# Arab AI - Authentication API Documentation

## Base URLs

### Production Environment

https://arabai-825528766846.europe-west1.run.app

### Development Environment

http://localhost:5000

**Note:** Replace the base URL in all examples below with the appropriate environment URL.

## Response Format

All API responses follow this standardized format:

### Success Response

{  
 "success": true,  
 "message": "Description of the operation",  
 "data": {  
 // Response data (when applicable)  
 }  
}

### Error Response

{  
 "success": false,  
 "message": "Error description",  
 "errors": [  
 // Validation errors array (when applicable)  
 ]  
}

## 🔓 Public Endpoints

### 1. Health Check

**Endpoint:** GET /health  
**Description:** Check if the server is running and healthy  
**Authentication:** None required

**Example Request:**

curl -X GET http://localhost:5000/health

**Example Response:**

{  
 "success": true,  
 "message": "Server is running",  
 "timestamp": "2025-09-05T00:06:06.911Z"  
}

### 2. API Information

**Endpoint:** GET /api  
**Description:** Get API version and available endpoints  
**Authentication:** None required

**Example Request:**

curl -X GET http://localhost:5000/api

**Example Response:**

{  
 "success": true,  
 "message": "Arab AI API",  
 "version": "1.0.0",  
 "endpoints": {  
 "auth": "/api/auth",  
 "health": "/health"  
 }  
}

### 3. User Signup

**Endpoint:** POST /api/auth/signup  
**Description:** Register a new user account  
**Authentication:** None required  
**Rate Limit:** 5 requests per 15 minutes

**Request Headers:**

Content-Type: application/json

**Request Body:**

{  
 "name": "John Doe",  
 "email": "john@example.com",  
 "password": "Password123"  
}

**Field Validation:** - name: 2-50 characters, letters and spaces only - email: Valid email format, max 100 characters, must be unique - password: 6-128 characters, must contain lowercase, uppercase, and number

**Example Request:**

curl -X POST http://localhost:5000/api/auth/signup \  
 -H "Content-Type: application/json" \  
 -d '{  
 "name": "John Doe",  
 "email": "john@example.com",  
 "password": "Password123"  
 }'

**Success Response (201):**

{  
 "success": true,  
 "message": "User created successfully",  
 "data": {  
 "user": {  
 "id": 1,  
 "name": "John Doe",  
 "email": "john@example.com",  
 "role": "user",  
 "is\_email\_verified": false,  
 "created\_at": "2025-09-05T00:06:06.911Z",  
 "updated\_at": "2025-09-05T00:06:06.911Z"  
 },  
 "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",  
 "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."  
 }  
}

**Error Responses:** - **400 Bad Request:** Validation errors - **409 Conflict:** Email already exists - **429 Too Many Requests:** Rate limit exceeded - **500 Internal Server Error:** Server error

### 4. User Login

**Endpoint:** POST /api/auth/login  
**Description:** Authenticate user and get access tokens  
**Authentication:** None required  
**Rate Limit:** 5 requests per 15 minutes

**Request Headers:**

Content-Type: application/json

**Request Body:**

{  
 "email": "john@example.com",  
 "password": "Password123"  
}

**Field Validation:** - email: Valid email format - password: Required

**Example Request:**

curl -X POST http://localhost:5000/api/auth/login \  
 -H "Content-Type: application/json" \  
 -d '{  
 "email": "john@example.com",  
 "password": "Password123"  
 }'

**Success Response (200):**

{  
 "success": true,  
 "message": "Login successful",  
 "data": {  
 "user": {  
 "id": 1,  
 "name": "John Doe",  
 "email": "john@example.com",  
 "role": "user",  
 "is\_email\_verified": false,  
 "created\_at": "2025-09-05T00:06:06.911Z",  
 "updated\_at": "2025-09-05T00:06:06.911Z"  
 },  
 "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",  
 "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."  
 }  
}

**Error Responses:** - **400 Bad Request:** Validation errors - **401 Unauthorized:** Invalid email or password - **429 Too Many Requests:** Rate limit exceeded - **500 Internal Server Error:** Server error

### 5. User Logout

**Endpoint:** POST /api/auth/logout  
**Description:** Logout user (client-side token removal)  
**Authentication:** None required

