ASMC Mobile App - Comprehensive Documentation

A modern React Native mobile application for the Anushaktinagar Sports Management Committee (ASMC), providing members with access to activities, events, hall bookings, and comprehensive membership management features.

Table of Contents

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

Quick Start

Prerequisites

- **Node.js**: 18.0.0 or higher
- **React Native CLI**: Latest version
- **Android Studio**: For Android development
- **Java Development Kit (JDK)**: Version 11 or higher
- **ASMC API**: Backend server running on port 7055

Local Development Setup

```
# Clone the repository
git clone <repository-url>
cd asmcdae-mobile

# Install dependencies
npm install

# For iOS (if needed)
cd ios && pod install && cd ..

# Start Metro bundler
npm run start:reset

# Run on Android
npm run android
```

```
# Run on iOS (if available)
npm run ios
```

Production Build

```
# Build Android APK
cd android
./gradlew assembleRelease

# Build Android AAB (for Play Store)
./gradlew bundleRelease
```

Architecture Overview

Technology Stack

- **Frontend Framework**: React Native 0.79.5
- **State Management**: React Query (TanStack Query) 5.0.0
- **Navigation**: React Navigation 7.x
- **HTTP Client**: Axios 1.10.0
- **Form Management**: Formik 2.4.6 + Yup 1.6.1
- **UI Components**: Custom components with React Native
- **Storage**: AsyncStorage 2.1.2
- **Icons**: React Native Vector Icons 10.2.0
- **Calendar**: React Native Calendars 1.1313.0
- **Image Handling**: React Native Image Picker 7.1.0
- **WebView**: React Native WebView 13.15.0

Project Structure

```
asmcdae-mobile/
├── android/                 # Android-specific files
│   ├── app/                # Android app configuration
│   ├── build.gradle        # Gradle build configuration
│   └── gradle.properties   # Gradle properties
├── assets/                 # Static assets
│   ├── icons/             # App icons and images
│   ├── font/              # Custom fonts
│   └── ic_launcher/       # App launcher icons
├── src/
│   ├── components/        # Reusable UI components
│   │   ├── common/        # Common/shared components
│   │   └── Home/          # Home-specific components
│   ├── containers/        # Screen containers
│   │   ├── Activities/    # Activity management
│   │   ├── Events/        # Event management
│   │   ├── Halls/         # Hall management
│   │   ├── Profile/       # User profile
│   │   └── ...            # Other screens
│   ├── contexts/          # React contexts
│   ├── helpers/           # Utility functions
│   ├── hooks/             # Custom React hooks
│   ├── lib/                # Third-party library configurations
│   ├── routes/             # Navigation configuration
│   └── styles/             # Global styles
├── App.js                  # Main app component
├── index.js                # App entry point
└── package.json            # Dependencies and scripts
```

Component Architecture

The mobile app follows a **Container-Component pattern** with React Query integration:

1. **Containers**: Screen-level components with business logic
2. **Components**: Reusable UI components for presentation
3. **Contexts**: Global state management with React Context
4. **Hooks**: Custom hooks for data fetching and state management

Installation & Setup

Environment Configuration

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

```
# API Configuration
API_BASE_URL=http://localhost:7055/api
API_TIMEOUT=30000

# App Configuration
APP_NAME=ASMC Mobile
APP_VERSION=1.0.0

# Authentication
JWT_STORAGE_KEY=asmc_mobile_token
REFRESH_TOKEN_KEY=asmc_mobile_refresh

# Feature Flags
ENABLE_DEBUG_MODE=true
ENABLE_LOGGING=true
```

Android Setup

1. **Install Android Studio**
2. **Configure Android SDK**
3. **Set up Android Emulator**
4. **Configure Environment Variables**:

```
# Add to ~/.bashrc or ~/.zshrc
export ANDROID_HOME=$HOME/Android/Sdk
export PATH=$PATH:$ANDROID_HOME/emulator
export PATH=$PATH:$ANDROID_HOME/tools
export PATH=$PATH:$ANDROID_HOME/tools/bin
export PATH=$PATH:$ANDROID_HOME/platform-tools
```

