Authentication API Documentation

Welcome to the Authentication API! This guide will help you integrate our user authentication system into your JavaScript or Python applications. We've built this API to handle all the common authentication needs - user registration, login, email verification, password resets, and phone verification via OTP.

Getting Started

Our API provides two separate user types (User1 and User2) with identical functionality. Think of them as different user pools for your application - maybe one for customers and another for admins, or whatever fits your use case.

Base URL: `http://your-domain.com`

Authentication Flow

All protected endpoints require a JWT token in the Authorization header:

```
Authorization: Bearer your_jwt_token_here
```

User Registration

Create a New User

Endpoint: `POST /User1/create` or `POST /User2/create`

Register a new user in your system. The API will automatically send welcome and verification emails.

Request Body

```
{
  "first_name": "John",
  "last_name": "Doe",
  "email": "john.doe@example.com",
  "phone": "+1234567890",
  "country_code": "US",
  "password": "securePassword123"
}
```

JavaScript Example

```javascript
const registerUser = async (userData) => {
  try {
    const response = await fetch('http://your-domain.com/User1/create', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(userData)
    });

    const result = await response.json();
    console.log('User created:', result);
    return result;
  } catch (error) {
    console.error('Registration failed:', error);
  }
};

// Usage
const newUser = {
  first_name: "John",
  last_name: "Doe",
  email: "john.doe@example.com",
  phone: "+1234567890",
  country_code: "US",
  password: "securePassword123"
};
```

```
registerUser(newUser);
```

Python Example

```
import requests
import json

def register_user(user_data):
    url = "http://your-domain.com/User1/create"
    headers = {"Content-Type": "application/json"}

    try:
        response = requests.post(url, headers=headers, json=user_data)
        response.raise_for_status()
        return response.json()
    except requests.exceptions.RequestException as e:
        print(f"Registration failed: {e}")
        return None

# Usage
new_user = {
    "first_name": "John",
    "last_name": "Doe",
    "email": "john.doe@example.com",
    "phone": "+1234567890",
    "country_code": "US",
    "password": "securePassword123"
}

result = register_user(new_user)
print(result)
```

Response

```
{
  "message": "User created successfully",
  "Email": "john.doe@example.com",
  "id": "generated_user_id"
}
```

User Authentication

Login

Endpoint: `POST /User1/login` or `POST /User2/login`

Authenticate a user and receive a JWT token for accessing protected endpoints.

Request Body (Form Data)

- `username`: User's email address
- `password`: User's password

JavaScript Example

```javascript
const loginUser = async (email, password) => {
  const formData = new FormData();
  formData.append('username', email);
  formData.append('password', password);

  try {
    const response = await fetch('http://your-domain.com/User1/login', {
      method: 'POST',
      body: formData
    });

    const result = await response.json();

    if (result.access_token) {
      // Store token for future requests
      localStorage.setItem('authToken', result.access_token);
      console.log('Login successful');
    }

    return result;
  } catch (error) {
    console.error('Login failed:', error);
  }
};

// Usage
loginUser('john.doe@example.com', 'securePassword123');
```

Python Example

```
import requests

def login_user(email, password):
    url = "http://your-domain.com/User1/login"
    data = {
        'username': email,
        'password': password
    }

    try:
        response = requests.post(url, data=data)
        response.raise_for_status()
        result = response.json()

        if 'access_token' in result:
            # Store token for future requests
            token = result['access_token']
            print("Login successful")
            return token
    except requests.exceptions.RequestException as e:
        print(f"Login failed: {e}")
        return None

# Usage
token = login_user('john.doe@example.com', 'securePassword123')
```

Response

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "bearer"
}
```

User Profile Management

Get User Profile

Endpoint: `GET /User1/?id=user_id` or `GET /User2/?id=user_id`

Retrieve user information by ID. Requires authentication.

JavaScript Example

```
const getUserProfile = async (userId) ⇒ {
  const token = localStorage.getItem('authToken');

  try {
    const response = await fetch(`http://your-domain.com/User1/?id=${userId}
`, {
      method: 'GET',
      headers: {
        'Authorization': `Bearer ${token}`,
        'Content-Type': 'application/json'
      }
    });
    const userProfile = await response.json();
    console.log('User profile:', userProfile);
    return userProfile;
  } catch (error) {
    console.error('Failed to fetch user profile:', error);
  }
};
```

Python Example

```
def get_user_profile(user_id, token):
    url = f"http://your-domain.com/User1/?id={user_id}"
    headers = {
        'Authorization': f'Bearer {token}',
        'Content-Type': 'application/json'
    }

    try:
        response = requests.get(url, headers=headers)
        response.raise_for_status()
        return response.json()
    except requests.exceptions.RequestException as e:
        print(f"Failed to fetch profile: {e}")
        return None
```

Delete User Account

Endpoint: `DELETE /User1/delete?id=user_id&email=user_email` or `DELETE /User2/delete?id=user_id&email=user_email`

Permanently delete a user account. Requires authentication.

JavaScript Example

