NextGate Authentication V2 (DEPRECATED)

Author: Josh Backend Team
Last Updated: 2025-01-24
Version: v1.0

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

Short Description: NextGate Authentication API provides a comprehensive, passwordless-first authentication system with OAuth2 support (Google, Apple), multi-channel OTP verification, device tracking, session management, and a 6-step user onboarding flow. Designed for mobile-first applications with security features including risk assessment, rate limiting, and refresh token rotation.

Hints:

- □□ **Passwordless by Default**: Users can authenticate via OTP (SMS/Email) without ever setting a password
- □□ **Device Trust**: Each device is tracked and can be managed by the user
- □□ **Token Rotation**: Refresh tokens are rotated on each use for enhanced security
- □ **OTP Validity**: All OTP codes expire in 10 minutes with max 3 verification attempts
- □□ **Multi-Channel**: OTP can be sent via Email or SMS based on user preference
- □□ **Metadata System**: Onboarding responses include `nextStepMetadata` for pre-filling forms

---

Authentication Architecture Overview

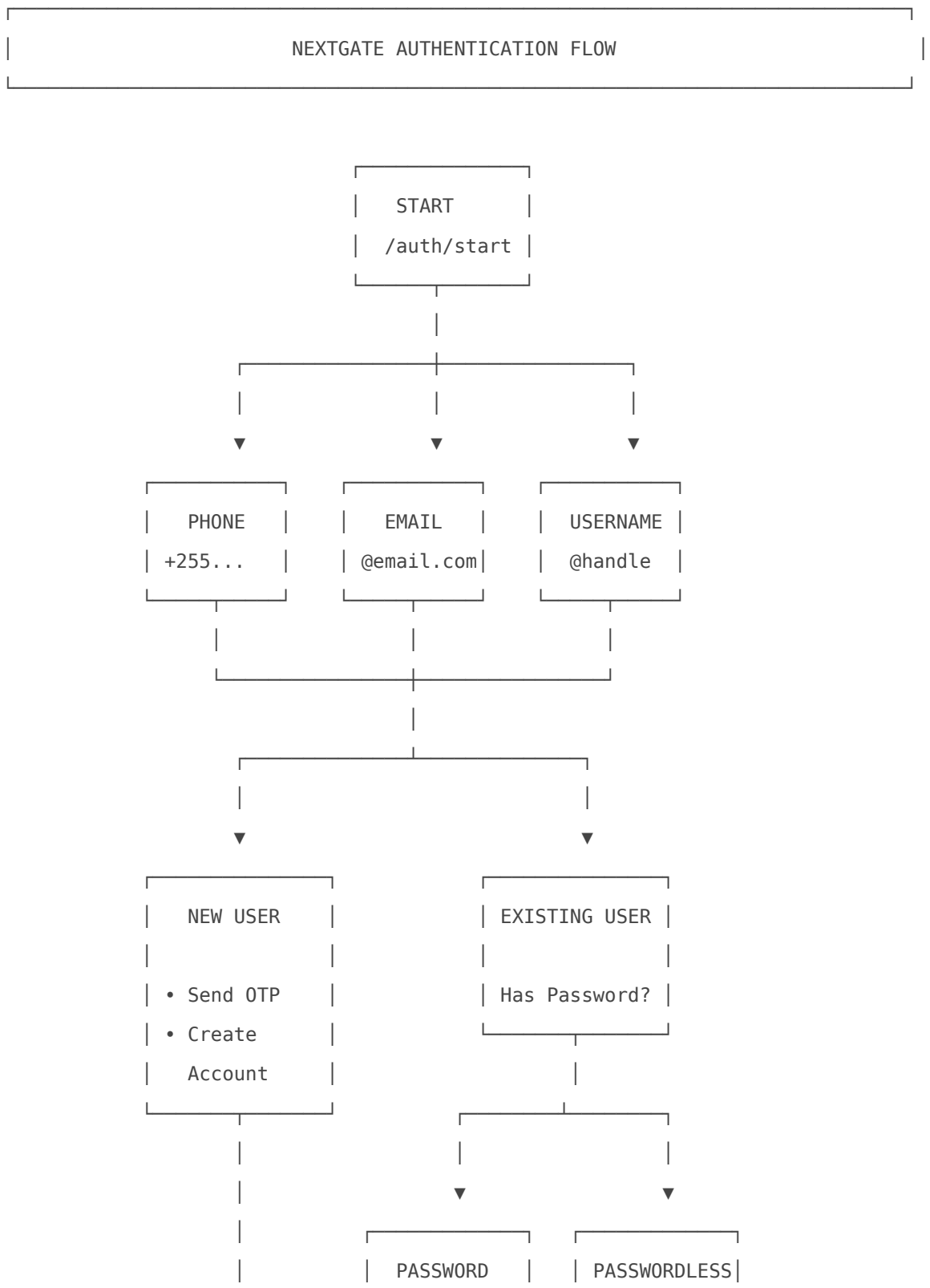
System Design Philosophy

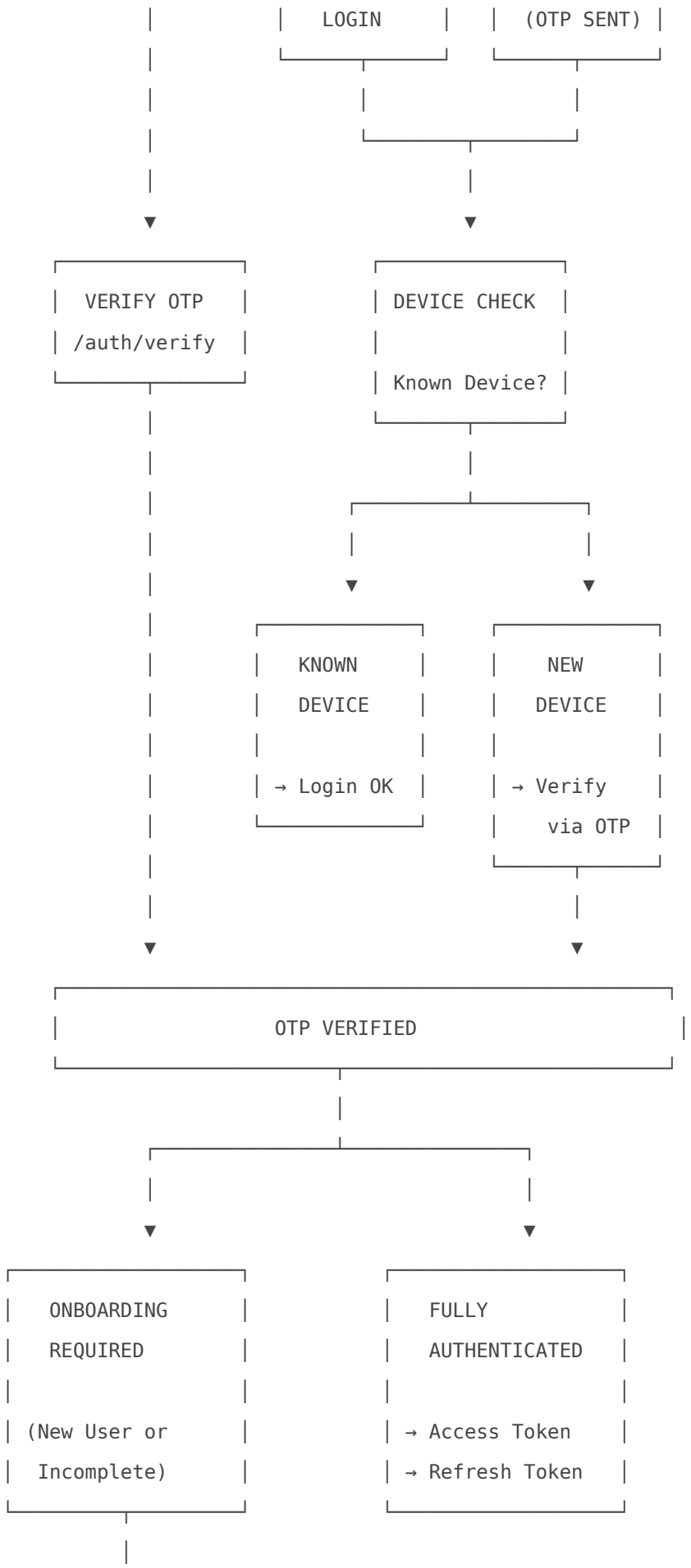
NextGate Authentication is built on three core principles:

