StandAlonePlayer Component Documentation

Table of Contents

Overview

The **StandAlonePlayer** is a unified React Native component that can play multiple types of educational content in a single, reusable screen. It supports various MIME types including videos, PDFs, interactive content, assessments, and more. This component eliminates the need for separate player screens for each content type, providing a streamlined solution for content playback.

Key Features

- **Multi-format Support**: Handles 10+ different content MIME types
- **Offline & Online Playback**: Supports both online streaming and offline downloaded content
- **Automatic Content Detection**: Automatically selects the appropriate player based on MIME type
- **Telemetry Tracking**: Built-in tracking for content consumption analytics
- **Assessment Support**: Integrated assessment/quiz player with result tracking
- **Android 15 Compatibility**: Includes safe area handling for modern Android devices
- **Orientation Management**: Automatic landscape/portrait locking based on content type
- **YouTube Integration**: Native YouTube player support

Supported MIME Types

The component supports the following content types:

1. Sunbird Content Player

- application/vnd.ekstep.ecml-archive - ECML interactive content
- application/vnd.ekstep.html-archive - HTML5 games/content
- application/vnd.ekstep.h5p-archive - H5P interactive content
- video/x-youtube - YouTube videos

2. Sunbird PDF Player

- application/pdf - PDF documents

3. Sunbird EPUB Player

- application/epub - EPUB e-books

4. Sunbird Video Player

- video/mp4 - MP4 video files
- video/webm - WebM video files

- audio/mp3 - MP3 audio files
- audio/wav - WAV audio files

5. Sunbird QuML Player

- application/vnd.sunbird.questionset - Question sets/assessments

---

Dependencies

Required NPM Packages

```
{
  "react": "18.2.0",
  "react-native": "0.74.2",
  "@react-navigation/native": "^6.1.17",
  "@react-navigation/native-stack": "^6.9.26",
  "react-native-webview": "^13.12.5",
  "react-native-youtube-iframe": "^2.4.1",
  "react-native-fs": "^2.20.0",
  "react-native-zip-archive": "^7.0.0",
  "react-native-orientation-locker": "^1.7.0",
  "react-native-safe-area-context": "^4.10.5",
  "react-native-config": "^1.5.2",
  "@react-native-async-storage/async-storage": "^1.24.0"
}
```

Additional Dependencies

- react-native-safe-area-context - For safe area insets
- @react-navigation/native - For navigation
- react-native-config - For environment configuration

Installation Steps

Step 1: Install NPM Dependencies

```
npm install react-native-webview
npm install react-native-youtube-iframe
npm install react-native-fs
npm install react-native-zip-archive
npm install react-native-orientation-locker
npm install react-native-safe-area-context
npm install react-native-config
npm install @react-navigation/native
npm install @react-navigation/native-stack
npm install @react-native-async-storage/async-storage
```

Step 2: Install Native Dependencies (iOS)

```
cd ios
pod install
cd ..
```

Step 3: Android Permissions

Add the following permissions to android/app/src/main/AndroidManifest.xml:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Step 4: Copy Player Libraries

The component requires Sunbird player libraries to be placed in the native assets:

For Android:

- Copy player libraries to: android/app/src/main/assets/libs/

- Required folders:
    - sunbird-content-player/
    - sunbird-pdf-player/
    - sunbird-epub-player/
    - sunbird-video-player/
    - sunbird-quml-player/

For iOS:

- Copy player libraries to: ios/learnerapp/assets/assets/libs/
- Same folder structure as Android

Step 5: Copy Component Files

Copy the following files to your project:

```
src/screens/PlayerScreen/StandAlonePlayer/
├── StandAlonePlayer.js
└── data.js
```

Setup Instructions

Step 1: Add to Navigation Stack

In your navigation file (e.g., src/Routes/Public/StackScreen.js):

```
import StandAlonePlayer from
'../../screens/PlayerScreen/StandAlonePlayer/StandAlonePlayer';

// Inside your Stack.Navigator
<Stack.Screen
  name="StandAlonePlayer"
  component={StandAlonePlayer}
  options={{
    headerShown: false,
    lazy: true,
  }}
/>;
```

Step 2: Create Required Utility Functions

The component requires several utility functions. Ensure these exist in your project:

File: src/utils/API/ApiCalls.js

- readContent(content_do_id) - Fetches content metadata
- hierarchyContent(content_do_id) - Fetches question set hierarchy
- listQuestion(url, identifiers) - Fetches questions for assessments
- assessmentTracking(...) - Saves assessment results
- telemetryTracking(telemetryArray) - Saves telemetry events
- contentTracking(...) - Saves content consumption tracking

File: src/utils/Helper/JSHelper.js

- getData(key, type) - Retrieves data from storage
- storeData(key, value, type) - Stores data in storage
- removeData(key) - Removes data from storage