```
const deleteUser = async (userId, email) => {
  const token = localStorage.getItem('authToken');

  try {
    const response = await fetch(`http://your-domain.com/User1/delete?id=${u
serId}&email=${email}`, {
      method: 'DELETE',
      headers: {
        'Authorization': `Bearer ${token}`,
        'Content-Type': 'application/json'
      }
    });

    const result = await response.json();
    console.log('Account deleted:', result);
    return result;
  } catch (error) {
    console.error('Failed to delete account:', error);
  }
};
```

Email Verification

Verify Email Address

Endpoint: `GET /User1/verify?token=verification_token` or `GET /User2/verify?token=verification_token`

Verify a user's email address using the token sent during registration.

This endpoint is typically accessed via email link, but you can also call it programmatically:

JavaScript Example

```
const verifyEmail = async (token) => {
  try {
    const response = await fetch(`http://your-domain.com/User1/verify?token=
${token}`, {
      method: 'GET'
    });

    // This returns an HTML page, but you can check the response status
    if (response.ok) {
      console.log('Email verified successfully');
      return true;
    }
  } catch (error) {
    console.error('Email verification failed:', error);
    return false;
  }
};
```

Password Management

Request Password Reset

Endpoint: `GET /User1/forget_password?email=user_email` or `GET /User2/forget_password?email=user_email`

Send a password reset link to the user's email address.

JavaScript Example

```
const requestPasswordReset = async (email) => {
  try {
    const response = await fetch(`http://your-domain.com/User1/forget_passw
ord?email=${email}`, {
      method: 'GET'
    });

    const result = await response.json();
    console.log('Password reset link sent:', result);
    return result;
  } catch (error) {
    console.error('Failed to send reset link:', error);
  }
};

// Usage
requestPasswordReset('john.doe@example.com');
```

Python Example

```
def request_password_reset(email):
    url = f"http://your-domain.com/User1/forget_password?email={email}"

    try:
        response = requests.get(url)
        response.raise_for_status()
        return response.json()
    except requests.exceptions.RequestException as e:
        print(f"Failed to send reset link: {e}")
        return None
```

Reset Password

Endpoint: `POST /User1/reset_password?token=reset_token` or `POST /User2/reset_password?token=reset_token`

Reset user's password using the token from the reset email.

Request Body (Form Data)

- `password`: New password

JavaScript Example

```
const resetPassword = async (token, newPassword) => {
  const formData = new FormData();
  formData.append('password', newPassword);
  try {
    const response = await fetch(`http://your-domain.com/User1/reset_passwor
d?token=${token}`, {
      method: 'POST',
      body: formData
    });

    // This returns an HTML page indicating success/failure
    if (response.ok) {
      console.log('Password reset successful');
      return true;
    }
  } catch (error) {
    console.error('Password reset failed:', error);
    return false;
  }
};
```

Phone Verification (OTP)

Generate OTP

Endpoint: `POST /User1/otp_phone?id=user_id&phone=phone_number` or `POST /User2/otp_phone?id=user_id&phone=phone_number`

Generate and send an OTP to the user's phone number.

JavaScript Example

```
const generateOTP = async (userId, phoneNumber) => {
  try {
    const response = await fetch(`http://your-domain.com/User1/otp_phone?id
=${userId}&phone=${phoneNumber}`, {
      method: 'POST'
    });
    const result = await response.json();
    console.log('OTP generated:', result);

    // Store the token for verification
    const otpToken = result.otp;
    return otpToken;
  } catch (error) {
    console.error('OTP generation failed:', error);
  }
};

// Usage
const otpToken = await generateOTP('user123', '+1234567890');
```

Python Example

```
def generate_otp(user_id, phone_number):
    url = f"http://your-domain.com/User1/otp_phone?id={user_id}&phone={pho
ne_number}"

    try:
        response = requests.post(url)
        response.raise_for_status()
        result = response.json()

        # Return the OTP token for verification
        return result.get('otp')
    except requests.exceptions.RequestException as e:
        print(f"OTP generation failed: {e}")
        return None
```

Verify OTP

Endpoint: `GET /User1/verify_otp_phone?token=otp_token&otp=123456` or `GET /User2/verify_otp_phone?token=otp_token&otp=123456`

Verify the OTP entered by the user.

JavaScript Example

