ASMC Next - Installation & Setup Guide

Comprehensive guide for setting up the ASMC Next.js frontend application development environment, including prerequisites, installation steps, configuration, and troubleshooting.

󰟀 Table of Contents

Prerequisites

System Requirements

Minimum Requirements

- **Operating System**: Windows 10+, macOS 10.15+, or Ubuntu 18.04+
- **RAM**: 4GB minimum, 8GB recommended
- **Storage**: 10GB free space
- **Node.js**: 18.0.0 or higher
- **npm**: 8.0.0 or higher

Recommended Requirements

- **Operating System**: Windows 11, macOS 12+, or Ubuntu 20.04+
- **RAM**: 8GB or higher
- **Storage**: 20GB free space
- **Node.js**: 18.17.0 or higher
- **npm**: 9.0.0 or higher

Required Software

Core Software

1. **Node.js**

    - **Version**: 18.0.0 or higher
    - **Download**: Node.js Official Website
    - **LTS Version**: Recommended for production

2. **npm**

    - **Version**: 8.0.0 or higher
    - **Included**: Comes with Node.js installation

3. **Git**

    - **Version**: 2.30.0 or higher
    - **Download**: Git Official Website

Optional Software

1. **VS Code** (Recommended)
   - **Download**: Visual Studio Code
   - **Extensions**: ES7+ React/Redux/React-Native snippets, Prettier, ESLint
2. **Chrome DevTools**
   - **Purpose**: Debugging and development
   - **Extensions**: React Developer Tools, Redux DevTools

Environment Setup

Node.js Installation

Using Node Version Manager (Recommended)

Windows (using nvm-windows):

```
# Download and install nvm-windows from:
# https://github.com/coreybutler/nvm-windows/releases

# Install Node.js 18
nvm install 18.17.0
nvm use 18.17.0
nvm alias default 18.17.0

# Verify installation
node --version  # Should output v18.17.0
npm --version   # Should output 9.x.x
```

macOS/Linux (using nvm):

```
# Install nvm
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh | bash

# Restart terminal or source profile
source ~/.bashrc  # or ~/.zshrc

# Install and use Node.js 18
nvm install 18.17.0
nvm use 18.17.0
nvm alias default 18.17.0

# Verify installation
node --version  # Should output v18.17.0
npm --version   # Should output 9.x.x
```

Direct Installation

1. Visit Node.js Official Website
2. Download the LTS version for your operating system
3. Run the installer and follow the setup wizard
4. Verify installation:

```
node --version
npm --version
```

Git Setup

Install Git

Windows:

1. Download Git from git-scm.com
2. Run the installer with default settings
3. Open Git Bash to verify installation

macOS:

```
# Install Xcode Command Line Tools (includes Git)
xcode-select --install

# Or install via Homebrew
brew install git
```

Ubuntu/Debian:

```
sudo apt update
sudo apt install git
```

Configure Git

```
# Set user information
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"

# Set default branch name
git config --global init.defaultBranch main

# Verify configuration
git config --list
```

Development Tools Setup

VS Code Extensions

Install the following recommended extensions:

```
# Essential extensions
code --install-extension ms-vscode.vscode-typescript-next
code --install-extension bradlc.vscode-tailwindcss
code --install-extension esbenp.prettier-vscode
code --install-extension ms-vscode.vscode-eslint
code --install-extension dsznajder.es7-react-js-snippets
code --install-extension ms-vscode.vscode-json
code --install-extension redhat.vscode-yaml
```

Browser Extensions

1. **Chrome Extensions**:
    - React Developer Tools
    - Redux DevTools
    - Lighthouse (for performance testing)
2. **Firefox Extensions**:
    - React Developer Tools
    - Redux DevTools

Project Installation

1. Clone Repository

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

# Check repository status
git status
git branch -a
```

2. Install Dependencies

```bash
# Install npm dependencies
npm install

# Verify installation
npm list --depth=0
```

3. Verify Project Structure

```bash
# Check project structure
ls -la

# Verify key files exist
ls -la package.json
ls -la next.config.mjs
ls -la pages/
ls -la components/
ls -la redux/
```

4. Check Next.js Environment

```bash
# Check Next.js version
npx next --version

# Verify Next.js configuration
cat next.config.mjs
```

Configuration

Environment Configuration

1. Create Environment Files