Development Scripts

```
# Start Metro bundler
npm start

# Start with cache reset
npm run start:reset

# Run on Android
npm run android

# Run on Android (debug mode)
npm run android:dev

# Debug mode with cache reset
npm run debug

# Run tests
npm test

# Lint code
npm run lint

# React DevTools
npm run devtools
```

Environment Configuration

API Configuration

Location: `/src/helpers/constants.js`

```
export const baseUrl = 'http://localhost:7055/api';
export const apiEndpoints = {
    auth: {
        login: '/auth/login',
        logout: '/auth/logout',
        refresh: '/auth/refresh',
        profile: '/auth/profile',
    },
    members: {
        list: '/members',
        profile: '/members/profile',
        update: '/members/update',
    },
    activities: {
        list: '/activities',
        detail: '/activities',
        enroll: '/activities/enroll',
    },
    events: {
        list: '/events',
        detail: '/events',
        register: '/events/register',
    },
    halls: {
        list: '/halls',
        detail: '/halls',
        book: '/halls/book',
    },
    payments: {
        history: '/payments/history',
        process: '/payments/process',
    },
};
```

Axios Configuration

Location: `/src/helpers/axios.js`

```
import axios from 'axios';
import AsyncStorage from '@react-native-async-storage/async-storage';
import { baseUrl } from './constants';

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

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

// Response interceptor
axiosInstance.interceptors.response.use(
    (response) => response,
    async (error) => {
        if (error.response?.status === 401) {
            // Handle token expiration
            await AsyncStorage.removeItem('authToken');
            await AsyncStorage.removeItem('refreshToken');
            // Navigate to login
        }
        return Promise.reject(error);
    },
);
```

```
export default axiosInstance;
```

App Structure

Navigation Structure

The app uses React Navigation with a tab-based navigation system:

Main Tabs

1. **Home** - Dashboard and quick access
2. **Events** - Event listings and registration
3. **Halls** - Hall bookings and information
4. **Activities** - Activity enrollment and management
5. **Profile** - User profile and settings

Stack Navigation

- **Public Stack**: Login, Forgot Password
- **Private Stack**: Main app screens with nested navigation

Screen Components

Core Screens

1. **HomeContainer** (`/src/containers/Home/`)
   - Dashboard with quick actions
   - Recent activities and events
   - Membership status display
2. **EventsContainer** (`/src/containers/Events/`)
   - Event listings with filtering
   - Event details and registration
   - Event booking management
3. **ActivitiesContainer** (`/src/containers/Activities/`)
   - Activity listings and categories
   - Activity enrollment
   - Activity schedules and details
4. **HallsContainer** (`/src/containers/Halls/`)
   - Hall listings and availability
   - Hall booking system
   - Booking history and management
5. **ProfileContainer** (`/src/containers/Profile/`)
   - User profile information
   - Settings and preferences
   - Membership details

Supporting Screens

- **LoginContainer** - User authentication
- **ForgotPasswordContainer** - Password recovery

- **EditProfileContainer** - Profile editing
- **ChangePasswordContainer** - Password change
- **PaymentHistoryContainer** - Payment records
- **BookingFormContainer** - Booking forms
- **WebViewContainer** - External web content

Common Components

Location: `/src/components/common/`

Header Component

```
import React from 'react';
import { View, Text, TouchableOpacity } from 'react-native';
import MaterialIcons from 'react-native-vector-icons/MaterialIcons';

const Header = ({ title, showBack, showLogout, showProfile, onBack, onLogout }) => {
    return (
        <View style={styles.header}>
            {showBack && (
                <TouchableOpacity onPress={onBack} style={styles.backButton}>
                    <MaterialIcons name="arrow-back" size={24} color="#000" />
                </TouchableOpacity>
            )}
            <Text style={styles.title}>{title}</Text>
            {showLogout && (
                <TouchableOpacity onPress={onLogout} style={styles.logoutButton}>
                    <MaterialIcons name="logout" size={24} color="#000" />
                </TouchableOpacity>
            )}
        </View>
    );
};
```