```
const verifyOTP = async (otpToken, userEnteredOTP) => {
  try {
    const response = await fetch(`http://your-domain.com/User1/verify_otp_pho
ne?token=${otpToken}&otp=${userEnteredOTP}`, {
      method: 'GET'
    });

    const result = await response.json();

    if (result.message === "OTP verified successfully") {
      console.log('Phone number verified!');
      return true;
    } else {
      console.log('Invalid OTP');
      return false;
    }
  } catch (error) {
    console.error('OTP verification failed:', error);
    return false;
  }
};

// Usage
const isValid = await verifyOTP(otpToken, '123456');
```

Python Example

```
def verify_otp(otp_token, user_entered_otp):
    url = f"http://your-domain.com/User1/verify_otp_phone?token={otp_token}&
otp={user_entered_otp}"
```

```
    try:
        response = requests.get(url)
        response.raise_for_status()
        result = response.json()

        return result.get('message') == "OTP verified successfully"
    except requests.exceptions.RequestException as e:
        print(f"OTP verification failed: {e}")
        return False
```

Session Management

Logout

Endpoint: `GET /User1/logout` or `GET /User2/logout`

Invalidate the current authentication token. Requires authentication.

JavaScript Example

```javascript
const logoutUser = async () => {
  const token = localStorage.getItem('authToken');

  try {
    const response = await fetch('http://your-domain.com/User1/logout', {
      method: 'GET',
      headers: {
        'Authorization': `Bearer ${token}`,
        'Content-Type': 'application/json'
      }
    });

    const result = await response.json();

    if (response.ok) {
      // Clear stored token
      localStorage.removeItem('authToken');
      console.log('Logged out successfully');
    }

    return result;
  } catch (error) {
    console.error('Logout failed:', error);
  }
};
```

Python Example

```
def logout_user(token):
    url = "http://your-domain.com/User1/logout"
    headers = {
        'Authorization': f'Bearer {token}',
        'Content-Type': 'application/json'
    }

    try:
        response = requests.get(url, headers=headers)
        response.raise_for_status()
        result = response.json()

        if response.status_code == 200:
            print("Logged out successfully")
            return True

    except requests.exceptions.RequestException as e:
        print(f"Logout failed: {e}")
        return False
```

Error Handling

The API returns standard HTTP status codes. Here are the common ones you'll encounter:

- **200**: Success
- **400**: Bad Request (invalid data)
- **401**: Unauthorized (invalid/missing token)
- **403**: Forbidden (email not verified)
- **404**: Not Found (user doesn't exist)
- **422**: Validation Error (check your request format)

Example Error Response

```
{
  "detail": "Invalid email or password"
}
```

Complete Integration Example

Here's a complete example showing how to integrate the authentication flow:

JavaScript Integration

```
class AuthAPI {
  constructor(baseURL) {
    this.baseURL = baseURL;
    this.token = localStorage.getItem('authToken');
  }

  async register(userData) {
    const response = await fetch(`${this.baseURL}/User1/create`, {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(userData)
    });
    return response.json();
  }

  async login(email, password) {
    const formData = new FormData();
    formData.append('username', email);
    formData.append('password', password);

    const response = await fetch(`${this.baseURL}/User1/login`, {
      method: 'POST',
      body: formData
    });

    const result = await response.json();
    if (result.access_token) {
      this.token = result.access_token;
      localStorage.setItem('authToken', this.token);
    }
    return result;
  }

  async getProfile(userId) {
    const response = await fetch(`${this.baseURL}/User1/?id=${userId}`, {
      headers: { 'Authorization': `Bearer ${this.token}` }
    });
    return response.json();
  }

  async logout() {
    const response = await fetch(`${this.baseURL}/User1/logout`, {
      headers: { 'Authorization': `Bearer ${this.token}` }
    });

    if (response.ok) {
      this.token = null;
      localStorage.removeItem('authToken');
    }
    return response.json();
  }
}

// Usage
const auth = new AuthAPI('http://your-domain.com');
```

Python Integration

```
import requests

class AuthAPI:
    def __init__(self, base_url):
        self.base_url = base_url
        self.token = None

    def register(self, user_data):
        url = f"{self.base_url}/User1/create"
        response = requests.post(url, json=user_data)
        return response.json()

    def login(self, email, password):
        url = f"{self.base_url}/User1/login"
        data = {'username': email, 'password': password}
        response = requests.post(url, data=data)
        result = response.json()

        if 'access_token' in result:
            self.token = result['access_token']

        return result

    def get_profile(self, user_id):
        url = f"{self.base_url}/User1/?id={user_id}"
        headers = {'Authorization': f'Bearer {self.token}'}
        response = requests.get(url, headers=headers)
        return response.json()

    def logout(self):
        url = f"{self.base_url}/User1/logout"
        headers = {'Authorization': f'Bearer {self.token}'}
        response = requests.get(url, headers=headers)

        if response.ok:
            self.token = None

        return response.json()

# Usage
auth = AuthAPI('http://your-domain.com')
```