Contents

SSO (Single Sign-On) Setup Guide

Overview

The MagicWork app now supports Single Sign-On (SSO) with Google and Apple. Users can sign in using their Google or Apple accounts instead of creating a new account with email/password.

Implementation Details

1. OAuth Providers

- **Google OAuth** - Sign in with Google account
- **Apple OAuth** - Sign in with Apple ID

2. UI Components

- OAuth buttons added to Login screen (`/login`)
- OAuth buttons added to Sign Up screen (`/signup`)
- Buttons styled to match app design
- Loading states during OAuth flow

3. Authentication Flow

1. User taps "Continue with Google" or "Continue with Apple"
2. App opens browser/WebView for OAuth authentication
3. User authenticates with provider (Google/Apple)
4. Provider redirects back to app via deep link: `magicwork://auth-callback`
5. `AuthStateListener` detects successful authentication
6. User is automatically navigated to feed screen

4. Deep Link Configuration

iOS (Info.plist):

```
<key>CFBundleURLTypes</key>
<array>
    <dict>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>magicwork</string>
        </array>
    </dict>
</array>
```

Android (AndroidManifest.xml):

```
<intent-filter>
    <action android:name="android.intent.action.VIEW"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <category android:name="android.intent.category.BROWSABLE"/>
    <data
        android:scheme="magicwork"
        android:host="auth-callback"/>
</intent-filter>
```

Supabase Dashboard Configuration

Step 1: Enable OAuth Providers

1. Go to Supabase Dashboard
2. Select your project
3. Navigate to **Authentication** → **Providers**
4. Enable **Google** and **Apple** providers

Step 2: Configure Google OAuth

1. **Get Google OAuth Credentials:**
   - Go to Google Cloud Console
   - Create a new project or select existing
   - Enable Google+ API
   - Go to **Credentials** → **Create Credentials** → **OAuth 2.0 Client ID**
   - Application type: **Web application**
   - Authorized redirect URIs:
     `https://ejhafhggndirnxmwrtgm.supabase.co/auth/v1/callback`
   - Copy **Client ID** and **Client Secret**
2. **Configure in Supabase:**
   - In Supabase Dashboard → Authentication → Providers → Google
   - Paste **Client ID** and **Client Secret**
   - Click **Save**

Step 3: Configure Apple OAuth

1. **Get Apple OAuth Credentials:**
   - Go to Apple Developer Portal
   - Navigate to **Certificates, Identifiers & Profiles**
   - Create a **Services ID** (if not exists)
   - Enable **Sign in with Apple**
   - Configure **Return URLs**:
     `https://ejhafhggndirnxmwrtgm.supabase.co/auth/v1/callback`
   - Create a **Key** for Sign in with Apple
   - Download the key file (.p8)
   - Note the **Key ID** and **Team ID**
2. **Configure in Supabase:**
   - In Supabase Dashboard → Authentication → Providers → Apple
   - Enter **Services ID** (Client ID)
   - Enter **Team ID**
   - Enter **Key ID**
   - Upload **Key File** (.p8)
   - Click **Save**

Step 4: Configure Redirect URLs

In Supabase Dashboard → Authentication → URL Configuration:

Site URL:

```
magicwork://auth-callback
```

Redirect URLs (add these):

```
magicwork://auth-callback
https://ejhafhggndirnxmwrtgm.supabase.co/auth/v1/callback
```

Testing OAuth

Test Google Sign-In:

1. Open app → Navigate to Login or Sign Up
2. Tap "Continue with Google"
3. Browser should open with Google sign-in
4. Sign in with Google account
5. Should redirect back to app
6. Should navigate to feed screen automatically

Test Apple Sign-In:

1. Open app → Navigate to Login or Sign Up
2. Tap "Continue with Apple"
3. Browser should open with Apple sign-in
4. Sign in with Apple ID
5. Should redirect back to app
6. Should navigate to feed screen automatically

Troubleshooting:

OAuth not working: - Check Supabase Dashboard → Authentication → Providers are enabled - Verify redirect URLs are configured correctly - Check deep link configuration in Info.plist (iOS) and AndroidManifest.xml (Android) - Verify OAuth credentials are correct in Supabase

Deep link not working: - iOS: Check Info.plist has `CFBundleURLSchemes` with `magicwork` - Android: Check AndroidManifest.xml has intent-filter with `magicwork://auth-callback` - Test deep link: `magicwork://auth-callback` should open the app

Session not persisting: - Check Supabase session is being stored - Verify `AuthStateListener` is listening for auth state changes - Check `AuthService.initialize()` is called on app start

---

Files Modified

Created:

- `lib/widgets/auth_state_listener.dart` - Listens for OAuth callbacks

Modified:

- `lib/services/auth_service.dart` - Added `signInWithOAuth()` method
- `lib/providers/auth_provider.dart` - Added `signInWithGoogle()` and `signInWithApple()` methods
- `lib/screens/login_screen.dart` - Added OAuth buttons
- `lib/screens/signup_screen.dart` - Added OAuth buttons
- `lib/main.dart` - Added `AuthStateListener` widget
- `ios/Runner/Info.plist` - Added deep link configuration
- `android/app/src/main/AndroidManifest.xml` - Added deep link configuration