File: src/utils/JsHelper/Helper.js

- getDataFromStorage(key) - Gets data from AsyncStorage
- findObjectByIdentifier(array, identifier) - Finds object by identifier
- logEventFunction(eventObject) - Logs analytics events

File: src/utils/API/AuthService.js

- storeAsessmentOffline(userId, identifier, payload) - Stores assessment offline
- storeTelemetryOffline(userId, telemetryArray) - Stores telemetry offline
- storeTrackingOffline(...) - Stores content tracking offline

Step 3: Environment Configuration

Create a .env file with required configuration:

```
QUESTION_LIST_URL=https://your-api-url/list/questions
```

Step 4: Required Components

Ensure these components exist in your project:

- src/components/SafeAreaWrapper/SafeAreaWrapper.js
- src/components/GlobalText/GlobalText.js
- src/screens/Assessment/TestResultModal.js
- src/components/MimeAletModal/MimeAlertModal.js

Step 5: Language Context Setup

The component uses a translation context. Ensure you have:

```
// src/context/LanguageContext.js
import { useTranslation } from '../../../context/LanguageContext';
```

Usage Examples

Example 1: Basic Content Playback

```
import { useNavigation } from '@react-navigation/native';

const MyComponent = () => {
  const navigation = useNavigation();

  const playContent = () => {
    navigation.navigate('StandAlonePlayer', {
      content_do_id: 'content-123',
      content_mime_type: 'application/pdf',
      title: 'My PDF Document',
      isOffline: false,
      course_id: 'course-456',
      unit_id: 'unit-789',
    });
  };

  return <Button title="Play Content" onPress={playContent} />;
};
```

Example 2: Playing YouTube Video

```
navigation.navigate('StandAlonePlayer', {
  content_do_id: 'youtube-content-123',
  content_mime_type: 'video/x-youtube',
  title: 'Educational Video',
  isOffline: false,
  course_id: 'course-456',
  unit_id: 'unit-789',
});
```

Example 3: Playing Assessment/Quiz

```
navigation.navigate('StandAlonePlayer', {
  content_do_id: 'assessment-123',
  content_mime_type: 'application/vnd.sunbird.questionset',
  title: 'Math Quiz',
  isOffline: false,
  course_id: 'course-456',
  unit_id: 'unit-789',
});
```

Example 4: Playing Offline Content

```
navigation.navigate('StandAlonePlayer', {
  content_do_id: 'content-123',
  content_mime_type: 'video/mp4',
  title: 'Offline Video',
  isOffline: true,
  course_id: 'course-456',
  unit_id: 'unit-789',
});
```

Example 5: From Content Card (Real Implementation)

```
// From src/screens/Dashboard/ContentCard.js
const handlePress = (data) => {
  navigation.push('StandAlonePlayer', {
    content_do_id: data?.identifier || data?.id,
    content_mime_type: data?.mimeType || data?.app,
    isOffline: false,
    course_id: course_id,
    unit_id: unit_id,
  });
};
```

Example 6: From Deep Link

```
// From src/utils/JsHelper/DeepLink.js
navigation.push('StandAlonePlayer', {
  content_do_id: identifier,
  content_mime_type: content_response?.result?.content?.mimeType,
  isOffline: false,
  course_id: identifier,
  unit_id: identifier,
});
```

Route Parameters

The component accepts the following route parameters:

Parameter	Type	Required	Description
content_do_id	string	Yes	Unique identifier for the content
content_mime_type	string	Yes	MIME type of the content (see supported types)
title	string	No	Display title for the content
isOffline	boolean	No	Whether content is available offline (default: false)
course_id	string	No	Course identifier for tracking
unit_id	string	No	Unit identifier for tracking

Parameter Details

content_do_id

- Unique identifier for the content item
- Used to fetch content metadata and files
- Example: "do_1234567890"

content_mime_type

- Must be one of the supported MIME types listed above

- Determines which player library to use
- Example: "application/pdf"

title

- Optional display title
- Used in modals and result screens
- Example: "Introduction to Mathematics"

isOffline

- Boolean flag indicating offline availability
- If true, component looks for locally stored content first
- Default: false

course_id

- Optional course identifier
- Used for content tracking and analytics
- Example: "course_123"

unit_id

- Optional unit/module identifier
- Used for content tracking and analytics
- Example: "unit_456"

Component Features

1. Automatic Content Detection

The component automatically detects the content type based on content_mime_type and loads the appropriate player.

