers process actions and update state
3. **Component Re-render**: Components re-render based on state changes
4. **API Calls**: RTK Query handles API calls and caching
5. **Cache Management**: Automatic cache invalidation and updates

Routing

Next.js Pages Router

The application uses Next.js Pages Router for file-based routing with dynamic routes and API routes.

Route Structure

```
pages/
├── _app.js                # App wrapper (global layout)
├── _document.js           # Document wrapper (HTML structure)
├── index.js               # Home page (/)
├── sign-in.js             # Sign in page (/sign-in)
├── dashboard.js           # Dashboard (/dashboard)
├── about-us/
│   ├── index.js           # About us (/about-us)
│   ├── mission-vision.js  # Mission & vision (/about-us/mission-vision)
│   └── leadership.js      # Leadership (/about-us/leadership)
├── events/
│   ├── index.js           # Events listing (/events)
│   ├── [event_id].js      # Event details (/events/123)
│   └── booking/
│       └── [event_id].js  # Event booking (/events/booking/123)
├── facilities/
│   ├── index.js           # Facilities listing (/facilities)
│   ├── sports-activity.js # Sports activities (/facilities/sports-activity)
│   └── halls.js           # Halls listing (/facilities/halls)
├── booking/
│   ├── sports-booking/
│   │   └── [activity_id].js # Sports booking (/booking/sports-booking/123)
│   └── hall-booking/
│       └── [hall_id].js     # Hall booking (/booking/hall-booking/123)
├── members/
│   └── [member_id].js     # Member profile (/members/123)
```

```
├── booked-events/
│   ├── index.js            # Booked events (/booked-events)
│   └── [booking_id].js     # Event booking details (/booked-events/123)
├── booked-activity.js      # Booked activities (/booked-activity)
├── booked-halls.js         # Booked halls (/booked-halls)
├── payment-status.js       # Payment status (/payment-status)
├── pending-payment.js      # Pending payments (/pending-payment)
├── contact-us.js           # Contact page (/contact-us)
├── gallery.js              # Gallery (/gallery)
├── faqs.js                 # FAQ (/faqs)
├── notices.js              # Notices (/notices)
├── membership.js           # Membership (/membership)
├── forgot-password.js      # Password reset (/forgot-password)
├── change-password.js      # Change password (/change-password)
├── privacy-policy.js       # Privacy policy (/privacy-policy)
└── coming-soon.js          # Coming soon (/coming-soon)
```

Dynamic Routing

```jsx
// pages/events/[event_id].js
import { useRouter } from 'next/router';
import { useGetEventQuery } from '@/redux/common/commonApis';

const EventDetails = () => {
    const router = useRouter();
    const { event_id } = router.query;

    const { data: event, isLoading, error } = useGetEventQuery(event_id);

    if (isLoading) return <div>Loading...</div>;
    if (error) return <div>Error loading event</div>;
    if (!event) return <div>Event not found</div>;

    return (
        <div>
            <h1>{event.title}</h1>
            <p>{event.description}</p>
            {/* Event details content */}
        </div>
    );
};

export default EventDetails;
```

Protected Routes

```jsx
// utils/auth.js
import { getCookieData } from './helper';

export const withAuth = (WrappedComponent) => {
    return (props) => {
        const router = useRouter();
```

```
        const token = getCookieData('token');

        useEffect(() => {
            if (!token) {
                router.push('/sign-in');
            }
        }, [token, router]);

        if (!token) {
            return <div>Redirecting...</div>;
        }

        return <WrappedComponent {...props} />;
    };
};

// Usage in pages
// pages/dashboard.js
import { withAuth } from '@/utils/auth';

const Dashboard = () => {
    return <div>Dashboard content</div>;
};

export default withAuth(Dashboard);
```

API Routes

```
// pages/api/health.js
export default function handler(req, res) {
    res.status(200).json({
        status: 'healthy',
        timestamp: new Date().toISOString(),
        version: process.env.npm_package_version,
    });
}
```

Components

Component Architecture

The application follows a container-component pattern with Redux integration for state management.