**Example Request:**

curl -X POST http://localhost:5000/api/auth/logout

**Success Response (200):**

{  
 "success": true,  
 "message": "Logout successful"  
}

## 🔒 Protected Endpoints

**Authentication Required:** All protected endpoints require a valid JWT token in the Authorization header.

**Authorization Header Format:**

Authorization: Bearer YOUR\_JWT\_TOKEN\_HERE

### 6. Get User Profile

**Endpoint:** GET /api/auth/profile  
**Description:** Retrieve current user’s profile information  
**Authentication:** Required

**Request Headers:**

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

**Example Request:**

curl -X GET http://localhost:5000/api/auth/profile \  
 -H "Authorization: Bearer YOUR\_JWT\_TOKEN\_HERE"

**Success Response (200):**

{  
 "success": true,  
 "message": "Profile retrieved successfully",  
 "data": {  
 "user": {  
 "id": 1,  
 "name": "John Doe",  
 "email": "john@example.com",  
 "role": "user",  
 "is\_email\_verified": false,  
 "created\_at": "2025-09-05T00:06:06.911Z",  
 "updated\_at": "2025-09-05T00:06:06.911Z"  
 }  
 }  
}

**Error Responses:** - **401 Unauthorized:** Missing or invalid token - **404 Not Found:** User not found - **500 Internal Server Error:** Server error

### 7. Update User Profile

**Endpoint:** PUT /api/auth/profile  
**Description:** Update current user’s profile information  
**Authentication:** Required

**Request Headers:**

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...  
Content-Type: application/json

**Request Body:**

{  
 "name": "John Smith"  
}

**Field Validation:** - name: Optional, 2-50 characters, letters and spaces only

**Example Request:**

curl -X PUT http://localhost:5000/api/auth/profile \  
 -H "Authorization: Bearer YOUR\_JWT\_TOKEN\_HERE" \  
 -H "Content-Type: application/json" \  
 -d '{  
 "name": "John Smith"  
 }'

**Success Response (200):**

{  
 "success": true,  
 "message": "Profile updated successfully",  
 "data": {  
 "user": {  
 "id": 1,  
 "name": "John Smith",  
 "email": "john@example.com",  
 "role": "user",  
 "is\_email\_verified": false,  
 "created\_at": "2025-09-05T00:06:06.911Z",  
 "updated\_at": "2025-09-05T00:06:10.123Z"  
 }  
 }  
}

**Error Responses:** - **400 Bad Request:** Validation errors - **401 Unauthorized:** Missing or invalid token - **500 Internal Server Error:** Server error

### 8. Change Password

**Endpoint:** PUT /api/auth/change-password  
**Description:** Change current user’s password  
**Authentication:** Required

**Request Headers:**

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...  
Content-Type: application/json

**Request Body:**

{  
 "currentPassword": "Password123",  
 "newPassword": "NewPassword456",  
 "confirmPassword": "NewPassword456"  
}

**Field Validation:** - currentPassword: Required, must match user’s current password - newPassword: 6-128 characters, must contain lowercase, uppercase, and number - confirmPassword: Must match newPassword

**Example Request:**

curl -X PUT http://localhost:5000/api/auth/change-password \  
 -H "Authorization: Bearer YOUR\_JWT\_TOKEN\_HERE" \  
 -H "Content-Type: application/json" \  
 -d '{  
 "currentPassword": "Password123",  
 "newPassword": "NewPassword456",  
 "confirmPassword": "NewPassword456"  
 }'

**Success Response (200):**

{  
 "success": true,  
 "message": "Password changed successfully"  
}

**Error Responses:** - **400 Bad Request:** Validation errors - **401 Unauthorized:** Missing/invalid token or incorrect current password - **404 Not Found:** User not found - **500 Internal Server Error:** Server error

## 🔒 Security Features

### Rate Limiting

* **Authentication endpoints** (/api/auth/signup, /api/auth/login): 5 requests per 15 minutes
* **Global rate limit**: 100 requests per 15 minutes per IP

### Password Security

* Passwords are hashed using bcryptjs with 12 salt rounds
* Password requirements: 6-128 characters, must contain lowercase, uppercase, and number