2. Offline Content Support

- Automatically checks for offline content

- Downloads content if not available locally
- Supports offline playback for all content types

3. Content Download

- Automatic download for ECML, HTML, H5P, and YouTube content
- Progress tracking during download
- Automatic extraction of ZIP files

4. Telemetry Tracking

- Tracks START, INTERACT, and END events
- Stores telemetry offline if network unavailable
- Syncs telemetry when network available

5. Assessment Results

- Displays assessment results in modal
- Tracks scores and time spent
- Stores results offline if needed

6. Orientation Management

- Automatically locks to landscape for video/interactive content
- Locks to portrait for PDFs and assessments
- Handles orientation changes gracefully

7. Safe Area Handling

- Android 15 compatible safe area insets
- Handles notches and navigation bars
- Landscape mode padding for side navigation

8. YouTube Integration

- Native YouTube player using react-native-youtube-iframe

- Extracts video ID from various YouTube URL formats
- Tracks video start and end events

9. Error Handling

- Displays error modals for unsupported content
- Handles network errors gracefully
- Shows download buttons when content unavailable

10. Back Button Handling

- Custom back button handler
- Saves tracking data before navigation
- Prevents accidental app exit

File Structure

```
src/screens/PlayerScreen/StandAlonePlayer/
├── StandAlonePlayer.js    # Main component file
└── data.js                # Player configuration objects

Required Supporting Files:
src/utils/API/
├── ApiCalls.js            # API functions
└── AuthService.js         # Offline storage functions

src/utils/Helper/
└── JSHelper.js            # Storage helper functions

src/utils/JsHelper/
└── Helper.js              # Utility functions

src/components/
├── SafeAreaWrapper/       # Safe area component
├── GlobalText/            # Text component
```

```
└── MimeAletModal/          # Error modal

src/screens/Assessment/
└── TestResultModal.js      # Assessment result modal
```

Configuration

Player Configuration Objects

The component uses configuration objects defined in data.js:

1. **qumlPlayerConfig** - For question sets/assessments
2. **contentPlayerConfig** - For ECML, HTML, H5P content
3. **pdfPlayerConfig** - For PDF documents
4. **videoPlayerConfig** - For video/audio files
5. **epubPlayerConfig** - For EPUB e-books

Customizing Player Config

Edit src/screens/PlayerScreen/StandAlonePlayer/data.js to customize:

- Side menu options (share, download, exit, print, replay)
- End page display
- User data
- Context information

Environment Variables

Required environment variables (in .env):

```
QUESTION_LIST_URL=https://api.example.com/list/questions
```

Player Libraries Setup

Android Setup

1. **Create assets directory:**
   `android/app/src/main/assets/libs/`
2. **Copy player libraries:**
   - Download Sunbird player libraries
   - Extract to android/app/src/main/assets/libs/
   - Structure should be:
     ```
     libs/
     ├── sunbird-content-player/
     │   └── index.html
     ├── sunbird-pdf-player/
     │   └── index.html
     ├── sunbird-epub-player/
     │   └── index.html
     ├── sunbird-video-player/
     │   └── index.html
     └── sunbird-quml-player/
         └── index.html
     ```
3. **Update build.gradle (if needed):**
   ```
   android {
       ...
       packagingOptions {
           pickFirst 'lib/x86/libc++_shared.so'
           pickFirst 'lib/x86_64/libc++_shared.so'
           pickFirst 'lib/armeabi-v7a/libc++_shared.so'
           pickFirst 'lib/arm64-v8a/libc++_shared.so'
       }
   }
   ```

iOS Setup

1. **Create assets directory:**
   `ios/learnerapp/assets/assets/libs/`
2. **Copy player libraries:**

    - Same structure as Android
    - Add to Xcode project
    - Ensure "Create folder references" is selected
3. **Update Info.plist:**

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

API Requirements

Required API Endpoints

1. **Content Read API**
    - Endpoint: GET /content/v3/read/{contentId}
    - Returns: Content metadata with artifactUrl, downloadUrl, mimeType
2. **Hierarchy Content API**
    - Endpoint: GET /content/v3/hierarchy/{contentId}
    - Returns: Question set hierarchy with childNodes
3. **List Questions API**
    - Endpoint: POST /list/questions
    - Body: Array of question identifiers
    - Returns: Question objects
4. **Assessment Tracking API**
    - Endpoint: POST /assessment/track
    - Body: Score details, max score, time spent
    - Returns: Assessment result
5. **Telemetry Tracking API**
    - Endpoint: POST /telemetry
    - Body: Array of telemetry events
    - Returns: Success status
6. **Content Tracking API**

    - Endpoint: POST /content/track
    - Body: User ID, course ID, content ID, tracking details
    - Returns: Success status

API Response Format

Content Read Response:

```
{
  "result": {
    "content": {
      "identifier": "do_123",
      "mimeType": "application/pdf",
      "artifactUrl": "https://...",
      "downloadUrl": "https://...",
      "name": "Content Title"
    }
  }
}
```