InputField Component

```
import React from 'react';
import { TextInput, View, Text } from 'react-native';

const InputField = ({
    label,
    value,
    onChangeText,
    placeholder,
    secureTextEntry,
    error,
    ...props
}) => {
    return (
        <View style={styles.container}>
            {label && <Text style={styles.label}>{label}</Text>}
            <TextInput
                style={[styles.input, error && styles.inputError]}
                value={value}
                onChangeText={onChangeText}
                placeholder={placeholder}
                secureTextEntry={secureTextEntry}
                {...props}
            />
            {error && <Text style={styles.errorText}>{error}</Text>}
        </View>
    );
};
```

State Management

React Query Integration

Location: `/src/lib/queryClient.js`

```
import { QueryClient } from '@tanstack/react-query';

export const queryClient = new QueryClient({
    defaultOptions: {
        queries: {
            staleTime: 5 * 60 * 1000, // 5 minutes
            cacheTime: 10 * 60 * 1000, // 10 minutes
            retry: 2,
            refetchOnWindowFocus: false,
        },
        mutations: {
            retry: 1,
        },
    },
});
```

Authentication Context

Location: `/src/contexts/AuthContext.js`

```
import React, { createContext, useContext, useReducer, useEffect } from 'react';
import AsyncStorage from '@react-native-async-storage/async-storage';

const AuthContext = createContext();

const initialState = {
    user: null,
    token: null,
    isAuth: false,
    isLoading: true,
};

const authReducer = (state, action) => {
    switch (action.type) {
        case 'LOGIN_SUCCESS':
            return {
                ...state,
                user: action.payload.user,
                token: action.payload.token,
                isAuth: true,
                isLoading: false,
            };
        case 'LOGOUT':
            return {
                ...state,
                user: null,
                token: null,
                isAuth: false,
                isLoading: false,
            };
        case 'SET_LOADING':
            return {
                ...state,
                isLoading: action.payload,
            };
        default:
            return state;
    }
};

export const AuthProvider = ({ children }) => {
    const [state, dispatch] = useReducer(authReducer, initialState);

    useEffect(() => {
        checkAuthStatus();
    }, []);

    const checkAuthStatus = async () => {
        try {
            const token = await AsyncStorage.getItem('authToken');
            const user = await AsyncStorage.getItem('userData');

            if (token && user) {
                dispatch({
                    type: 'LOGIN_SUCCESS',
                    payload: {
                        token,
                        user: JSON.parse(user),
                    },
                });
            } else {
                dispatch({ type: 'SET_LOADING', payload: false });
            }
        } catch (error) {
            dispatch({ type: 'SET_LOADING', payload: false });
        }
    };
    const login = async (credentials) => {
        try {
            dispatch({ type: 'SET_LOADING', payload: true });

            const response = await axiosInstance.post('/auth/login', credentials);
            const { token, user } = response.data;

            await AsyncStorage.setItem('authToken', token);
            await AsyncStorage.setItem('userData', JSON.stringify(user));

            dispatch({
                type: 'LOGIN_SUCCESS',
                payload: { token, user },
            });

            return { success: true };
        } catch (error) {
            dispatch({ type: 'SET_LOADING', payload: false });
            return { success: false, error: error.response?.data?.message };
        }
    };

    const logout = async () => {
        try {
            await AsyncStorage.removeItem('authToken');
            await AsyncStorage.removeItem('userData');
            dispatch({ type: 'LOGOUT' });
        } catch (error) {
            console.error('Logout error:', error);
        }
    };

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

export const useAuth = () => {
    const context = useContext(AuthContext);
    if (!context) {
        throw new Error('useAuth must be used within AuthProvider');
    }
    return context;
};
```