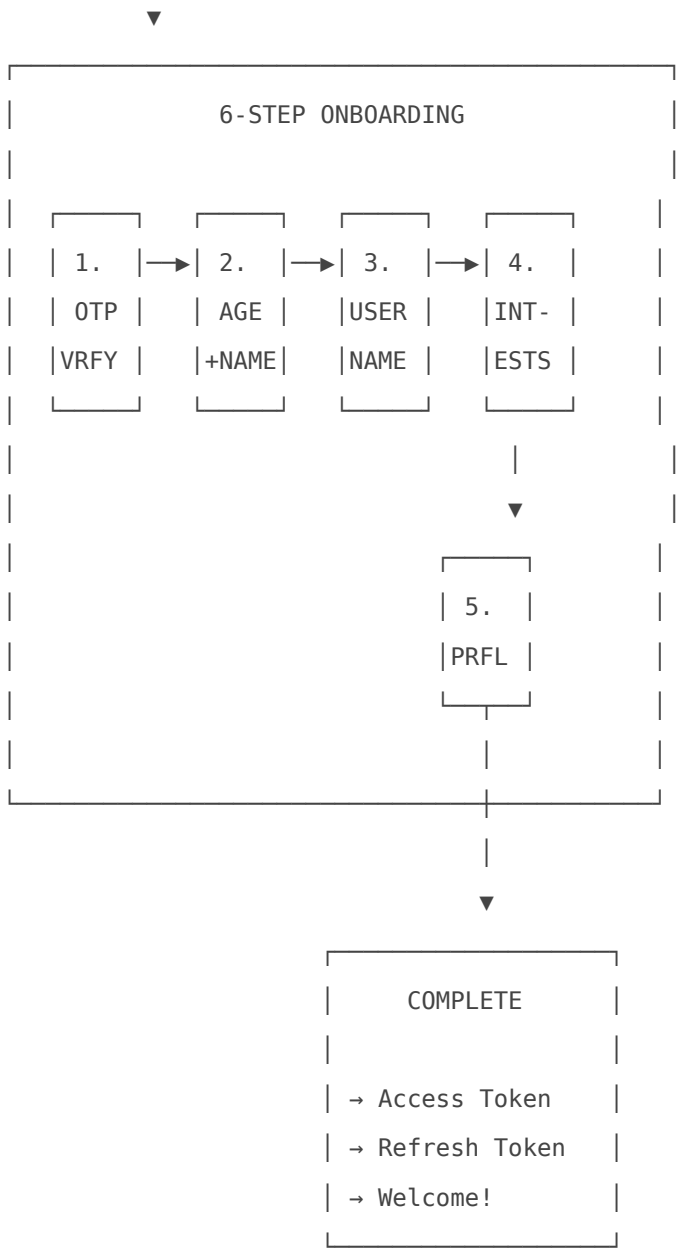
1. **Passwordless-First**: Users authenticate primarily via OTP, with password as an optional secondary method

2. **Device-Aware Security**: Every login tracks device information for risk assessment
3. **Progressive Onboarding**: New users complete a 6-step guided setup process

Authentication Flow Diagram




LOGIN
(OTP SENT)
VERIFY OTP
/auth/verify
DEVICE CHECK
Known Device?
KNOWN
DEVICE
→ Login OK
NEW
DEVICE
→ Verify
via OTP
OTP VERIFIED
ONBOARDING
REQUIRED
(New User or
Incomplete)
FULLY
AUTHENTICATED
→ Access Token
→ Refresh Token



Token Types

Token	Purpose	Expiry	Storage
`tempToken`	OTP verification, flow state	10 min	Memory only
`onboardingToken`	Onboarding step progression	30 min	Memory only
`onboardingRefreshToken`	Refresh onboarding token	24 hours	Secure storage
`accessToken`	API authentication	1 hour	Secure storage
`refreshToken`	Get new access tokens	30 days	Secure storage
`deviceVerificationToken`	New device verification	10 min	Memory only

Onboarding Steps Explained

Step	Enum Value	What Happens	Metadata Provided
1	`INITIATED`	Account created, OTP sent	-
2	`OTP_VERIFIED`	OTP confirmed, ready for age	`firstName`, `lastName`, `profilePicture` (from OAuth)
3	`AGE_VERIFIED`	Age confirmed, names saved	`suggestedUsernames` (5 smart suggestions)
4	`USERNAME_SET`	Username chosen	-
5	`INTERESTS_SELECTED`	Interests saved	`firstName`, `lastName`, `username`, `suggestedDisplayName`, `profilePicture`
6	`COMPLETED`	Profile complete, tokens issued	-

Device Fingerprinting Guide

Why Device Fingerprinting?

Device fingerprinting creates a unique identifier for each device/browser combination. This enables:

- **Security**: Detect new/unknown devices attempting login
- **Trust Levels**: Known devices can skip additional verification
- **Session Management**: Users can see and revoke device access

Implementation for Frontend

Recommended Library: FingerprintJS

```
npm install @fingerprintjs/fingerprintjs
```

React Implementation

```
// hooks/useDeviceFingerprint.js
import { useState, useEffect } from 'react';
import FingerprintJS from '@fingerprintjs/fingerprintjs';

export const useDeviceFingerprint = () => {
  const [fingerprint, setFingerprint] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const generateFingerprint = async () => {
      try {
        const fp = await FingerprintJS.load();
        const result = await fp.get();

        setFingerprint({
          deviceId: result.visitorId,
          confidence: result.confidence.score,
          components: {
            platform: result.components.platform?.value,
            timezone: result.components.timezone?.value,
            language: result.components.languages?.value?.[0],
            screenResolution: result.components.screenResolution?.value,
            colorDepth: result.components.colorDepth?.value,
          }
        });
      } catch (error) {
        console.error('Fingerprint generation failed:', error);
        // Fallback: Generate a random UUID and store in localStorage
        let fallbackId = localStorage.getItem('ng_device_id');
        if (!fallbackId) {
          fallbackId = crypto.randomUUID();
          localStorage.setItem('ng_device_id', fallbackId);
        }
        setFingerprint({ deviceId: fallbackId, confidence: 0.5 });
      } finally {
        setLoading(false);
      }
    };
```

```
    generateFingerprint();
  }, []);

  return { fingerprint, loading };
};
```

Usage in Authentication

```
// components/LoginForm.jsx
import { useDeviceFingerprint } from '../hooks/useDeviceFingerprint';

const LoginForm = () => {
  const { fingerprint, loading } = useDeviceFingerprint();

  const handleStartAuth = async (identifier) => {
    if (loading || !fingerprint) {
      console.warn('Device fingerprint not ready');
      return;
    }

    const response = await fetch('/api/v1/auth/start', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        identifier: identifier,
        deviceId: fingerprint.deviceId,
        deviceName: getDeviceName(),
        platform: getPlatform(),
      }),
    });

    const data = await response.json();
    // Handle response...
  };

  return (/* Your form JSX */);
};

// Helper: Generate human-readable device name
```

```
const getDeviceName = () => {
  const ua = navigator.userAgent;
  let browser = 'Unknown Browser';
  let os = 'Unknown OS';

  if (ua.includes('Chrome')) browser = 'Chrome';
  else if (ua.includes('Firefox')) browser = 'Firefox';
  else if (ua.includes('Safari')) browser = 'Safari';
  else if (ua.includes('Edge')) browser = 'Edge';

  if (ua.includes('Windows')) os = 'Windows';
  else if (ua.includes('Mac')) os = 'macOS';
  else if (ua.includes('Linux')) os = 'Linux';
  else if (ua.includes('Android')) os = 'Android';
  else if (ua.includes('iPhone') || ua.includes('iPad')) os = 'iOS';

  return `${browser} on ${os}`;
};

// Helper: Get platform type
const getPlatform = () => {
  const ua = navigator.userAgent;
  if (ua.includes('Android')) return 'ANDROID';
  if (ua.includes('iPhone') || ua.includes('iPad')) return 'IOS';
  return 'WEB';
};
```

