Authentication

Author: Josh S. Sakweli, Backend Lead Team
Last Updated: 2025-01-05
Version: v1.0

Base URL: `https://api.fursahub.com/api/v1`

Short Description: Authentication endpoints for Fursa Hub platform. Handles user registration/login via Firebase, token management, and session control. All new users start the onboarding flow after successful authentication.

Hints:

- Firebase handles the actual sign-in (Google, Apple, Email) - your app gets a Firebase ID token
- Send that Firebase token to our backend to get Fursa Hub access tokens
- Access tokens expire in 1 hour, use refresh token to get new ones
- Pass `preferredLanguage` and `theme` during first authentication to set user preferences

---

Authentication Flow

```
┌─────────────────────────────────────────────────────────────────────┐
│                        AUTHENTICATION FLOW                          │
├─────────────────────────────────────────────────────────────────────┤
│                                                                     │
│  1. USER OPENS APP                                                  │
│     └── App shows language selector (calls GET /languages)          │
│     └── User picks language (e.g., "sw" for Swahili)                │
│     └── App stores language + theme preference locally              │
│                                                                     │
│  2. USER SIGNS IN VIA FIREBASE                                      │
│     └── Google Sign-In / Apple Sign-In / Email+Password             │
│     └── Firebase returns ID Token                                   │
│                                                                     │
│  3. APP SENDS TO FURSA HUB BACKEND                                  │
│     └── POST /auth/firebase/authenticate                            │
│     └── Include: firebaseToken, preferredLanguage, theme            │
```

```
│                                                                    │
│  4. BACKEND RESPONSE                                               │
│     ├── NEW USER: Creates account, returns tokens + onboarding status  │
│     └── EXISTING USER: Returns tokens + current onboarding status      │
│                                                                    │
│  5. CHECK ONBOARDING STATUS                                        │
│     └── onboarding.isComplete = false → Navigate to onboarding flow    │
│     └── onboarding.isComplete = true → Navigate to home screen         │
│                                                                    │
│  6. TOKEN MANAGEMENT                                               │
│     └── Store accessToken (for API calls)                          │
│     └── Store refreshToken (for renewing accessToken)              │
│     └── When accessToken expires → POST /auth/refresh              │
│                                                                    │
└────────────────────────────────────────────────────────────────────┘
```

Standard Response Format

Success Response Structure

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Operation completed successfully",
  "action_time": "2025-01-05T10:30:45",
  "data": { }
}
```

Error Response Structure

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Error description",
  "action_time": "2025-01-05T10:30:45",
  "data": "Error description"
}
```

Endpoints

1. Authenticate with Firebase

Purpose: Exchange Firebase ID token for Fursa Hub access tokens. Creates new user if first time.

Endpoint: `POST` `{base_url}/auth/firebase/authenticate`

Access Level: ◫ Public

Authentication: None (Firebase token in body)

Request Headers:

Header	Type	Required	Description
Content-Type	string	Yes	`application/json`

Request JSON Sample:

```
{
  "firebaseToken": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
  "preferredLanguage": "sw",
  "theme": "DARK",
  "deviceInfo": "Android 14, Samsung Galaxy S24"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
firebaseToken	string	Yes	Firebase ID token from client SDK	Must be valid Firebase token
preferredLanguage	string	No	User's language preference	2-5 chars (e.g., "en", "sw", "fr")
theme	string	No	UI theme preference	enum: `LIGHT`, `DARK`, `SYSTEM`
deviceInfo	string	No	Device information for session tracking	Max 255 chars

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Authentication successful",
  "action_time": "2025-01-05T10:30:45",
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "tokenType": "Bearer",
    "expiresIn": 3600,
    "user": {
      "id": "550e8400-e29b-41d4-a716-446655440000",
      "email": "user@example.com",
      "username": "johndoe",
      "phoneNumber": null,
      "fullName": "John Doe",
      "profilePhotoUrl": "https://lh3.googleusercontent.com/...",
      "isPhoneVerified": false,
      "isEmailVerified": true,
      "preferredLanguage": "sw",
      "theme": "DARK",
      "authProvider": "GOOGLE",
      "role": "ROLE_USER",
      "createdAt": "2025-01-05T10:30:45"
    },
    "onboarding": {
      "isComplete": false,
      "currentStep": "PENDING_PHONE_VERIFICATION"
    }
  }
}
```