Component Types

1. Presentation Components

- Pure React components
- Receive data via props
- Handle UI rendering and user interactions
- Located in `components/` directory

2. Container Components

- Connected to Redux store

- Handle data fetching and state management
- Pass data to presentation components
- Located in `container/` directory

3. Layout Components

- Provide page structure and navigation
- Include header, footer, and sidebar
- Handle global UI state
- Located in `components/includes/`

Component Examples

Presentation Component

```
// components/event/EventCard.js
import React from 'react';
import { Card, Button } from 'react-bootstrap';

const EventCard = ({ event, onRegister, onViewDetails }) => {
    return (
        <Card className="event-card">
            <Card.Img variant="top" src={event.image} />
            <Card.Body>
                <Card.Title>{event.title}</Card.Title>
                <Card.Text>{event.description}</Card.Text>
                <div className="event-actions">
                    <Button variant="primary" onClick={() => onViewDetails(event.id)}>
                        View Details
                    </Button>
                    <Button variant="success" onClick={() => onRegister(event.id)}>
                        Register
                    </Button>
                </div>
            </Card.Body>
        </Card>
    );
};

export default EventCard;
```

Container Component

```
// container/Events/EventsContainer.jsx
import React, { useEffect } from 'react';
import { connect } from 'react-redux';
import { useGetEventsQuery } from '@/redux/common/commonApis';
import EventCard from '@/components/event/EventCard';
import { setLoading, setError } from '@/redux/common/commonSlice';

const EventsContainer = ({ setLoading, setError, navigate }) => {
    const { data: events, isLoading, error } = useGetEventsQuery();

    useEffect(() => {
        setLoading(isLoading);
        if (error) {
```

```
            setError(error.message);
        }
    }, [isLoading, error, setLoading, setError]);

    const handleViewDetails = (eventId) => {
        navigate(`/events/${eventId}`);
    };

    const handleRegister = (eventId) => {
        navigate(`/events/booking/${eventId}`);
    };

    if (isLoading) return <div>Loading events...</div>;
    if (error) return <div>Error loading events</div>;

    return (
        <div className="events-container">
            <h1>Upcoming Events</h1>
            <div className="events-grid">
                {events?.map((event) => (
                    <EventCard
                        key={event.id}
                        event={event}
                        onViewDetails={handleViewDetails}
                        onRegister={handleRegister}
                    />
                ))}
            </div>
        </div>
    );
};

const mapDispatchToProps = {
    setLoading,
    setError,
};

export default connect(null, mapDispatchToProps)(EventsContainer);
```

Common Components

1. Layout Components

- `Header.js` - Navigation header
- `Footer.js` - Site footer
- `Sidebar.js` - Navigation sidebar

2. Form Components

- `InputBox.js` - Custom input component
- `ContactForm.js` - Contact form
- `ImageCropUpload.js` - Image upload with cropping

3. UI Components

- `Loader.js` - Loading spinner
- `Modal.js` - Modal dialog

- `ToastContainer.js` - Toast notifications
- `Pagination.js` - Pagination component

4. Feature Components

- `EventCard.js` - Event display card
- `ActivityCard.js` - Activity display card
- `HallsCard.js` - Hall display card
- `BookingModel.js` - Booking modal

Styling

CSS Architecture

The application uses a combination of Bootstrap 5 and custom SCSS for styling.

Styling Stack

- **Bootstrap 5**: CSS framework for responsive design
- **SCSS**: CSS preprocessor for advanced styling
- **Font Awesome**: Icon library
- **AOS**: Animation on scroll library
- **Custom CSS**: Component-specific styles

SCSS Structure

```scss
// styles/style.scss
@import 'bootstrap/scss/bootstrap';
@import 'aos/dist/aos.css';
@import 'slick-carousel/slick/slick.css';
@import 'slick-carousel/slick/slick-theme.css';

// Custom variables
$primary-color: #007bff;
$secondary-color: #6c757d;
$success-color: #28a745;
$danger-color: #dc3545;
$warning-color: #ffc107;
$info-color: #17a2b8;

// Custom mixins
@mixin flex-center {
    display: flex;
    justify-content: center;
    align-items: center;
}

@mixin button-variant($color, $background, $border) {
    color: $color;
    background-color: $background;
    border-color: $border;

    &:hover {
        background-color: darken($background, 7.5%);
        border-color: darken($border, 10%);
    }
```

```scss
}

// Component styles
.event-card {
    @include flex-center;
    margin-bottom: 1rem;

    .card-body {
        padding: 1.5rem;
    }

    .event-actions {
        margin-top: 1rem;

        .btn {
            margin-right: 0.5rem;
        }
    }
}

.booking-form {
    .form-group {
        margin-bottom: 1rem;

        label {
            font-weight: 600;
            margin-bottom: 0.5rem;
        }

        .form-control {
            border-radius: 0.375rem;
            border: 1px solid #ced4da;

            &:focus {
                border-color: $primary-color;
                box-shadow: 0 0 0 0.2rem rgba(0, 123, 255, 0.25);
            }
        }
    }
}

// Responsive design
@media (max-width: 768px) {
    .event-card {
        margin-bottom: 0.5rem;

        .event-actions {
            .btn {
                width: 100%;
                margin-right: 0;
                margin-bottom: 0.5rem;
            }
```

```
        }
    }
}
```