Custom Hooks

useActivity Hook

Location: `/src/hooks/useActivity.js`

```
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import axiosInstance from '../helpers/axios';

export const useActivities = () => {
    return useQuery({
        queryKey: ['activities'],
        queryFn: async () => {
            const response = await axiosInstance.get('/activities');
            return response.data;
        },
    });
};

export const useActivity = (id) => {
    return useQuery({
        queryKey: ['activity', id],
        queryFn: async () => {
            const response = await axiosInstance.get(`/activities/${id}`);
            return response.data;
        },
        enabled: !!id,
    });
};

export const useEnrollActivity = () => {
    const queryClient = useQueryClient();

    return useMutation({
        mutationFn: async (activityId) => {
            const response = await
axiosInstance.post(`/activities/${activityId}/enroll`);
            return response.data;
        },
        onSuccess: () => {
            queryClient.invalidateQueries(['activities']);
            queryClient.invalidateQueries(['enrolled-activities']);
        },
    });
};
```

API Integration

API Service Functions

Location: `/src/helpers/api.js`

```
import axiosInstance from './axios';

export const authAPI = {
    login: (credentials) => axiosInstance.post('/auth/login', credentials),
    logout: () => axiosInstance.post('/auth/logout'),
    refreshToken: (refreshToken) => axiosInstance.post('/auth/refresh', { refreshToken
}),
    getProfile: () => axiosInstance.get('/auth/profile'),
    updateProfile: (data) => axiosInstance.put('/auth/profile', data),
};

export const activitiesAPI = {
    getActivities: (params) => axiosInstance.get('/activities', { params }),
    getActivity: (id) => axiosInstance.get(`/activities/${id}`),
    enrollActivity: (id) => axiosInstance.post(`/activities/${id}/enroll`),
    getEnrolledActivities: () => axiosInstance.get('/activities/enrolled'),
};

export const eventsAPI = {
    getEvents: (params) => axiosInstance.get('/events', { params }),
    getEvent: (id) => axiosInstance.get(`/events/${id}`),
    registerEvent: (id) => axiosInstance.post(`/events/${id}/register`),
    getRegisteredEvents: () => axiosInstance.get('/events/registered'),
};

export const hallsAPI = {
    getHalls: (params) => axiosInstance.get('/halls', { params }),
    getHall: (id) => axiosInstance.get(`/halls/${id}`),
    bookHall: (data) => axiosInstance.post('/halls/book', data),
    getBookedHalls: () => axiosInstance.get('/halls/booked'),
};

export const paymentsAPI = {
    getPaymentHistory: () => axiosInstance.get('/payments/history'),
    processPayment: (data) => axiosInstance.post('/payments/process', data),
};
```

Error Handling

```
import Toast from 'react-native-toast-message';

export const handleAPIError = (error) => {
    const message = error.response?.data?.message || 'An error occurred';

    Toast.show({
        type: 'error',
        text1: 'Error',
        text2: message,
    });
};

export const handleAPISuccess = (message) => {
    Toast.show({
        type: 'success',
        text1: 'Success',
        text2: message,
    });
};
```