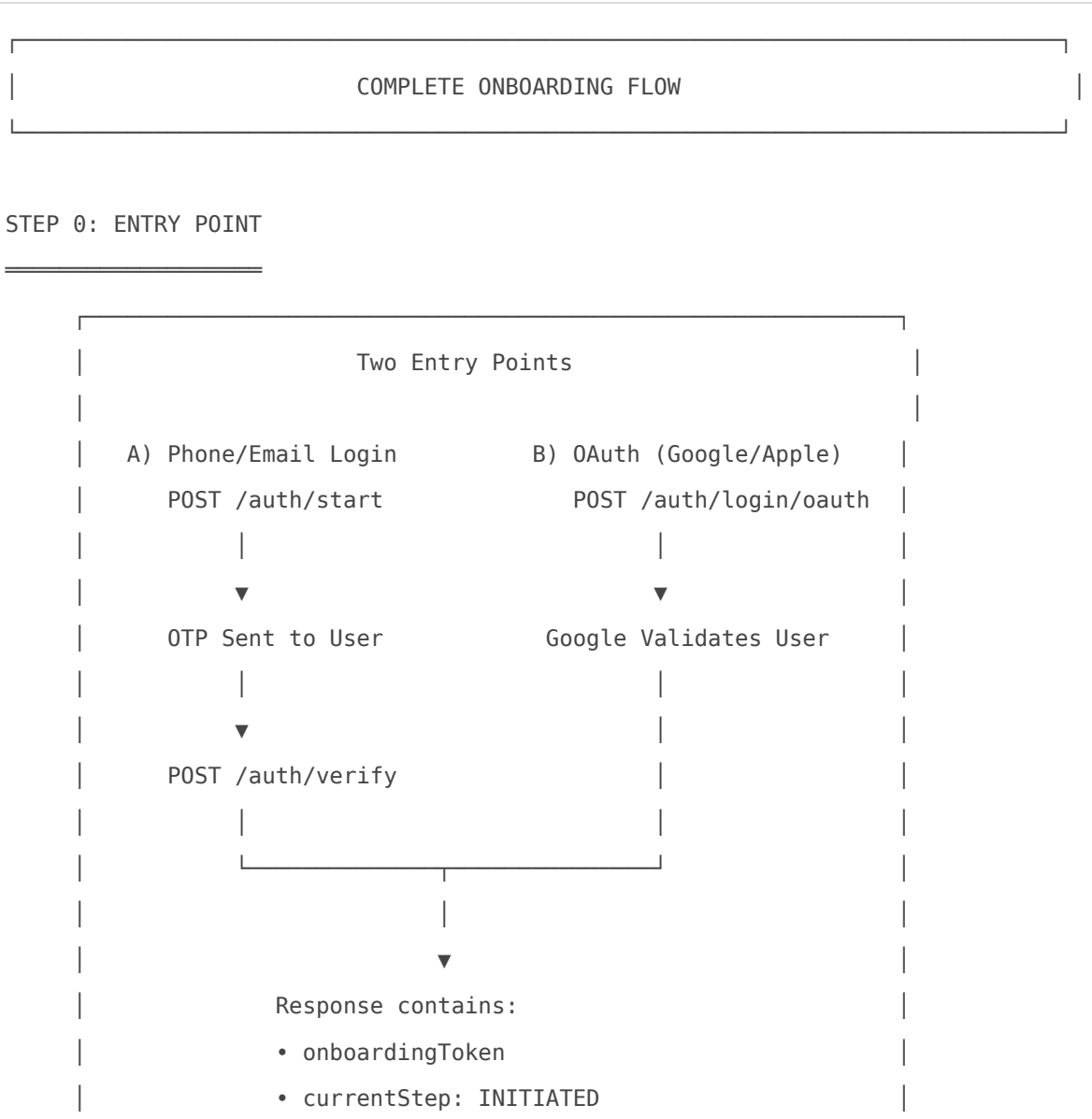
Alternative: Custom Fingerprint (No Library)

```
// utils/customFingerprint.js
export const generateCustomFingerprint = async () => {
  const components = [];

  // Screen properties
  components.push(window.screen.width);
  components.push(window.screen.height);
  components.push(window.screen.colorDepth);
  components.push(window.devicePixelRatio);
```

```
  // Timezone & Language
  components.push(Intl.DateTimeFormat().resolvedOptions().timeZone);
  components.push(navigator.language);
  components.push(navigator.platform);
  components.push(navigator.hardwareConcurrency || 'unknown');

  // Canvas fingerprint
  try {
    const canvas = document.createElement('canvas');
    const ctx = canvas.getContext('2d');
    ctx.textBaseline = 'top';
    ctx.font = '14px Arial';
    ctx.fillText('NextGate Device FP', 2, 2);
    components.push(canvas.toDataURL());
  } catch (e) {
    components.push('canvas-blocked');
  }

  // WebGL renderer
  try {
    const canvas = document.createElement('canvas');
    const gl = canvas.getContext('webgl');
    const debugInfo = gl.getExtension('WEBGL_debug_renderer_info');
    if (debugInfo) {
      components.push(gl.getParameter(debugInfo.UNMASKED_RENDERER_WEBGL));
    }
  } catch (e) {
    components.push('webgl-blocked');
  }

  // Create hash
  const fingerprint = components.join('|||');
  const encoder = new TextEncoder();
  const data = encoder.encode(fingerprint);
  const hashBuffer = await crypto.subtle.digest('SHA-256', data);
  const hashArray = Array.from(new Uint8Array(hashBuffer));
  return hashArray.map(b => b.toString(16).padStart(2, '0')).join('');
};
```

Required Headers for All Auth Requests

```
const authHeaders = {
  'Content-Type': 'application/json',
  'X-Device-Id': fingerprint.deviceId,
  'X-Device-Name': getDeviceName(),
  'X-Platform': getPlatform(),
};

// For authenticated requests:
authHeaders['Authorization'] = `Bearer ${accessToken}`;

// For session management:
authHeaders['X-Session-Id'] = sessionId;
```



```
│              • nextStep: AGE_VERIFIED                        │
│              • nextStepMetadata: {                           │
│                  firstName: "John",    // from OAuth        │
│                  lastName: "Doe",      // from OAuth        │
│                  profilePicture: "..." // from OAuth        │
│                  hasOAuthData: true                          │
│                }                                             │
└──────────────────────────────────────────────────────────────┘
                              │
                              ▼
STEP 1: AGE & NAME VERIFICATION
═══════════════════════════════

POST /auth/onboarding/age


┌──────────────────────────────────────────────────────────────┐
│  UI:                                                         │
│  ┌──────────────────────────────────────────────────────┐    │
│  │  □□Let's verify your age                            │  │
│  │                                                      │  │
│  │  First Name: [John_____________] <- Pre-filled OAuth  │  │
│  │  Last Name:  [Doe_____________] <- Pre-filled OAuth  │  │
│  │  Birth Date: [____/____/______]                      │  │
│  │                                                      │  │
│  │  [Continue →]                                        │  │
│  └──────────────────────────────────────────────────────┘    │
│                                                              │
│  REQUEST:                                                    │
│  {                                                           │
│    "onboardingToken": "eyJ...",                              │
│    "firstName": "John",                                      │
│    "lastName": "Doe",                                        │
│    "birthDate": "1999-05-15"                                 │
│  }                                                           │
│                                                              │
│  RESPONSE:                                                   │
│  {                                                           │
│    "onboardingToken": "eyJ...",                              │
│    "accountTier": "FULL",                                    │
│    "age": 25,                                                │
```

```
    |    "currentStep": "AGE_VERIFIED",                              |
    |    "nextStep": "USERNAME_SET",                                 |
    |    "nextStepMetadata": {                                       |
    |      "suggestedUsernames": [                                   |
    |        "johndoe99", "john_doe_official", "jdoe_tz"             |
    |      ]                                                         |
    |    }                                                           |
    |  }                                                             |
    |                                                                |
    |  ⚠  age < 13  -> blocked, account deleted                      |
    |  ⚠  age 13-17 -> accountTier = RESTRICTED                      |
    └────────────────────────────────────────────────────────────────┘
                                |
                                ▼
STEP 2: USERNAME SELECTION
══════════════════════════

POST /auth/onboarding/username
POST /auth/onboarding/check-username  (real-time availability)

    ┌────────────────────────────────────────────────────────────────┐
    |  UI:                                                           |
    |  ┌──────────────────────────────────────────────────────┐      |
    |  |  □□Choose your username                             |  |   |
    |  |                                                      |  |   |
    |  |  @[_________________]                               |  |   |
    |  |                                                      |  |   |
    |  |  Suggestions:                                       |  |   |
    |  |  [ johndoe99 ]  [ john_doe ]  [ jdoe_tz ]           |  |   |
    |  |  [ the_johndoe ]  [ johnd_pro ]                     |  |   |
    |  |                                                      |  |   |
    |  |  [Continue →]                                       |  |   |
    |  └──────────────────────────────────────────────────────┘      |
    |                                                                |
    |  RESPONSE:                                                     |
    |  {                                                             |
    |    "onboardingToken": "eyJ...",                                |
    |    "username": "johndoe99",                                    |
    |    "currentStep": "USERNAME_SET",                              |
    |    "nextStep": "CONTACT_VERIFIED",          <-- NEW            |
```