### JWT Tokens

* **Access Token**: Expires in 7 days (configurable via JWT\_EXPIRES\_IN)
* **Refresh Token**: Expires in 30 days (configurable via JWT\_REFRESH\_EXPIRES\_IN)
* Tokens are signed with JWT\_SECRET and JWT\_REFRESH\_SECRET

### Input Validation

* All input is validated using express-validator
* Email normalization and sanitization
* XSS protection through input escaping

### CORS & Security Headers

* CORS configured for specified frontend origins
* Helmet.js provides security headers
* Request size limits (10MB)

## 📊 HTTP Status Codes

| Code | Description |
| --- | --- |
| 200 | OK - Request successful |
| 201 | Created - Resource created successfully |
| 400 | Bad Request - Invalid input or validation error |
| 401 | Unauthorized - Authentication required or failed |
| 403 | Forbidden - Insufficient permissions |
| 404 | Not Found - Resource not found |
| 409 | Conflict - Resource already exists |
| 413 | Payload Too Large - Request body too large |
| 429 | Too Many Requests - Rate limit exceeded |
| 500 | Internal Server Error - Server error |

## 🧪 Testing with Postman

### Complete Testing Flow

**Use Production URL:** https://arabai-825528766846.europe-west1.run.app  
**Use Development URL:** http://localhost:5000

1. **Health Check**

* # Production  
  GET https://arabai-825528766846.europe-west1.run.app/health  
    
  # Development   
  GET http://localhost:5000/health

1. **Create User**

* # Production  
  POST https://arabai-825528766846.europe-west1.run.app/api/auth/signup  
    
  # Development  
  POST http://localhost:5000/api/auth/signup  
    
  Body: {  
   "name": "Test User",  
   "email": "test@example.com",  
   "password": "Test123456"  
  }

1. **Login**

* # Production  
  POST https://arabai-825528766846.europe-west1.run.app/api/auth/login  
    
  # Development  
  POST http://localhost:5000/api/auth/login  
    
  Body: {  
   "email": "test@example.com",  
   "password": "Test123456"  
  }
* *Save the token from response for next requests*

1. **Get Profile**

* # Production  
  GET https://arabai-825528766846.europe-west1.run.app/api/auth/profile  
    
  # Development  
  GET http://localhost:5000/api/auth/profile  
    
  Headers: Authorization: Bearer YOUR\_TOKEN\_HERE

1. **Update Profile**

* # Production  
  PUT https://arabai-825528766846.europe-west1.run.app/api/auth/profile  
    
  # Development  
  PUT http://localhost:5000/api/auth/profile  
    
  Headers: Authorization: Bearer YOUR\_TOKEN\_HERE  
  Body: { "name": "Updated Test User" }

1. **Change Password**

* # Production  
  PUT https://arabai-825528766846.europe-west1.run.app/api/auth/change-password  
    
  # Development  
  PUT http://localhost:5000/api/auth/change-password  
    
  Headers: Authorization: Bearer YOUR\_TOKEN\_HERE  
  Body: {  
   "currentPassword": "Test123456",  
   "newPassword": "NewPassword123",  
   "confirmPassword": "NewPassword123"  
  }

1. **Logout**

* # Production  
  POST https://arabai-825528766846.europe-west1.run.app/api/auth/logout  
    
  # Development  
  POST http://localhost:5000/api/auth/logout

## 🔧 Environment Variables

Required environment variables for the API:

# Database Configuration  
DATABASE\_URL=postgres://username:password@localhost:5432/arabai\_db  
  
# JWT Configuration  
JWT\_SECRET=your\_super\_secret\_jwt\_key\_here  
JWT\_EXPIRES\_IN=7d  
JWT\_REFRESH\_SECRET=your\_super\_secret\_refresh\_jwt\_key\_here  
JWT\_REFRESH\_EXPIRES\_IN=30d  
  
# App Configuration  
NODE\_ENV=development  
PORT=5000  
  
# Security  
BCRYPT\_ROUNDS=12

## 📝 Notes

* All timestamps are in ISO 8601 format (UTC)
* Passwords are never returned in API responses
* Email addresses are stored in lowercase
* User roles are either ‘user’ or ‘admin’
* The is\_email\_verified field is for future email verification functionality