▯ UI Components

Style System

Location: `/src/styles/styles.js`

```
import { StyleSheet, Dimensions } from 'react-native';

const { width, height } = Dimensions.get('window');

export const colors = {
    primary: '#1976d2',
    secondary: '#dc004e',
    success: '#4caf50',
    warning: '#ff9800',
    error: '#f44336',
    background: '#fafafa',
    surface: '#ffffff',
    text: '#212121',
    textSecondary: '#757575',
    border: '#e0e0e0',
};

export const typography = {
    h1: {
        fontSize: 32,
        fontWeight: 'bold',
        fontFamily: 'PlusJakartaSans-Bold',
    },
    h2: {
        fontSize: 24,
        fontWeight: '600',
        fontFamily: 'PlusJakartaSans-SemiBold',
    },
    h3: {
        fontSize: 20,
        fontWeight: '600',
        fontFamily: 'PlusJakartaSans-SemiBold',
    },
    body: {
        fontSize: 16,
        fontFamily: 'PlusJakartaSans-Regular',
    },
    caption: {
        fontSize: 14,
        fontFamily: 'PlusJakartaSans-Regular',
    },
};

export const spacing = {
    xs: 4,
    sm: 8,
    md: 16,
    lg: 24,
    xl: 32,
    xxl: 48,
};

export const borderRadius = {
    sm: 4,
    md: 8,
    lg: 12,
    xl: 16,
    round: 50,
};

export default StyleSheet.create({
    container: {
        flex: 1,
        backgroundColor: colors.background,
    },
    loadingContainer: {
        flex: 1,
        justifyContent: 'center',
        alignItems: 'center',
    },
    header: {
        flexDirection: 'row',
        alignItems: 'center',
        justifyContent: 'space-between',
        paddingHorizontal: spacing.md,
        paddingVertical: spacing.sm,
        backgroundColor: colors.surface,
        elevation: 2,
        shadowColor: '#000',
        shadowOffset: { width: 0, height: 2 },
        shadowOpacity: 0.1,
        shadowRadius: 4,
    },
    card: {
        backgroundColor: colors.surface,
        borderRadius: borderRadius.md,
        padding: spacing.md,
        margin: spacing.sm,
        elevation: 2,
        shadowColor: '#000',
        shadowOffset: { width: 0, height: 2 },
        shadowOpacity: 0.1,
        shadowRadius: 4,
    },
    button: {
        backgroundColor: colors.primary,
        paddingHorizontal: spacing.lg,
        paddingVertical: spacing.md,
        borderRadius: borderRadius.md,
        alignItems: 'center',
    },
    buttonText: {
        color: colors.surface,
        fontSize: 16,
        fontWeight: '600',
        fontFamily: 'PlusJakartaSans-SemiBold',
    },
    input: {
        borderWidth: 1,
        borderColor: colors.border,
        borderRadius: borderRadius.md,
        paddingHorizontal: spacing.md,
        paddingVertical: spacing.sm,
        fontSize: 16,
        fontFamily: 'PlusJakartaSans-Regular',
        backgroundColor: colors.surface,
    },
    inputError: {
        borderColor: colors.error,
    },
    errorText: {
        color: colors.error,
        fontSize: 14,
        fontFamily: 'PlusJakartaSans-Regular',
        marginTop: spacing.xs,
    },
});
```

Custom Components

Card Component

```
import React from 'react';
import { View, Text, TouchableOpacity, Image } from 'react-native';
import styles, { colors, spacing } from '../../styles/styles';

const Card = ({ title, description, image, onPress, children }) => {
    return (
        <TouchableOpacity style={styles.card} onPress={onPress}>
            {image && <Image source={{ uri: image }} style={styles.cardImage} />}
            {title && <Text style={styles.cardTitle}>{title}</Text>}
            {description && <Text style={styles.cardDescription}>{description}</Text>}
            {children}
        </TouchableOpacity>
    );
};
```

Loading Component

```
import React from 'react';
import { View, ActivityIndicator, Text } from 'react-native';
import styles from '../../styles/styles';

const Loading = ({ message = 'Loading...', size = 'large' }) => {
    return (
        <View style={styles.loadingContainer}>
            <ActivityIndicator size={size} color="#1976d2" />
            <Text style={styles.loadingText}>{message}</Text>
        </View>
    );
};
```