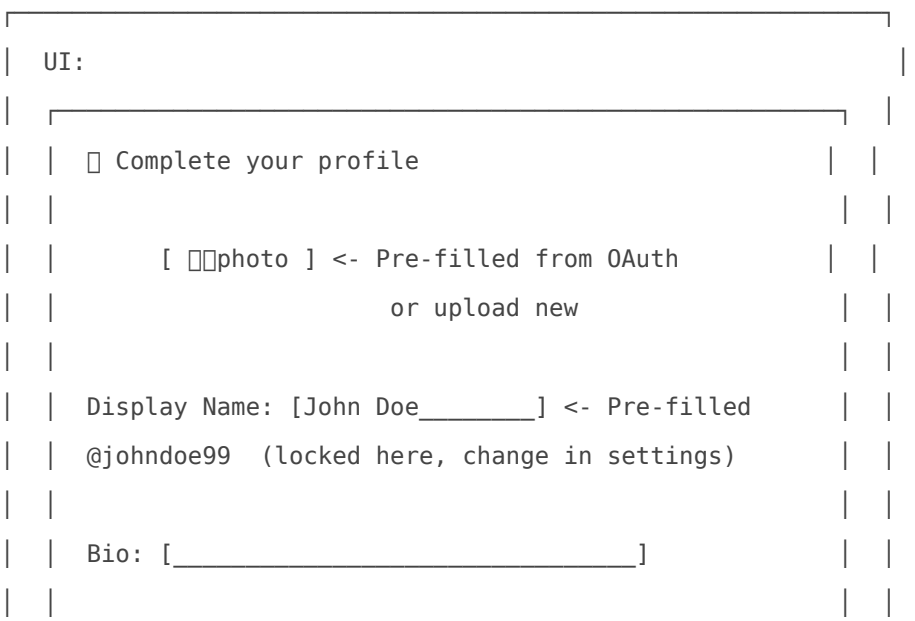
```
    |    "nextStepMetadata": {                                    |
    |      "contactVerification": {                               |
    |        "requiredContactType": "EMAIL",                      |
    |        "requiresInput": true,                               |
    |        "alreadyVerified": {                                 |
    |          "type": "PHONE",                                   |
    |          "maskedValue": "... ... ..50"                      |
    |        },                                                   |
    |        "reasons": [                                         |
    |          "TICKET_DELIVERY",                                 |
    |          "ACCOUNT_RECOVERY",                                |
    |          "ACCOUNT_SECURITY"                                 |
    |        ]                                                    |
    |      }                                                      |
    |    }                                                        |
    |  }                                                          |
    └─────────────────────────────────────────────────────────────┘
                              |
                              ▼
STEP 3: CONTACT VERIFICATION                          <-- NEW STEP
═══════════════════════════

POST /auth/onboarding/contact/initiate
POST /auth/onboarding/contact/verify
POST /auth/onboarding/contact/edit    (inline, no page nav)

    ┌─────────────────────────────────────────────────────────────┐
    |  UI (single page, three inline states):                     |
    |                                                             |
    |  STATE A — Input (if requiresInput = true):                 |
    |  ┌──────────────────────────────────────────────────────┐   |
    |  |  □□Add your email address                          |  |
    |  |                                                      |  |
    |  |  Your phone ... ...50 is already verified □         |  |
    |  |                                                      |  |
    |  |  We need your email for:                             |  |
    |  |  • Ticket delivery                                   |  |
    |  |  • Account recovery                                  |  |
    |  |  • Security alerts                                   |  |
    |  |                                                      |  |
```

```
|  |  Email: [_____________________]                |  |
|  |  [Send Code →]                                 |  |
|  └─────────────────────────────────────────────────┘  |
|                                                      |
|  INITIATE REQUEST:                                   |
|  {                                                   |
|    "onboardingToken": "eyJ...",                      |
|    "contactValue": "john@gmail.com"                  |
|  }                                                   |
|                                                      |
|  INITIATE RESPONSE:                                  |
|  {                                                   |
|    "tempToken": "eyJ...",                            |
|    "onboardingToken": "eyJ...",                      |
|    "maskedValue": "j......@g.....com",               |
|    "contactType": "EMAIL",                           |
|    "expiresIn": 600                                  |
|  }                                                   |
|                                                      |
|  STATE B — OTP Input:                                |
|  ┌─────────────────────────────────────────────────┐  |
|  |  Enter the code sent to j......@g.....com      |  |
|  |                                                |  |
|  |  [_] [_] [_] [_] [_] [_]                       |  |
|  |                                                |  |
|  |  [Edit email] · Resend (60s cooldown)          |  |
|  └─────────────────────────────────────────────────┘  |
|                                                      |
|  VERIFY REQUEST:                                     |
|  {                                                   |
|    "onboardingToken": "eyJ...",                      |
|    "tempToken": "eyJ...",                            |
|    "otp": "847291"                                   |
|  }                                                   |
|                                                      |
|  VERIFY RESPONSE:                                    |
|  {                                                   |
|    "onboardingToken": "eyJ...",                      |
|    "currentStep": "CONTACT_VERIFIED",                |
```

```
      |    "nextStep": "INTERESTS_SELECTED",                          |
      |    "verified": true                                           |
      |  }                                                            |
      |                                                               |
      |  STATE C — Edit (tapping [Edit email] transforms inline):     |
      |  ┌──────────────────────────────────────────────────────┐     |
      |  |  New email: [_____________________________]         |     |
      |  |  [Send new code →]       [Cancel]                    |     |
      |  └──────────────────────────────────────────────────────┘     |
      |                                                               |
      |  EDIT REQUEST:                                                |
      |  {                                                            |
      |    "onboardingToken": "eyJ...",                               |
      |    "tempToken": "eyJ...",                                     |
      |    "newContactValue": "other@gmail.com"                       |
      |  }                                                            |
      |  -> Returns new tempToken + new maskedValue                   |
      |  -> Previous OTP invalidated immediately                      |
      |                                                               |
      |  requiredContactType by signup method:                        |
      |  • Phone user   -> verify EMAIL                               |
      |  • Email user   -> verify PHONE                               |
      |  • Google/Apple -> verify PHONE (email via OAuth already)     |
      └──────────────────────────────────────────────────────────────┘
                                |
                                ▼
STEP 4: INTERESTS SELECTION
═══════════════════════════

GET  /interests/categories/all
POST /auth/onboarding/interests


      ┌──────────────────────────────────────────────────────────────┐
      |  UI:                                                         |
      |  ┌──────────────────────────────────────────────────────┐    |
      |  |  □□What are you interested in?  (select at least 3) |    |
      |  |                                                      |    |
      |  |  [□□Music] [□ Sports] [□□Gaming] [□□Tech]            |    |
      |  |  [□□Movies] [□□Books] [□□Food]  [✈□ Travel]          |    |
      |  |                          ... more                    |    |
```

STEP 5: PROFILE COMPLETION (FINAL)