Bootstrap Customization

```
// Custom Bootstrap variables
$primary: #007bff;
$secondary: #6c757d;
$success: #28a745;
$danger: #dc3545;
$warning: #ffc107;
$info: #17a2b8;

$font-family-base: 'Inter', sans-serif;
$font-size-base: 1rem;
$line-height-base: 1.5;

$border-radius: 0.375rem;
$border-radius-sm: 0.25rem;
$border-radius-lg: 0.5rem;

$box-shadow: 0 0.125rem 0.25rem rgba(0, 0, 0, 0.075);
$box-shadow-sm: 0 0.125rem 0.25rem rgba(0, 0, 0, 0.075);
$box-shadow-lg: 0 1rem 3rem rgba(0, 0, 0, 0.175);
```

Responsive Design

```
// Breakpoints
$breakpoints: (
    xs: 0,
    sm: 576px,
    md: 768px,
    lg: 992px,
    xl: 1200px,
    xxl: 1400px,
);

// Mobile-first approach
.container {
    @media (min-width: 576px) {
        max-width: 540px;
    }

    @media (min-width: 768px) {
        max-width: 720px;
    }

    @media (min-width: 992px) {
        max-width: 960px;
    }
```

```
    @media (min-width: 1200px) {
        max-width: 1140px;
    }
}

// Responsive utilities
.d-mobile-none {
    @media (max-width: 767px) {
        display: none !important;
    }
}

.d-mobile-block {
    @media (max-width: 767px) {
        display: block !important;
    }
}
```

Build & Deployment

Build Process

Development Build

```
# Start development server
npm run dev

# Development server runs on http://localhost:3000
# Hot reload enabled for development
```

Production Build

```
# Build for production
npm run build

# Start production server
npm start

# Or use PM2 for process management
pm2 start npm --name "asmc-next" -- start
```

Build Optimization

```
// next.config.mjs
const nextConfig = {
    reactStrictMode: true,
    images: {
        domains: ['localhost', 'api.asmcdae.in', 'ik.imagekit.io'],
        formats: ['image/webp', 'image/avif'],
    },
    env: {
        BASE_URL: process.env.BASE_URL,
```

```
        NEXT_PUBLIC_API_URL: process.env.NEXT_PUBLIC_API_URL,
    },
    // Performance optimizations
    swcMinify: true,
    compress: true,
    poweredByHeader: false,

    // Bundle analysis
    webpack: (config, { isServer }) => {
        if (!isServer) {
            config.resolve.fallback = {
                ...config.resolve.fallback,
                fs: false,
            };
        }
        return config;
    },
};

export default nextConfig;
```

Deployment Options

1. Static Export (Recommended)

```
# Build static export
npm run build
npm run export

# Deploy to CDN or static hosting
# Files will be in 'out' directory
```

2. Server Deployment

```
# Build application
npm run build

# Start production server
npm start

# Or use PM2
pm2 start npm --name "asmc-next" -- start
pm2 save
pm2 startup
```

3. Docker Deployment

```
# Dockerfile
FROM node:18-alpine AS base

# Install dependencies only when needed
FROM base AS deps
```

```
RUN apk add --no-cache libc6-compat
WORKDIR /app

COPY package.json package-lock.json ./
RUN npm ci --only=production

# Rebuild the source code only when needed
FROM base AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY . .

RUN npm run build

# Production image, copy all the files and run next
FROM base AS runner
WORKDIR /app

ENV NODE_ENV production

RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

COPY --from=builder /app/public ./public
COPY --from=builder --chown=nextjs:nodejs /app/.next/standalone ./
COPY --from=builder --chown=nextjs:nodejs /app/.next/static ./.next/static

USER nextjs

EXPOSE 3000

ENV PORT 3000

CMD ["node", "server.js"]
```

4. Nginx Configuration

```
# /etc/nginx/sites-available/asmc-next
server {
    listen 80;
    server_name asmcdae.in www.asmcdae.in;

    # Redirect HTTP to HTTPS
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name asmcdae.in www.asmcdae.in;

    # SSL configuration
    ssl_certificate /etc/letsencrypt/live/asmcdae.in/fullchain.pem;
```

```
    ssl_certificate_key /etc/letsencrypt/live/asmcdae.in/privkey.pem;

    # Security headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin" always;

    # Gzip compression
    gzip on;
    gzip_vary on;
    gzip_min_length 1024;
    gzip_types text/plain text/css text/xml text/javascript application/javascript
application/xml+rss application/json;

    # Static files
    location /_next/static/ {
        alias /var/www/asmc-next/.next/static/;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # API routes
    location /api/ {
        proxy_pass http://localhost:7055/api/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }

    # Main application
    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Environment Configuration

Environment Variables