Authentication & Authorization

Login Flow

```
import React, { useState } from 'react';
import { View, Text, Alert } from 'react-native';
import { useAuth } from '../contexts/AuthContext';
import { InputField } from '../components/common/InputField';
import { Button } from '../components/common/Button';

const LoginScreen = () => {
    const [credentials, setCredentials] = useState({
        email: '',
        password: '',
    });
    const [loading, setLoading] = useState(false);
    const { login } = useAuth();

    const handleLogin = async () => {
        setLoading(true);
        const result = await login(credentials);
        setLoading(false);

        if (!result.success) {
            Alert.alert('Login Failed', result.error);
        }
    };

    return (
        <View style={styles.container}>
            <Text style={styles.title}>Welcome to ASMC</Text>
            <InputField
                label="Email"
                value={credentials.email}
                onChangeText={(text) => setCredentials({ ...credentials, email: text
})}
                placeholder="Enter your email"
                keyboardType="email-address"
                autoCapitalize="none"
            />

            <InputField
                label="Password"
                value={credentials.password}
                onChangeText={(text) =>
                    setCredentials({ ...credentials, password: text })
                }
                placeholder="Enter your password"
                secureTextEntry
            />

            <Button
                title="Login"
                onPress={handleLogin}
                loading={loading}
                disabled={loading}
            />
        </View>
    );
};
```

Protected Routes

```
import React from 'react';
import { useAuth } from '../contexts/AuthContext';
import LoginScreen from '../containers/Login';

const ProtectedRoute = ({ children }) => {
    const { isAuth, isLoading } = useAuth();

    if (isLoading) {
        return <LoadingScreen />;
    }

    if (!isAuth) {
        return <LoginScreen />;
    }

    return children;
};
```

Features Overview

Core Features

1. **User Authentication**
    - Login/logout functionality
    - Password recovery
    - Profile management
2. **Activities Management**
    - Browse available activities
    - Enroll in activities
    - View enrolled activities
    - Activity schedules and details
3. **Events Management**
    - Browse upcoming events
    - Register for events
    - View registered events
    - Event details and information
4. **Hall Booking System**
    - Browse available halls
    - Book halls for specific dates/times
    - View booking history
    - Manage existing bookings
5. **Payment Integration**
    - View payment history
    - Process payments for activities/events
    - Membership fee payments
6. **Profile Management**
    - View and edit profile information
    - Change password
    - View membership details
    - Family member management

Advanced Features

- **Offline Support**: Cached data for offline viewing
- **Push Notifications**: Activity and event reminders
- **Image Upload**: Profile picture and document upload
- **Calendar Integration**: Activity and event scheduling
- **Search and Filtering**: Advanced search capabilities
- **Responsive Design**: Optimized for different screen sizes

Build & Deployment

Android Build

Debug Build

```
# Clean and build debug APK
cd android
./gradlew clean
./gradlew assembleDebug

# Install on device
adb install app/build/outputs/apk/debug/app-debug.apk
```

Release Build

```
# Generate signed APK
cd android
./gradlew assembleRelease

# Generate AAB for Play Store
./gradlew bundleRelease
```

Build Configuration

Android App Configuration

Location: `android/app/build.gradle`

```
android {
    compileSdkVersion 34
    buildToolsVersion "34.0.0"

    defaultConfig {
        applicationId "com.asmc.mobile"
        minSdkVersion 21
        targetSdkVersion 34
        versionCode 1
        versionName "1.0.0"
    }

    signingConfigs {
        release {
            storeFile file('release.keystore')
            storePassword 'your_store_password'
            keyAlias 'your_key_alias'
            keyPassword 'your_key_password'
        }
    }

    buildTypes {
        release {
            signingConfig signingConfigs.release
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-
rules.pro'
        }
    }
}
```

Deployment Checklist

Pre-deployment

- [ ] Update version numbers
- [ ] Test on multiple devices
- [ ] Verify all features work
- [ ] Check API endpoints
- [ ] Review security configurations

Post-deployment

- [ ] Monitor app performance
- [ ] Check crash reports
- [ ] Monitor user feedback
- [ ] Update documentation