POST /auth/onboarding/profile




[□□Complete Setup]
HEADERS:
X-Device-Id: "abc123fingerprint"
X-Device-Name: "Chrome on macOS"
X-Platform: "WEB"
REQUEST:
{
"onboardingToken": "eyJ...",
"displayName": "John Doe",
"bio": "Tech enthusiast | Based in Dar es Salaam",
"profilePictureUrl": "https://storage.nextgate.com/..."
}
RESPONSE:
{
"accessToken": "eyJ...",
"refreshToken": "eyJ...",
"systemUsername": "su_uuid",
"username": "johndoe99",
"displayName": "John Doe",
"currentStep": "COMPLETED",
"onboardingComplete": true,
"message": "Welcome to NextGate!"
}

□□WELCOME!
User is fully
authenticated and
can access the app

Handling Onboarding Resume

When a user returns with incomplete onboarding:

```
const handleOnboardingResponse = (response) => {
  const { currentStep, nextStep, nextStepMetadata, onboardingToken } = response.data;

  sessionStorage.setItem('onboardingToken', onboardingToken);

  switch (nextStep) {
    case 'AGE_VERIFIED':
      navigate('/onboarding/age', { state: { metadata: nextStepMetadata } });
      break;
    case 'USERNAME_SET':
      navigate('/onboarding/username', { state: { metadata: nextStepMetadata } });
      break;
    case 'INTERESTS_SELECTED':
      navigate('/onboarding/interests');
      break;
    case 'PROFILE_COMPLETED':
      navigate('/onboarding/profile', { state: { metadata: nextStepMetadata } });
      break;
    case 'COMPLETED':
      handleFullAuth(response.data);
      break;
  }
};
```

Standard Response Format

All API responses follow a consistent structure:

Success Response Structure:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Operation completed successfully",
```

```
  "action_time": "2025-01-24T10:30:45",
  "data": { }
}
```

Error Response Structure:

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Error description",
  "action_time": "2025-01-24T10:30:45",
  "data": "Error description"
}
```

Standard Response Fields:

Field	Type	Description
`success`	boolean	`true` for successful operations, `false` for errors
`httpStatus`	string	HTTP status name (OK, BAD_REQUEST, NOT_FOUND, etc.)
`message`	string	Human-readable message
`action_time`	string	ISO 8601 timestamp
`data`	object/string	Response payload or error details

API Endpoints

Authentication Initialization

1. Start Authentication

Purpose: Initiate authentication flow for phone, email, or username

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

Access Level: □□ Public

Authentication: None

Request JSON Sample:

```
{
  "identifier": "+255712345678",
  "deviceId": "a1b2c3d4e5f6789",
  "deviceName": "Chrome on macOS",
  "platform": "WEB"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
identifier	string	Yes	Phone (+255...), email, or @username	Valid format
deviceId	string	Yes	Device fingerprint	Non-empty
deviceName	string	Yes	Human-readable device name	Non-empty
platform	string	Yes	Device platform	enum: `IOS`, `ANDROID`, `WEB`

Success Response JSON Sample (New User):

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Authentication initiated",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "maskedIdentifier": "••• ••• ••78",
    "provider": "PHONE",
    "otpExpiresIn": 600,
    "isNewUser": true,
    "onboardingComplete": false,
    "currentStep": "INITIATED",
    "hasPassword": false,
    "requiresOtpChannelSelection": false,
    "message": "OTP sent to ••• ••• ••78"
  }
```

```
}
```

Success Response JSON Sample (Existing User with Password):

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Authentication initiated",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "maskedIdentifier": "j••••••@g••••.com",
    "provider": "EMAIL",
    "otpExpiresIn": 600,
    "isNewUser": false,
    "onboardingComplete": true,
    "currentStep": "COMPLETED",
    "hasPassword": true,
    "requiresOtpChannelSelection": false,
    "message": "Choose login method: password or OTP"
  }
}
```

Success Response JSON Sample (Username - Channel Selection Required):

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Authentication initiated",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "hasPassword": true,
    "requiresOtpChannelSelection": true,
    "availableChannels": [
      { "type": "EMAIL", "maskedValue": "j••••••@g••••.com", "isPrimary": true },
      { "type": "PHONE", "maskedValue": "••• ••• ••78", "isPrimary": false }
    ],
    "isNewUser": false,
    "onboardingComplete": true,
```

```
    "message": "Select where to receive OTP"
  }
}
```

Success Response Fields:

Field	Description
tempToken	Temporary token for next step
maskedIdentifier	Masked phone/email for display
provider	Identifier type: `PHONE`, `EMAIL`
otpExpiresIn	OTP validity in seconds (600 = 10 minutes)
isNewUser	Whether this is a new registration
onboardingComplete	Whether user completed onboarding
currentStep	Current onboarding step
hasPassword	Whether user has set a password
requiresOtpChannelSelection	If true, user must choose OTP channel
availableChannels	List of available OTP channels

Error Responses:

Identifier Blocked (400):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "This phone is blocked",
  "action_time": "2025-01-24T10:30:45",
  "data": "This phone is blocked"
}
```

Account Not Found - Username (404):

```
{
  "success": false,
  "httpStatus": "NOT_FOUND",
  "message": "Account not found",
  "action_time": "2025-01-24T10:30:45",
  "data": "Account not found"
}
```

2. Verify Authentication OTP

Purpose: Verify OTP and complete authentication or continue to onboarding

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

Access Level: Public

Authentication: None

Request JSON Sample:

```
{
  "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "otp": "123456",
  "deviceId": "a1b2c3d4e5f6789",
  "deviceName": "Chrome on macOS",
  "platform": "WEB"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
tempToken	string	Yes	Token from /auth/start	Valid JWT
otp	string	Yes	6-digit OTP code	Exactly 6 digits
deviceId	string	No	Device fingerprint	-
deviceName	string	No	Human-readable device name	-
platform	string	No	Device platform	enum: `IOS`, `ANDROID`, `WEB`

Success Response JSON Sample (New User - Start Onboarding):

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Verification successful",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
```

```
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "onboardingComplete": false,
    "currentStep": "OTP_VERIFIED",
    "nextStep": "AGE_VERIFIED",
    "nextStepMetadata": null,
    "message": "OTP verified. Let's set up your account."
  }
}
```

Success Response JSON Sample (OAuth User - Has Profile Data):

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Verification successful",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "onboardingComplete": false,
    "currentStep": "OTP_VERIFIED",
    "nextStep": "AGE_VERIFIED",
    "nextStepMetadata": {
      "firstName": "John",
      "lastName": "Doe",
      "profilePicture": "https://lh3.googleusercontent.com/...",
      "hasOAuthData": true
    },
    "message": "OTP verified. Let's set up your account."
  }
}
```

Success Response JSON Sample (Existing User - Fully Authenticated):

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Verification successful",
  "action_time": "2025-01-24T10:30:45",
  "data": {
```

```
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "systemUsername": "su_550e8400-e29b-41d4-a716-446655440000",
    "username": "johndoe99",
    "displayName": "John Doe",
    "accountTier": "FULL",
    "onboardingComplete": true,
    "currentStep": "COMPLETED",
    "message": "Login successful"
  }
}
```


Error Responses:

Invalid OTP (403):


```
{
  "success": false,
  "httpStatus": "FORBIDDEN",
  "message": "Invalid OTP code",
  "action_time": "2025-01-24T10:30:45",
  "data": "Invalid OTP code"
}
```


Max Attempts Exceeded (403):


```
{
  "success": false,
  "httpStatus": "FORBIDDEN",
  "message": "Maximum verification attempts exceeded",
  "action_time": "2025-01-24T10:30:45",
  "data": "Maximum verification attempts exceeded"
}
```


---

3. Send OTP to Channel

Purpose: Send OTP to a specific channel when user has multiple options

Endpoint: `POST` `{base_url}/auth/send-otp-to-channel`

Access Level: □□ Public

Authentication: None

Request JSON Sample:

```
{
  "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "channel": "EMAIL"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
tempToken	string	Yes	Token from /auth/start	Valid JWT
channel	string	Yes	Preferred OTP channel	enum: `EMAIL`, `PHONE`