Success Response Fields:

Field	Description
accessToken	JWT token for API requests (expires in 1 hour)
refreshToken	Token to get new accessToken (expires in 30 days)
tokenType	Always "Bearer"
expiresIn	Access token lifetime in seconds

Field	Description
user	User profile information
user.theme	User's theme preference: LIGHT, DARK, or SYSTEM
onboarding.isComplete	`false` = must complete onboarding, `true` = can access app
onboarding.currentStep	Current onboarding step (see Onboarding docs)

---

2. Refresh Access Token

Purpose: Get a new access token using refresh token when current one expires.

Endpoint: POST `{base_url}/auth/refresh`

Access Level: Public

Authentication: None (refresh token in body)

Request JSON Sample:

```
{
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
refreshToken	string	Yes	Refresh token from authentication	Must be valid, non-expired

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Token refreshed successfully",
  "action_time": "2025-01-05T11:30:45",
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "tokenType": "Bearer",
```

```
    "expiresIn": 3600,
    "user": { ... },
    "onboarding": { ... }
  }
}
```

Error Responses:

Invalid Refresh Token (401):

```
{
  "success": false,
  "httpStatus": "UNAUTHORIZED",
  "message": "Invalid refresh token",
  "action_time": "2025-01-05T11:30:45",
  "data": "Invalid refresh token"
}
```

Expired Refresh Token (401):

```
{
  "success": false,
  "httpStatus": "UNAUTHORIZED",
  "message": "Refresh token expired",
  "action_time": "2025-01-05T11:30:45",
  "data": "Refresh token expired"
}
```

---

3. Logout

Purpose: Invalidate all refresh tokens for the user, ending all sessions.

Endpoint: `POST` `{base_url}/auth/logout`

Access Level: □□ Protected

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description

Authorization	string	Yes	`Bearer {accessToken}`

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Logged out successfully",
  "action_time": "2025-01-05T12:00:00",
  "data": null
}
```

---

4. Get Supported Languages

Purpose: Get list of supported languages for language selector screen.

Endpoint: `GET` `{base_url}/languages`

Access Level: □□ Public

Authentication: None

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Languages retrieved successfully",
  "action_time": "2025-01-05T10:00:00",
  "data": [
    {
      "code": "en",
      "name": "English",
      "nativeName": "English"
    },
    {
      "code": "sw",
      "name": "Swahili",
      "nativeName": "Kiswahili"
    },
```

```
    {
      "code": "fr",
      "name": "French",
      "nativeName": "Français"
    },
    {
      "code": "zh",
      "name": "Chinese",
      "nativeName": "□□"
    }
  ]
}
```

Frontend Implementation Guide

Step 1: First App Launch

```
Show language selector screen
├── Call GET /languages to get options
├── User selects language
├── Store locally: selectedLanguage, theme (default: SYSTEM)
└── Navigate to sign-in screen
```

Step 2: Sign In

```
Firebase Sign-In
├── Use Firebase SDK (Google/Apple/Email)
├── On success, get Firebase ID token
└── Call POST /auth/firebase/authenticate with:
    - firebaseToken
    - preferredLanguage (from step 1)
    - theme (from step 1)
    - deviceInfo (optional)
```

Step 3: Handle Response

```
Check response.data.onboarding.isComplete
├── false → Navigate to onboarding flow
│   └── Start at response.data.onboarding.currentStep
└── true → Navigate to home screen
```

Step 4: Store Tokens

```
Save securely:
├── accessToken → For Authorization header
├── refreshToken → For token renewal
└── user data → For UI display
```

Step 5: API Calls

```
All protected endpoints:
├── Add header: Authorization: Bearer {accessToken}
├── On 401 error → Try refresh token
│   ├── Success → Retry original request
│   └── Fail → Force re-login
└── Continue normal flow
```

Revision #2
Created 2 January 2026 07:56:35 by Admin Qbit
Updated 5 January 2026 16:15:22 by Admin Qbit