```bash
# Create environment files
touch .env.local
touch .env.development
touch .env.production
```

2. Configure Environment Variables

.env.local (Local development):

```
# API Configuration
NEXT_PUBLIC_API_URL=http://localhost:7055

# App Configuration
NEXT_PUBLIC_APP_NAME=ASMC
NEXT_PUBLIC_APP_VERSION=1.0.0

# Feature Flags
NEXT_PUBLIC_ENABLE_DEBUG=true
NEXT_PUBLIC_ENABLE_ANALYTICS=false

# Development Settings
NODE_ENV=development
```

.env.development:

```
# Development API
NEXT_PUBLIC_API_URL=http://localhost:7055

# Debug settings
NEXT_PUBLIC_ENABLE_DEBUG=true
NEXT_PUBLIC_ENABLE_ANALYTICS=false
```

.env.production:

```
# Production API
NEXT_PUBLIC_API_URL=https://api.asmcdae.in

# Production settings
NEXT_PUBLIC_ENABLE_DEBUG=false
NEXT_PUBLIC_ENABLE_ANALYTICS=true
```

3. Update Constants File

utils/constants.js:

```javascript
export const BaseUrl = process.env.NEXT_PUBLIC_API_URL || 'http://localhost:7055';
export const appConfig = {
    name: process.env.NEXT_PUBLIC_APP_NAME || 'ASMC',
    version: process.env.NEXT_PUBLIC_APP_VERSION || '1.0.0',
    enableDebug: process.env.NEXT_PUBLIC_ENABLE_DEBUG === 'true',
    enableAnalytics: process.env.NEXT_PUBLIC_ENABLE_ANALYTICS === 'true',
};

export const apiEndpoints = {
    auth: {
        login: '/auth/login',
        logout: '/auth/logout',
        profile: '/auth/profile',
        changePassword: '/auth/change-password',
    },
    members: {
        profile: '/members/profile',
        update: '/members/update',
        family: '/members/family',
    },
    activities: {
        list: '/activities',
        detail: '/activities',
        enroll: '/activities/enroll',
        enrolled: '/activities/enrolled',
    },
    events: {
        list: '/events',
        detail: '/events',
        register: '/events/register',
        registered: '/events/registered',
    },
    halls: {
        list: '/halls',
        detail: '/halls',
        book: '/halls/book',
        bookings: '/halls/bookings',
    },
    payments: {
        history: '/payments/history',
        process: '/payments/process',
        methods: '/payments/methods',
    },
};
```

Next.js Configuration

1. Update Next.js Config

next.config.mjs:

```
const nextConfig = {
    reactStrictMode: true,
    swcMinify: true,
    compress: true,
    poweredByHeader: false,

    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
    experimental: {
        optimizeCss: true,
        optimizePackageImports: ['@fortawesome/react-fontawesome'],
    },

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

        // Bundle analyzer (uncomment for analysis)
        // if (process.env.ANALYZE === 'true') {
        //     const { BundleAnalyzerPlugin } = require('webpack-bundle-analyzer');
        //     config.plugins.push(
        //         new BundleAnalyzerPlugin({
        //             analyzerMode: 'server',
        //             openAnalyzer: true,
        //         })
        //     );
        // }

        return config;
    },

    // Headers for security
    async headers() {
        return [
            {
                source: '/(.*)',
                headers: [
                    {
                        key: 'X-Frame-Options',
                        value: 'DENY',
                    },
                    {
                        key: 'X-Content-Type-Options',
                        value: 'nosniff',
                    },
                    {
                        key: 'Referrer-Policy',
                        value: 'strict-origin-when-cross-origin',
                    },
                ],
            },
        ];
    },
};

export default nextConfig;
```

2. Configure TypeScript (Optional)

tsconfig.json:

```
{
    "compilerOptions": {
        "target": "es5",
        "lib": ["dom", "dom.iterable", "es6"],
        "allowJs": true,
        "skipLibCheck": true,
        "strict": true,
        "forceConsistentCasingInFileNames": true,
        "noEmit": true,
        "esModuleInterop": true,
        "module": "esnext",
        "moduleResolution": "node",
        "resolveJsonModule": true,
        "isolatedModules": true,
        "jsx": "preserve",
        "incremental": true,
        "plugins": [
            {
                "name": "next"
            }
        ],
        "baseUrl": ".",
        "paths": {
            "@/*": ["./*"]
        }
    },
    "include": ["next-env.d.ts", "**/*.ts", "**/*.tsx", ".next/types/**/*.ts"],
    "exclude": ["node_modules"]
}
```

Redux Configuration

1. 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: {
                ignoredActions: [
                    'persist/PERSIST',
                    'persist/REHYDRATE',
                    'persist/PAUSE',
                    'persist/PURGE',
                    'persist/REGISTER',
                ],
            },
        }).concat([commonApis.middleware, mastersApis.middleware,
authApis.middleware]);
    },
    devTools: process.env.NODE_ENV !== 'production',
});

export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

2. API Configuration

utils/axios.js:

```
import axios from 'axios';
import { getCookieData, removeCookieData } from './helper';
import { BaseUrl } from './constants';
```

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

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

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

export default axiosInstance;
```

Development Setup

1. Start Development Server

```
# Start development server
npm run dev

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

2. Development Commands

```
# Development server with custom port
npm run dev -- -p 3001

# Development server with debug mode
```

```
DEBUG=* npm run dev

# Development server with verbose logging
npm run dev -- --verbose
```

3. Code Quality Tools

ESLint Configuration

.eslintrc.json:

```
{
    "extends": ["next/core-web-vitals", "prettier"],
    "rules": {
        "react/no-unescaped-entities": "off",
        "@next/next/no-page-custom-font": "off",
        "no-unused-vars": "warn",
        "no-console": "warn",
        "prefer-const": "error",
        "no-var": "error"
    },
    "ignorePatterns": ["node_modules/", ".next/", "out/", "build/"]
}
```

Prettier Configuration

.prettierrc:

```
{
    "semi": true,
    "trailingComma": "es5",
    "singleQuote": true,
    "printWidth": 80,
    "tabWidth": 2,
    "useTabs": false,
    "bracketSpacing": true,
    "arrowParens": "avoid"
}
```

4. Development Workflow

1. Feature Development

```
# Create feature branch
git checkout -b feature/new-feature

# Make changes and commit
git add .
git commit -m "feat: add new feature"

# Push to remote
git push origin feature/new-feature
```

2. Code Quality Checks

```bash
# Run linting
npm run lint

# Fix linting issues
npm run lint:fix

# Format code
npm run format

# Type checking (if using TypeScript)
npm run type-check
```

3. Testing

```bash
# Run tests
npm test

# Run tests in watch mode
npm run test:watch

# Run tests with coverage
npm run test:coverage
```

Testing Setup

1. Install Testing Dependencies

```bash
# Install testing dependencies
npm install --save-dev jest @testing-library/react @testing-library/jest-dom jest-
environment-jsdom

# Install additional testing utilities
npm install --save-dev @testing-library/user-event msw
```

2. Configure Jest

jest.config.js:

```javascript
const nextJest = require('next/jest');

const createJestConfig = nextJest({
    // Provide the path to your Next.js app to load next.config.js and .env files
    dir: './',
});

// Add any custom config to be passed to Jest
const customJestConfig = {
    setupFilesAfterEnv: ['<rootDir>/jest.setup.js'],
    testEnvironment: 'jest-environment-jsdom',
    testPathIgnorePatterns: ['<rootDir>/.next/', '<rootDir>/node_modules/'],
    moduleNameMapping: {
        '^@/(.*)$': '<rootDir>/$1',
    },
    collectCoverageFrom: [
        'components/**/*.{js,jsx}',
        'pages/**/*.{js,jsx}',
        'utils/**/*.{js,jsx}',
        '!**/*.d.ts',
        '!**/node_modules/**',
    ],
    coverageThreshold: {
        global: {
            branches: 70,
            functions: 70,
            lines: 70,
            statements: 70,
        },
    },
};

// createJestConfig is exported this way to ensure that next/jest can load the Next.js
config which is async
module.exports = createJestConfig(customJestConfig);
```

3. Jest Setup File

jest.setup.js:

```
import '@testing-library/jest-dom';