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "OTP sent",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "sentTo": "EMAIL",
    "maskedDestination": "j••••••@g••••.com",
    "expiresIn": 600,
    "message": "OTP sent to email"
  }
}
```

---

4. Login with Password

Purpose: Authenticate using password instead of OTP

Endpoint: `POST` `{base_url}/auth/login/password`

Access Level: □□ Public

Authentication: None

Request JSON Sample:

```
{
  "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "password": "mySecurePassword123",
  "deviceId": "a1b2c3d4e5f6789",
  "deviceName": "Chrome on macOS",
  "platform": "WEB"
}
```

Success Response JSON Sample (Known Device):

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Login successful",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "systemUsername": "su_550e8400-e29b-41d4-a716-446655440000",
    "username": "johndoe99",
    "displayName": "John Doe",
    "accountTier": "FULL",
    "newDevice": false,
    "requiresDeviceVerification": false,
    "message": "Login successful"
  }
}
```

Success Response JSON Sample (New Device - Verification Required):

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Login successful",
  "action_time": "2025-01-24T10:30:45",
```

```
  "data": {
    "newDevice": true,
    "requiresDeviceVerification": true,
    "deviceVerificationToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "requiresChannelSelection": false,
    "otpSentTo": "••• ••• ••78",
    "message": "Device verification required. OTP sent."
  }
}
```


Error Responses:

Invalid Password (403):


```
{
  "success": false,
  "httpStatus": "FORBIDDEN",
  "message": "Invalid password",
  "action_time": "2025-01-24T10:30:45",
  "data": "Invalid password"
}
```


---

5. OAuth Login (Google)

Purpose: Authenticate using Google OAuth2 authorization code flow

Endpoint: `POST` `{base_url}/auth/login/oauth`

Access Level: □□ Public

Authentication: None

Request JSON Sample:


```
{
  "provider": "GOOGLE",
  "code": "4/0AX4XfWh...",
  "redirectUri": "https://app.nextgate.com/auth/callback",
  "state": "random-state-string",
  "deviceId": "a1b2c3d4e5f6789",
  "deviceName": "Chrome on macOS",
```

```
  "platform": "WEB"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
provider	string	Yes	OAuth provider	enum: `GOOGLE`, `APPLE`
code	string	Yes	Authorization code	Non-empty
redirectUri	string	Yes	Redirect URI	Must match OAuth config
state	string	No	State for CSRF protection	-
deviceId	string	No	Device fingerprint	-
deviceName	string	No	Device name	-
platform	string	No	Device platform	enum: `IOS`, `ANDROID`, `WEB`

Success Response JSON Sample (New User):

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "OAuth login processed",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "newUser": true,
    "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "currentStep": "OTP_VERIFIED",
    "nextStep": "AGE_VERIFIED",
    "nextStepMetadata": {
      "firstName": "John",
      "lastName": "Doe",
      "profilePicture": "https://lh3.googleusercontent.com/a/...",
      "hasOAuthData": true
    },
    "message": "Complete your registration",
    "state": "random-state-string"
  }
}
```

Success Response JSON Sample (Existing User):


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "OAuth login processed",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "newUser": false,
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "systemUsername": "su_550e8400-e29b-41d4-a716-446655440000",
    "username": "johndoe99",
    "displayName": "John Doe",
    "accountTier": "FULL",
    "message": "Login successful",
    "state": "random-state-string"
  }
}
```


---

6. Verify Device

Purpose: Verify a new device using OTP

Endpoint: POST `{base_url}/auth/device/verify`

Access Level: □□ Public

Authentication: None

Request JSON Sample:


```
{
  "deviceVerificationToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "otp": "123456",
  "deviceId": "a1b2c3d4e5f6789",
  "deviceName": "Chrome on macOS",
  "platform": "WEB"
}
```


Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Device verified",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "systemUsername": "su_550e8400-e29b-41d4-a716-446655440000",
    "username": "johndoe99",
    "displayName": "John Doe",
    "accountTier": "FULL",
    "message": "Login successful"
  }
}
```


---

7. Resend OTP

Purpose: Resend OTP code (with rate limiting)

Endpoint: `POST` `{base_url}/auth/resend-otp`

Access Level: □□ Public

Authentication: None

Request JSON Sample:


```
{
  "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```


Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "OTP resend processed",
  "action_time": "2025-01-24T10:30:45",
  "data": {
```

```
    "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "maskedDestination": "••• ••• ••78",
    "sentTo": "PHONE",
    "expiresIn": 600,
    "attemptsRemaining": 4,
    "cooldownSeconds": 0,
    "message": "OTP resent"
  }
}
```

Cooldown Response:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "OTP resend processed",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "cooldownSeconds": 45,
    "attemptsRemaining": 3,
    "message": "Please wait 45 seconds before requesting again"
  }
}
```

Onboarding Endpoints

---

8. Set Age (Step 1)

Purpose: Verify user's age and save first/last name

Endpoint: `POST {base_url}/auth/onboarding/age`

Access Level: Public (requires valid onboarding token)

Request Body:

```
{
  "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
```

```
  "firstName": "John",
  "lastName": "Doe",
  "birthDate": "1999-05-15"
}
```

Parameter	Type	Required	Description	Validation
onboardingToken	string	Yes	Token from previous step	Valid JWT
firstName	string	Yes	User's first name	1-50 characters
lastName	string	Yes	User's last name	1-50 characters
birthDate	string	Yes	Date of birth	ISO date (YYYY-MM-DD), must be in past

Success Response:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Age verified",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "accountTier": "FULL",
    "age": 25,
    "restricted": false,
    "blocked": false,
    "currentStep": "AGE_VERIFIED",
    "nextStep": "USERNAME_SET",
    "nextStepMetadata": {
      "suggestedUsernames": [
        "johndoe99",
        "john_doe_official",
        "jdoe_tz",
        "the_johndoe",
        "johnd_pro"
      ]
    },
    "message": "Age verified successfully"
  }
}
```

Blocked Response (Under 13):


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Age verified",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "accountTier": null,
    "age": 12,
    "restricted": true,
    "blocked": true,
    "currentStep": "AGE_VERIFIED",
    "nextStep": null,
    "message": "You must be at least 13 years old to use NextGate"
  }
}
```


> If age < 13: account is blocked and deleted. If age 13-17: `accountTier = RESTRICTED`.

---

9. Check Username Availability

Purpose: Real-time username availability check

Endpoint: `POST {base_url}/auth/onboarding/check-username`

Access Level: Public

Request Body:


```
{
  "username": "johndoe99"
}
```


Response - Available:


```
{
  "success": true,
```

```
  "httpStatus": "OK",
  "message": "Username availability checked",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "username": "johndoe99",
    "available": true
  }
}
```

Response - Not Available:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Username not available",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "username": "johndoe",
    "available": false,
    "suggestions": ["johndoe99", "johndoe_official", "the_johndoe"]
  }
}
```

---

10. Set Username (Step 2)

Purpose: Set the user's chosen username

Endpoint: `POST {base_url}/auth/onboarding/username`

Access Level: Public (requires valid onboarding token)

Request Body:

```
{
  "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "username": "johndoe99"
}
```

Parameter	Type	Required	Description	Validation

onboardingToken	string	Yes	Token from previous step	Valid JWT
username	string	Yes	Chosen username	3-30 chars, starts with letter, alphanumeric + underscore

Success Response:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Username set",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "username": "johndoe99",
    "available": true,
    "currentStep": "USERNAME_SET",
    "nextStep": "CONTACT_VERIFIED",
    "nextStepMetadata": {
      "contactVerification": {
        "requiredContactType": "EMAIL",
        "requiresInput": true,
        "alreadyVerified": {
          "type": "PHONE",
          "maskedValue": "... ... ..50"
        },
        "reasons": ["TICKET_DELIVERY", "ACCOUNT_RECOVERY", "ACCOUNT_SECURITY"]
      }
    },
    "message": "Username set successfully"
  }
}
```


> `requiredContactType` depends on signup method:
>
> - Phone user -> must verify `EMAIL`
> - Email user -> must verify `PHONE`

> - Google/Apple user -> must verify `PHONE` (email already verified via OAuth)

---

11. Initiate Contact Verification (Step 3)

Purpose: Send OTP to the user's secondary contact (email or phone)

Endpoint: `POST {base_url}/auth/onboarding/contact/initiate`

Access Level: Public (requires valid onboarding token)