Question Set Response:

```
{
  "result": {
    "questionset": {
      "identifier": "do_123",
      "mimeType": "application/vnd.sunbird.questionset",
      "childNodes": ["q1", "q2", "q3"],
      "outcomeDeclaration": {...}
    }
  }
}
```

Troubleshooting

Issue 1: Player Not Loading

Symptoms: Blank screen or error message

Solutions:

1. Verify player libraries are in correct location
2. Check file paths in htmlFilePath variable
3. Ensure WebView permissions are granted
4. Check console for JavaScript errors

Issue 2: Content Not Downloading

Symptoms: Download button appears but nothing happens

Solutions:

1. Check network connectivity
2. Verify API endpoints are correct
3. Check storage permissions
4. Verify downloadUrl in content metadata

Issue 3: YouTube Videos Not Playing

Symptoms: YouTube player shows error

Solutions:

1. Verify YouTube video ID extraction
2. Check internet connectivity
3. Verify react-native-youtube-iframe is properly installed
4. Check YouTube API key (if required)

Issue 4: Assessment Results Not Saving

Symptoms: Results modal doesn't appear or data not saved

Solutions:

1. Check assessmentTracking API endpoint
2. Verify user ID is set
3. Check network connectivity
4. Verify offline storage functions

Issue 5: Orientation Not Locking

Symptoms: Screen rotates when it shouldn't

Solutions:

1. Verify react-native-orientation-locker is installed
2. Check orientation lock code in useEffect
3. Ensure proper cleanup in useEffect return

Issue 6: Safe Area Issues on Android 15

Symptoms: Content hidden behind navigation bars

Solutions:

1. Verify react-native-safe-area-context is installed
2. Check safe area insets calculation
3. Verify padding values in styles

Issue 7: Telemetry Not Syncing

Symptoms: Telemetry events not appearing in analytics

Solutions:

1. Check telemetryTracking API endpoint
2. Verify telemetry object structure
3. Check offline storage functions

4. Verify sync on app resume

Common Debugging Steps

1. **Enable Console Logging:**
   - Uncomment console.log statements in component
   - Check React Native debugger
2. **Check Storage:**
   ```
   // In component
   const checkStorage = async () => {
     const data = await getData('contentId', '');
     console.log('Stored content:', data);
   };
   ```
3. **Verify Route Params:**
   ```
   console.log('Route params:', route.params);
   ```
4. **Check WebView Messages:**
   - Monitor handleMessage function
   - Log all incoming messages

Best Practices

1. **Always Provide Required Params:**
   - Ensure content_do_id and content_mime_type are always provided
   - Validate MIME type before navigation
2. **Handle Loading States:**
   - Show loading indicators while content loads
   - Handle download progress for large files
3. **Error Handling:**
   - Always handle network errors
   - Provide fallback UI for unsupported content
4. **Memory Management:**
   - Clean up event listeners on unmount

    - Release orientation locks when done
5. **Offline Support:**
    - Check offline availability before navigation
    - Provide download option for offline content
6. **Tracking:**
    - Ensure user ID is set before playing content
    - Track content consumption for analytics

Version Information

- **Component Version:** 1.0.0
- **React Native Version:** 0.74.2
- **Last Updated:** 2024
- **Compatibility:** Android 5.0+, iOS 11.0+

Support & Contact

For issues, questions, or contributions:

- Check the troubleshooting section above
- Review the code comments in StandAlonePlayer.js
- Refer to Sunbird player documentation for player-specific issues

---

License

This component is part of the Pratham Learning App project. Please refer to the main project license for usage terms.

Appendix: Complete Navigation Example

```
// Complete example with error handling
import React from 'react';
import { View, Button, Alert } from 'react-native';
import { useNavigation } from '@react-navigation/native';
import { useInternet } from '../context/NetworkContext';

const ContentPlayerExample = ({ content }) => {
  const navigation = useNavigation();
  const { isConnected } = useInternet();

  const playContent = async () => {
    try {
      // Validate content
      if (!content?.identifier || !content?.mimeType) {
        Alert.alert('Error', 'Invalid content data');
        return;
      }

      // Check offline availability
      const isOffline = await checkOfflineContent(content.identifier);

      // Navigate to player
      navigation.navigate('StandAlonePlayer', {
        content_do_id: content.identifier,
        content_mime_type: content.mimeType,
        title: content.name || 'Content',
        isOffline: isOffline,
        course_id: content.courseId || '',
        unit_id: content.unitId || '',
      });
    } catch (error) {
      console.error('Error playing content:', error);
      Alert.alert('Error', 'Failed to play content');
    }
  };

  return (
    <View>
      <Button title="Play Content" onPress={playContent} />
    </View>
  );
};

export default ContentPlayerExample;
```

End of Documentation