Development Guidelines

Code Standards

1. **Component Structure**

   - Use functional components with hooks
   - Implement proper prop types
   - Follow consistent naming conventions

2. **File Organization**

   - Group related files in folders
   - Use descriptive file names
   - Maintain consistent import order

3. **Styling**

   - Use StyleSheet for performance
   - Follow design system guidelines
   - Implement responsive design

4. **State Management**

   - Use React Query for server state
   - Use React Context for global state
   - Minimize local state usage

Performance Best Practices

1. **Image Optimization**

   - Use appropriate image formats
   - Implement lazy loading
   - Optimize image sizes

2. **Memory Management**

   - Clean up subscriptions and timers

    - Use React.memo for expensive components
    - Implement proper navigation cleanup

3. **Network Optimization**
    - Implement request caching
    - Use pagination for large datasets
    - Optimize API calls

Testing Guidelines

```javascript
// Example test structure
import React from 'react';
import { render, fireEvent } from '@testing-library/react-native';
import LoginScreen from '../LoginScreen';

describe('LoginScreen', () => {
    it('renders login form correctly', () => {
        const { getByPlaceholderText, getByText } = render(<LoginScreen />);

        expect(getByPlaceholderText('Enter your email')).toBeTruthy();
        expect(getByPlaceholderText('Enter your password')).toBeTruthy();
        expect(getByText('Login')).toBeTruthy();
    });

    it('handles login submission', () => {
        const { getByPlaceholderText, getByText } = render(<LoginScreen />);

        fireEvent.changeText(
            getByPlaceholderText('Enter your email'),
            'test@example.com',
        );
        fireEvent.changeText(getByPlaceholderText('Enter your password'),
'password123');
        fireEvent.press(getByText('Login'));

        // Add assertions for login behavior
    });
});
```

Troubleshooting

Common Issues

1. Metro Bundler Issues

Problem: Metro bundler not starting or cache issues

Solution:

```bash
# Clear Metro cache
npm run start:reset

# Clear npm cache
```

```
npm cache clean --force

# Delete node_modules and reinstall
rm -rf node_modules package-lock.json
npm install
```

2. Android Build Issues

Problem: Gradle build failures

Solution:

```
# Clean Gradle cache
cd android
./gradlew clean

# Update Gradle wrapper
./gradlew wrapper --gradle-version=8.0

# Check Java version
java -version
```

3. API Connection Issues

Problem: Cannot connect to backend API

Solution:

- Check API base URL in constants
- Verify backend server is running
- Check network connectivity
- Review CORS configuration

4. Navigation Issues

Problem: Navigation not working properly

Solution:

- Check React Navigation version compatibility
- Verify navigation structure
- Check for missing dependencies
- Review navigation state management

Debug Tools

React Native Debugger

```
# Install React Native Debugger
npm install -g react-native-debugger

# Start debugger
npm run devtools
```

Flipper Integration

```
# Install Flipper
# Download from https://fbflipper.com/

# Configure in android/app/src/debug/java/
# Add Flipper initialization
```

Logging

```
import { Platform } from 'react-native';

const logger = {
    log: (message, data) => {
        if (__DEV__) {
            console.log(`[${new Date().toISOString()}] ${message}`, data);
        }
    },
    error: (message, error) => {
        if (__DEV__) {
            console.error(`[${new Date().toISOString()}] ${message}`, error);
        }
    },
};
```

Summary

The ASMC Mobile App provides a comprehensive mobile solution for sports management with:

- **Modern React Native Architecture** with React Query for state management
- **Comprehensive Feature Set** including activities, events, hall bookings, and payments
- **Robust Authentication** with JWT token management
- **Optimized Performance** with caching and offline support
- **Production-Ready Build** system for Android deployment
- **Extensive Documentation** for development and maintenance

This mobile application ensures members have easy access to all ASMC services and features on their mobile devices.

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