Request Body:

```
{
  "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "contactValue": "johndoe@gmail.com"
}
```

Parameter	Type	Required	Description	Validation
onboardingToken	string	Yes	Token from previous step	Valid JWT
contactValue	string	Conditional	Email or phone to verify	Required only if `requiresInput = true`

Success Response:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "OTP sent",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "maskedValue": "j......@g.....com",
    "contactType": "EMAIL",
    "expiresIn": 600,
    "verified": false
  }
```

```
}
```

---

12. Verify Contact OTP (Step 3 - continued)

Purpose: Submit the OTP sent to the secondary contact

Endpoint: `POST {base_url}/auth/onboarding/contact/verify`

Access Level: Public (requires valid onboarding token)

Request Body:

```
{
  "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "otp": "847291"
}
```

Parameter	Type	Required	Description
onboardingToken	string	Yes	Current onboarding token
tempToken	string	Yes	Token returned from initiate endpoint
otp	string	Yes	6-digit OTP code

Success Response:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Contact verified",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "currentStep": "CONTACT_VERIFIED",
    "nextStep": "INTERESTS_SELECTED",
    "verified": true
  }
}
```

---

13. Edit Contact (Step 3 - inline edit)

Purpose: Change the contact value before verifying — no page navigation, inline only

Endpoint: `POST {base_url}/auth/onboarding/contact/edit`

Access Level: Public (requires valid onboarding token)

Request Body:

```
{
  "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "newContactValue": "newemail@gmail.com"
}
```

Parameter	Type	Required	Description
onboardingToken	string	Yes	Current onboarding token
tempToken	string	Yes	Token from initiate endpoint
newContactValue	string	Yes	New email or phone number

Success Response: Same shape as initiate — returns a new `tempToken` and new `maskedValue` for the updated contact.

> Previous OTP is invalidated immediately. A fresh OTP is sent to the new contact.

Duplicate contact handling:

- Taken by a **verified** account -> hard block, error returned
- Taken by an **unverified** account -> released automatically, current user proceeds
- Same account re-entering same contact -> OTP resent, no release cycle

---

14. Get Interest Categories

Purpose: Get all available interest categories for selection

Endpoint: `GET {base_url}/interests/categories/all`

Access Level: Public

Success Response:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Categories retrieved",
  "action_time": "2025-01-24T10:30:45",
  "data": [
    {
      "id": "550e8400-e29b-41d4-a716-446655440001",
      "name": "Music",
      "icon": "□□",
      "description": "Concerts, artists, playlists",
      "displayOrder": 1,
      "isActive": true
    },
    {
      "id": "550e8400-e29b-41d4-a716-446655440002",
      "name": "Sports",
      "icon": "□",
      "description": "Games, teams, fitness",
      "displayOrder": 2,
      "isActive": true
    }
  ]
}
```


---

15. Set Interests (Step 4)

Purpose: Save user's selected interests

Endpoint: `POST {base_url}/auth/onboarding/interests`

Access Level: Public (requires valid onboarding token)

Request Body:

```
{
  "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "interestIds": [
    "550e8400-e29b-41d4-a716-446655440001",
    "550e8400-e29b-41d4-a716-446655440002",
    "550e8400-e29b-41d4-a716-446655440003"
  ]
}
```

Parameter	Type	Required	Description	Validation
onboardingToken	string	Yes	Token from previous step	Valid JWT
interestIds	array	Yes	List of category UUIDs	Min 3 items

Success Response:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Interests saved",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "selectedInterests": ["Music", "Sports", "Technology"],
    "count": 3,
    "currentStep": "INTERESTS_SELECTED",
    "nextStep": "PROFILE_COMPLETED",
    "nextStepMetadata": {
      "firstName": "John",
      "lastName": "Doe",
      "username": "johndoe99",
      "suggestedDisplayName": "John Doe",
      "profilePicture": "https://lh3.googleusercontent.com/a/..."
    },
    "message": "Interests saved successfully"
  }
}
```

---

16. Set Profile (Step 5 - Final)

Purpose: Complete profile setup and finish onboarding

Endpoint: `POST {base_url}/auth/onboarding/profile`

Access Level: Public (requires valid onboarding token)

Request Headers:

Header	Type	Required	Description
X-Device-Id	string	No	Device fingerprint
X-Device-Name	string	No	Device name
X-Platform	string	No	`IOS`, `ANDROID`, or `WEB`

Request Body:

```
{
  "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "displayName": "John Doe",
  "bio": "Tech enthusiast | Music lover | Based in Dar es Salaam □□□",
  "profilePictureUrl": "https://storage.nextgate.com/profiles/abc123.jpg"
}
```

Parameter	Type	Required	Description	Validation
onboardingToken	string	Yes	Token from previous step	Valid JWT
displayName	string	Yes	Public display name	1-50 characters
bio	string	No	User biography	Max 160 characters
profilePictureUrl	string	No	Profile picture URL	Valid URL

Success Response:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Welcome to NextGate!",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
```

```
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "systemUsername": "su_550e8400-e29b-41d4-a716-446655440000",
    "username": "johndoe99",
    "displayName": "John Doe",
    "currentStep": "COMPLETED",
    "onboardingComplete": true,
    "message": "Welcome to NextGate!"
  }
}
```


17. Upload Profile Picture (Onboarding Step)

Purpose: Upload a profile picture during onboarding, before completing profile setup **Endpoint**: `POST {base_url}/auth/onboarding/upload-profile-picture` **Access Level**: Public (requires valid onboarding token) **Request**: `multipart/form-data`

Parameter	Type	Required	Description	Validation
file	file	Yes	Profile picture file	Image files only, max 25MB
X-Onboarding-Token	header	Yes	Token from previous step	Valid onboarding JWT

Success Response:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Profile picture uploaded",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "fileName": "a3f1c2d4-...-uuid.jpg",
    "originalFileName": "my_photo.jpg",
    "objectKey": "profile/a3f1c2d4-...-uuid.jpg",
    "directory": "PROFILE",
    "contentType": "image/jpeg",
    "fileSize": 204800,
    "fileSizeFormatted": "200.0 KB",
    "permanentUrl": "https://files.nextgate.co.tz/bucket-id/profile/a3f1c2d4-...-uuid.jpg",
    "thumbnailUrl": "https://files.nextgate.co.tz/bucket-id/profile/a3f1c2d4-...-uuid.jpg",
    "blurHash": "LKO2?U%2Tw=w]~RBVZRi};RPxuwH",
```

```
    "fileExtension": ".jpg",
    "fileType": "IMAGE",
    "isImage": true,
    "isVideo": false,
    "isDocument": false,
    "isAudio": false,
    "width": 800,
    "height": 800,
    "dimensions": "800x800",
    "checksum": "d41d8cd98f00b204e9800998ecf8427e",
    "uploadedAt": "2025-01-24T10:30:45",
    "isPublic": true
  }
}
```


> Use the returned `permanentUrl` as the value of `profilePictureUrl` in the subsequent **Set Profile** request.

> This endpoint is only accessible when the user has completed the `INTERESTS_SELECTED` step. Calling it earlier will result in a `403` verification error.