// Mock Next.js router
jest.mock('next/router', () => ({
    useRouter() {
        return {
            route: '/',
            pathname: '/',
            query: {},
            asPath: '/',
            push: jest.fn(),
            pop: jest.fn(),
            reload: jest.fn(),
            back: jest.fn(),
            prefetch: jest.fn().mockResolvedValue(undefined),
            beforePopState: jest.fn(),
            events: {
                on: jest.fn(),
                off: jest.fn(),
                emit: jest.fn(),
            },
        };
    },
}));

// Mock window.matchMedia
```

```
Object.defineProperty(window, 'matchMedia', {
    writable: true,
    value: jest.fn().mockImplementation((query) => ({
        matches: false,
        media: query,
        onchange: null,
        addListener: jest.fn(),
        removeListener: jest.fn(),
        addEventListener: jest.fn(),
        removeEventListener: jest.fn(),
        dispatchEvent: jest.fn(),
    })),
});
```

4. Test Utilities

tests/test-utils.js:

```
import React from 'react';
import { render } from '@testing-library/react';
import { Provider } from 'react-redux';
import { store } from '../redux/store';

const AllTheProviders = ({ children }) => {
    return <Provider store={store}>{children}</Provider>;
};

const customRender = (ui, options) =>
    render(ui, { wrapper: AllTheProviders, ...options });

// re-export everything
export * from '@testing-library/react';

// override render method
export { customRender as render };
```

5. Write Tests

tests/components/EventCard.test.js:

```
import React from 'react';
import { render, screen, fireEvent } from '../test-utils';
import EventCard from '../../components/event/EventCard';

const mockEvent = {
    id: 1,
    title: 'Test Event',
    description: 'Test Description',
    image: '/test-image.jpg',
    date: '2025-01-15',
    location: 'Test Location',
};

```

```
describe('EventCard Component', () => {
    it('renders event information correctly', () => {
        const mockOnViewDetails = jest.fn();
        const mockOnRegister = jest.fn();

        render(
            <EventCard
                event={mockEvent}
                onViewDetails={mockOnViewDetails}
                onRegister={mockOnRegister}
            />,
        );

        expect(screen.getByText('Test Event')).toBeInTheDocument();
        expect(screen.getByText('Test Description')).toBeInTheDocument();
        expect(screen.getByText('Test Location')).toBeInTheDocument();
    });

    it('calls onViewDetails when view details button is clicked', () => {
        const mockOnViewDetails = jest.fn();
        const mockOnRegister = jest.fn();

        render(
            <EventCard
                event={mockEvent}
                onViewDetails={mockOnViewDetails}
                onRegister={mockOnRegister}
            />,
        );

        fireEvent.click(screen.getByText('View Details'));
        expect(mockOnViewDetails).toHaveBeenCalledWith(1);
    });

    it('calls onRegister when register button is clicked', () => {
        const mockOnViewDetails = jest.fn();
        const mockOnRegister = jest.fn();

        render(
            <EventCard
                event={mockEvent}
                onViewDetails={mockOnViewDetails}
                onRegister={mockOnRegister}
            />,
        );

        fireEvent.click(screen.getByText('Register'));
        expect(mockOnRegister).toHaveBeenCalledWith(1);
    });
});
```

6. Run Tests

```bash
# Run all tests
npm test

# Run tests in watch mode
npm run test:watch

# Run tests with coverage
npm run test:coverage

# Run specific test file
npm test EventCard.test.js
```

Production Build

1. Build for Production

```bash
# Build for production
npm run build

# Build with bundle analysis
ANALYZE=true npm run build

# Build with custom output directory
npm run build -- --outdir dist
```

2. Start Production Server

```bash
# Start production server
npm start

# Start with custom port
PORT=3001 npm start

# Start with PM2
pm2 start npm --name "asmc-next" -- start
```

3. PM2 Configuration

ecosystem.config.js:

```javascript
module.exports = {
    apps: [
        {
            name: 'asmc-next',
            script: 'npm',
            args: 'start',
            instances: 'max',
            exec_mode: 'cluster',
            env: {
                NODE_ENV: 'production',
                PORT: 3000,
            },
            env_production: {
                NODE_ENV: 'production',
                PORT: 3000,
            },
            error_file: './logs/err.log',
            out_file: './logs/out.log',
            log_file: './logs/combined.log',
            time: true,
            max_memory_restart: '1G',
            node_args: '--max-old-space-size=4096',
        },
    ],
};
```

4. Docker Configuration

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 . .

ENV NEXT_TELEMETRY_DISABLED 1

RUN npm run build

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

ENV NODE_ENV production
ENV NEXT_TELEMETRY_DISABLED 1

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

COPY --from=builder /app/public ./public

# Set the correct permission for prerender cache
RUN mkdir .next
RUN chown nextjs:nodejs .next

# Automatically leverage output traces to reduce image size
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
ENV HOSTNAME "0.0.0.0"

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

docker-compose.yml:

```
version: '3.8'

services:
    asmc-next:
        build: .
        ports:
            - '3000:3000'
        environment:
            - NODE_ENV=production
            - NEXT_PUBLIC_API_URL=https://api.asmcdae.in
        volumes:
            - ./logs:/app/logs
        restart: unless-stopped
        healthcheck:
            test: ['CMD', 'curl', '-f', 'http://localhost:3000/api/health']
            interval: 30s
            timeout: 10s
            retries: 3
```

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

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

# Or use cross-env for cross-platform
npm install --save-dev cross-env
# Then in package.json:
# "build": "cross-env NODE_OPTIONS='--max-old-space-size=4096' next build"
```

Problem: Module not found errors

```bash
# Solution: Check import paths and file structure
# Verify file exists and path is correct
# Check for typos in import statements
```

2. Runtime Errors

Problem: Hydration mismatch

```javascript
// 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

```javascript
// 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));
```

Problem: Redux DevTools not working

```javascript
// Solution: Enable DevTools in development
export const store = configureStore({
    reducer: rootReducer,
    middleware: (getDefaultMiddleware) =>
        getDefaultMiddleware({
            serializableCheck: false,
        }).concat([...apiMiddlewares]),
    devTools: process.env.NODE_ENV !== 'production',
});
```

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="data:image/jpeg;base64,..."
/>;
```

Problem: Large bundle size

```
# Solution: Analyze bundle
npm install --save-dev @next/bundle-analyzer

# Add to next.config.mjs
const withBundleAnalyzer = require('@next/bundle-analyzer')({
    enabled: process.env.ANALYZE === 'true',
});

module.exports = withBundleAnalyzer(nextConfig);

# Run analysis
ANALYZE=true npm run build
```

4. Development Issues

Problem: Hot reload not working

```
# Solution: Check file watching limits
# Linux/macOS
echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf
sudo sysctl -p

# Restart development server
npm run dev
```

Problem: Port already in use

```
# Solution: Use different port
npm run dev -- -p 3001
```

```
# Or kill process using port
# Find process
lsof -ti:3000
# Kill process
kill -9 $(lsof -ti:3000)
```

Debug Commands

Environment Check

```
# Check Node.js version
node --version

# Check npm version
npm --version

# Check Next.js version
npx next --version

# Check environment variables
echo $NODE_ENV
echo $NEXT_PUBLIC_API_URL
```

Development Debugging

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

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

# Enable verbose logging
npm run dev -- --verbose
```

Build Debugging

```
# Build with debug info
DEBUG=* npm run build

# Build with bundle analysis
ANALYZE=true npm run build

# Check build output
ls -la .next/
```

Performance Optimization

Bundle Analysis

```
# Install bundle analyzer
npm install --save-dev @next/bundle-analyzer
```

```
# Add to package.json
"analyze": "ANALYZE=true npm run build"

# Run analysis
npm run analyze
```

Performance Monitoring

```
// Add to _app.js
import { getCLS, getFID, getFCP, getLCP, getTTFB } from 'web-vitals';

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

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

Support Resources

Documentation

- Next.js Documentation
- Redux Toolkit Documentation
- React Documentation

Community

- Next.js GitHub
- Redux GitHub
- React GitHub

Tools

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

Summary

This comprehensive installation and setup guide covers:

- **Complete Environment Setup**: Node.js, Git, and development tools
- **Project Configuration**: Environment variables, Next.js config, and Redux setup
- **Development Workflow**: Running the app, debugging, and testing
- **Production Builds**: Creating optimized builds for deployment
- **Troubleshooting**: Common issues and their solutions

Following this guide ensures a smooth development experience for the ASMC Next.js application.

Last Updated: January 2025
Maintainer: ASMC Development Team