```
# .env.local
NEXT_PUBLIC_API_URL=https://api.asmcdae.in
NEXT_PUBLIC_APP_NAME=ASMC
NEXT_PUBLIC_APP_VERSION=1.0.0

# Production
NODE_ENV=production
PORT=3000
```

Environment-specific Builds

```
# Development
npm run dev

# Staging
NODE_ENV=staging npm run build
NODE_ENV=staging npm start

# Production
NODE_ENV=production npm run build
NODE_ENV=production npm start
```

Troubleshooting

Common Issues

1. Build Errors

Problem: Build fails with dependency errors

```
# Solution: Clear cache and reinstall
rm -rf node_modules package-lock.json
npm cache clean --force
npm install
npm run build
```

Problem: Memory issues during build

```
# Solution: Increase Node.js memory
export NODE_OPTIONS="--max-old-space-size=4096"
npm run build
```

2. Runtime Errors

Problem: Hydration mismatch

```
// Solution: Use dynamic imports for client-only components
import dynamic from 'next/dynamic';

const ClientOnlyComponent = dynamic(() => import('../components/ClientOnlyComponent'),
{
    ssr: false,
});
```

Problem: API connection issues

```
// Solution: Check environment variables
console.log('API URL:', process.env.NEXT_PUBLIC_API_URL);

// Verify API endpoint
fetch(`${process.env.NEXT_PUBLIC_API_URL}/health`)
    .then((response) => response.json())
    .then((data) => console.log('API Health:', data));
```

3. Performance Issues

Problem: Slow page loads

```
// Solution: Implement code splitting
import dynamic from 'next/dynamic';

const HeavyComponent = dynamic(() => import('../components/HeavyComponent'), {
    loading: () => <div>Loading...</div>,
    ssr: false,
});

// Use Next.js Image component
import Image from 'next/image';

<Image
    src="/images/hero.jpg"
    alt="Hero image"
    width={800}
    height={600}
    priority
    placeholder="blur"
    blurDataURL="..."
/>;
```

4. Redux Issues

Problem: State not updating

```
// Solution: Check Redux DevTools
import { createStore } from '@reduxjs/toolkit';

const store = createStore(
    rootReducer,
    window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__(),
);

// Verify action dispatching
console.log('Dispatching action:', action);
dispatch(action);
console.log('New state:', store.getState());
```

Debug Tools

1. Next.js Debug Mode

```
# Enable debug mode
DEBUG=* npm run dev

# Or specific debug namespaces
DEBUG=next:* npm run dev
```

2. Redux DevTools

```
// Install Redux DevTools Extension
// Available in Chrome/Firefox extensions

// Configure store with DevTools
import { composeWithDevTools } from '@redux-devtools-extension';

const store = createStore(rootReducer, composeWithDevTools());
```

3. Performance Monitoring

```
// Add performance monitoring
import { getCLS, getFID, getFCP, getLCP, getTTFB } from 'web-vitals';

function sendToAnalytics(metric) {
    // Send to analytics service
    console.log(metric);
}

getCLS(sendToAnalytics);
getFID(sendToAnalytics);
getFCP(sendToAnalytics);
getLCP(sendToAnalytics);
getTTFB(sendToAnalytics);
```

Support Resources

1. Documentation

- Next.js Documentation
- Redux Toolkit Documentation
- Bootstrap Documentation

2. Community

- Next.js GitHub
- Redux GitHub
- React GitHub

3. Tools

- Next.js Analytics
- Redux DevTools
- React Developer Tools

Additional Documentation

For more detailed information, refer to the following documentation files:

- **Architecture**: `docs/ARCHITECTURE.md`
- **API Integration**: `docs/API_INTEGRATION_GUIDE.md`
- **State Management**: `docs/STATE_MANAGEMENT.md`
- **Component Guide**: `docs/COMPONENTS.md`
- **Deployment**: `docs/DEPLOYMENT.md`
- **Troubleshooting**: `docs/TROUBLESHOOTING.md`

ASMC Next Frontend Application

A modern, scalable, and maintainable web application built with Next.js, providing an excellent user experience for ASMC members and visitors.

Last Updated: January 2025
Maintainer: ASMC Development Team