Security Notes

1. **OAuth Credentials:**
    - Never commit OAuth credentials to version control
    - Store credentials securely in Supabase Dashboard
    - Rotate credentials periodically
2. **Deep Links:**
    - Deep links are handled securely by Supabase
    - OAuth tokens are exchanged server-side
    - No sensitive data in deep link URLs
3. **Session Management:**
    - Sessions are managed by Supabase
    - Tokens are stored securely by Supabase SDK
    - Sessions automatically refresh when needed

Next Steps

1. **Configure OAuth in Supabase Dashboard** (see steps above)
2. **Test OAuth flows** on both iOS and Android
3. **Handle edge cases:**
    - User cancels OAuth flow
    - Network errors during OAuth
    - OAuth provider errors
4. **Optional Enhancements:**
    - Add more OAuth providers (Facebook, Twitter, etc.)
    - Customize OAuth button styling
    - Add OAuth account linking (link Google/Apple to existing email account)

OpenAI API Key Setup

The MagicWork app uses OpenAI's API for generating personalized meditation scripts, images, and content. You need to obtain an OpenAI API key and configure it in the app.

Step 1: Get OpenAI API Key

1. **Create OpenAI Account:**
    - Go to OpenAI Platform
    - Click **Sign Up** or **Log In**
    - Complete account registration (requires email verification)
2. **Add Payment Method:**
    - OpenAI requires a payment method to use the API
    - Go to **Settings** → **Billing**
    - Add a credit card or payment method
    - Note: OpenAI charges based on API usage (pay-as-you-go)
3. **Generate API Key:**

- Go to API Keys page
- Click **Create new secret key**
- Give it a name (e.g., "MagicWork App")
- Click **Create secret key**
- **IMPORTANT:** Copy the key immediately - you won't be able to see it again!
- The key will look like: `sk-proj-...` or `sk-...`

Step 2: Configure API Key in the App

The app supports multiple ways to configure the OpenAI API key:

Option A: Through App Settings (Recommended for Testing)

1. **Launch the app**
2. **Navigate to Settings/Account** (if available in UI)
3. **Enter API Key** in the settings screen
4. The key will be stored securely in SharedPreferences

Option B: Programmatically (For Development) The app uses `AppConfig` to manage the API key. You can set it programmatically:

```
import 'package:magicwork/config/app_config.dart';

// Set OpenAI API key
await AppConfig.setOpenAIApiKey('your-api-key-here');
```

Option C: Default Key (Development Only) For development, the app has a default API key configured in `lib/config/app_config.dart`:

```
return kDebugMode
    ? 'sk-proj-HPWdrXpryVctZdNJ-jflnCCExDB0CkM8_ejtuXOTLRPG-yi0rL88whmHGQt5M4ZiCsYsNkEotHT3Blb
    : null;
```

WARNING: Never commit API keys to version control! Use environment variables or secure storage in production.

Step 3: Verify API Key Works

1. **Test AI Features:**
   - Start a meditation practice
   - The app should generate personalized scripts using OpenAI
   - Check console logs for any API errors
2. **Check API Usage:**
   - Go to OpenAI Usage Dashboard
   - Monitor your API usage and costs
   - Set up usage limits if needed

Step 4: Secure Storage (Production)

For production apps, consider:

1. **Environment Variables:**
    - Store API key in environment variables
    - Never hardcode in source code
    - Use secure key management services
2. **Backend Proxy:**
    - Create a backend API that proxies OpenAI requests
    - Store API key on server only
    - App calls your backend, which calls OpenAI
3. **Key Rotation:**
    - Regularly rotate API keys
    - Revoke old keys when creating new ones
    - Monitor for unauthorized usage

OpenAI API Costs

Current Pricing (as of 2024): - **GPT-4o:** ~$2.50-$10 per 1M input tokens, ~$10-$30 per 1M output tokens - **DALL-E 3:** $0.040 per image (1024x1024), $0.080 per image (1024x1792) - **Text-to-Speech:** $15 per 1M characters

Cost Optimization Tips: - Use GPT-3.5-turbo for simple tasks (cheaper) - Cache generated content when possible - Set usage limits in OpenAI dashboard - Monitor usage regularly

Troubleshooting

API Key Not Working: - Verify key is correct (no extra spaces) - Check key hasn't been revoked - Ensure payment method is active - Check API usage limits in OpenAI dashboard

Rate Limit Errors: - OpenAI has rate limits based on your tier - Free tier: 3 requests/minute - Paid tier: Higher limits - Implement retry logic with exponential backoff

Authentication Errors: - Verify API key format (starts with `sk-`) - Check key hasn't expired - Ensure key has proper permissions

---

Status: [OK] Implementation Complete

OAuth SSO is fully implemented and ready for testing. Configure OAuth providers in Supabase Dashboard to enable Google and Apple sign-in.

OpenAI API key setup is documented above. Configure your API key to enable AI-powered features in the app.