Token Management

14. Refresh Access Token

Purpose: Get a new access token using refresh token (with rotation)

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

Access Level: □□ Public

Authentication: None (refresh token in body)

Request JSON Sample:

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

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Token refreshed",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "tokenType": "Bearer",
    "expiresIn": 3600,
    "message": "Tokens refreshed successfully"
  }
}
```

> ⚠ **Important**: The refresh token is rotated on each use. Always store the new `refreshToken` from the response.

Error Response (Token Reuse Detected):

```
{
  "success": false,
  "httpStatus": "UNAUTHORIZED",
  "message": "Security alert: Token reuse detected. Please login again.",
  "action_time": "2025-01-24T10:30:45",
  "data": "Security alert: Token reuse detected. Please login again."
}
```

---

15. Revoke Token

Purpose: Revoke a refresh token (logout)

Endpoint: `POST` `{base_url}/auth/token/revoke`

Access Level: □□ Public

Authentication: None

Request JSON Sample:

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

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Token revoked successfully",
  "action_time": "2025-01-24T10:30:45",
  "data": null
}
```

---

16. Refresh Onboarding Token

Purpose: Refresh onboarding token if it's about to expire

Endpoint: `POST` `{base_url}/auth/token/refresh-onboarding`

Access Level: □□ Public

Authentication: None

Request JSON Sample:

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

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Onboarding token refreshed",
```

```
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "onboardingToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "currentStep": "USERNAME_SET",
    "nextStep": "INTERESTS_SELECTED",
    "expiresIn": 1800,
    "message": "Token refreshed successfully"
  }
}
```

Password Management

17. Initiate Forgot Password

Purpose: Start password reset flow

Endpoint: `POST` `{base_url}/auth/password/forgot/initiate`

Access Level: □□ Public

Authentication: None

Request JSON Sample:


```
{
  "identifier": "johndoe99"
}
```


Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Request processed",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "maskedDestination": "j••••••@g••••.com",
    "sentTo": "EMAIL",
```

```
    "expiresIn": 600,
    "otpVerified": false,
    "passwordReset": false,
    "requiresChannelSelection": false,
    "message": "OTP sent to email"
  }
}
```

---

18. Verify Forgot Password OTP

Purpose: Verify OTP for password reset

Endpoint: `POST` `{base_url}/auth/password/forgot/verify-otp`

Access Level: ◫ Public

Authentication: None

Request JSON Sample:

```
{
  "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "otp": "123456"
}
```

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "OTP verified",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "resetToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "otpVerified": true,
    "passwordReset": false,
    "message": "OTP verified. Set your new password."
  }
}
```

---

19. Reset Forgotten Password

Purpose: Set new password after OTP verification

Endpoint: `POST` `{base_url}/auth/password/forgot/reset`

Access Level: □□ Public

Authentication: None

Request JSON Sample:

```
{
  "resetToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "newPassword": "newSecurePassword123",
  "confirmPassword": "newSecurePassword123"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
resetToken	string	Yes	Token from verify-otp	Valid JWT
newPassword	string	Yes	New password	Min 8 characters
confirmPassword	string	Yes	Confirmation	Must match newPassword

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Password reset successfully",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "otpVerified": true,
    "passwordReset": true,
    "message": "Password reset successfully"
  }
}
```

> ⚠ **Note**: After password reset, all active sessions are revoked. User must login again.

---

20. Change Password (Authenticated)

Purpose: Change password for logged-in user

Endpoint: `POST` `{base_url}/password/change`

Access Level: □□ Protected

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	`Bearer {accessToken}`

Request JSON Sample:

```
{
  "currentPassword": "oldPassword123",
  "newPassword": "newSecurePassword456",
  "confirmPassword": "newSecurePassword456"
}
```

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Password changed",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "success": true,
    "hadPassword": true,
    "message": "Password changed successfully"
  }
}
```

---

21. Set Password (for Passwordless Users)

Purpose: Set password for users who registered via OAuth or OTP only

Endpoint: `POST` `{base_url}/password/set`

Access Level: □□ Protected

Authentication: Bearer Token

Request JSON Sample:

```
{
  "newPassword": "myNewPassword123",
  "confirmPassword": "myNewPassword123"
}
```

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Password set successfully",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "success": true,
    "hadPassword": false,
    "message": "Password set successfully. You can now login with password."
  }
}
```

Session Management

22. Get All Sessions

Purpose: List all active sessions for the user

Endpoint: `GET` `{base_url}/auth/sessions`

Access Level: □□ Protected

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	`Bearer {accessToken}`
X-Session-Id	string	No	Current session ID

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Sessions retrieved",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "sessions": [
      {
        "id": "550e8400-e29b-41d4-a716-446655440001",
        "deviceId": "a1b2c3d4e5f6789",
        "deviceName": "Chrome on macOS",
        "platform": "WEB",
        "ipAddress": "192.168.1.1",
        "location": "Dar es Salaam, Tanzania",
        "lastActiveAt": "2025-01-24T10:30:45",
        "createdAt": "2025-01-20T08:00:00",
        "currentSession": true
      },
      {
        "id": "550e8400-e29b-41d4-a716-446655440002",
        "deviceId": "xyz789abc123",
        "deviceName": "Safari on iPhone",
        "platform": "IOS",
        "ipAddress": "192.168.1.2",
        "location": "Arusha, Tanzania",
        "lastActiveAt": "2025-01-23T15:20:00",
        "createdAt": "2025-01-15T12:00:00",
        "currentSession": false
      }
    ],
```

```
    "totalCount": 2
  }
}
```

---

23. Sign Out (Current Session)

Purpose: End the current session

Endpoint: POST `{base_url}/auth/sessions/sign-out`

Access Level: □□ Protected

Authentication: Bearer Token

Request JSON Sample:

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

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Signed out successfully",
  "action_time": "2025-01-24T10:30:45",
  "data": null
}
```

---

24. Sign Out All Sessions

Purpose: End all sessions including current (security logout)

Endpoint: POST `{base_url}/auth/sessions/sign-out-all`

Access Level: □□ Protected

Authentication: Bearer Token

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "All sessions terminated",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "terminatedCount": 4,
    "message": "All 4 sessions have been terminated. Please login again."
  }
}
```


---

25. Revoke Specific Session

Purpose: End a specific session by ID

Endpoint: DELETE `{base_url}/auth/sessions/{sessionId}`

Access Level: □□ Protected

Authentication: Bearer Token

Path Parameters:

Parameter	Type	Required	Description
sessionId	string	Yes	Session UUID to revoke

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Session revoked",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "sessionId": "550e8400-e29b-41d4-a716-446655440002",
    "message": "Session has been terminated"
  }
}
```

Device Management

26. Get All Devices

Purpose: List all registered/trusted devices

Endpoint: `GET` `{base_url}/auth/devices`

Access Level: □□ Protected

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	`Bearer {accessToken}`
X-Device-Id	string	No	Current device ID

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Devices retrieved",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "devices": [
      {
        "id": "550e8400-e29b-41d4-a716-446655440001",
        "deviceId": "a1b2c3d4e5f6789",
        "deviceName": "Chrome on macOS",
        "platform": "WEB",
        "lastIpAddress": "192.168.1.1",
        "lastLocation": "Dar es Salaam, Tanzania",
        "lastActiveAt": "2025-01-24T10:30:45",
        "firstSeenAt": "2025-01-01T08:00:00",
        "trustLevel": "TRUSTED",
        "isCurrentDevice": true
      }
    ],
```

```
    "totalCount": 1
  }
}
```

---

27. Remove Device

Purpose: Remove a device from trusted devices (revokes all sessions on that device)

Endpoint: DELETE `{base_url}/auth/devices/{deviceId}`

Access Level: Protected

Authentication: Bearer Token

Path Parameters:

Parameter	Type	Required	Description
deviceId	string	Yes	Device record UUID

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Device removed",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "deviceId": "550e8400-e29b-41d4-a716-446655440002",
    "sessionsRevoked": 2,
    "message": "Device removed and 2 sessions terminated"
  }
}
```

Account Linking

28. Get Linked Accounts

Purpose: List all linked OAuth providers and identifiers

Endpoint: `GET` `{base_url}/auth/linked-accounts`

Access Level: ◫ Protected

Authentication: Bearer Token

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Linked accounts retrieved",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "linkedAccounts": [
      {
        "type": "EMAIL",
        "value": "john.doe@gmail.com",
        "isPrimary": true,
        "isVerified": true,
        "linkedAt": "2025-01-01T08:00:00"
      },
      {
        "type": "PHONE",
        "value": "+255712345678",
        "isPrimary": false,
        "isVerified": true,
        "linkedAt": "2025-01-05T10:00:00"
      },
      {
        "type": "OAUTH",
        "provider": "GOOGLE",
        "email": "john.doe@gmail.com",
        "linkedAt": "2025-01-01T08:00:00"
      }
    ],
    "hasPassword": true,
    "canRemoveEmail": true,
    "canRemovePhone": true
  }
}
```

29. Link Email

Purpose: Add a new email to the account

Endpoint: POST `{base_url}/auth/linked-accounts/email/link`

Access Level: □□ Protected

Authentication: Bearer Token

Request JSON Sample:

```
{
  "email": "john.work@company.com"
}
```

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Verification email sent",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "email": "j••••••@c•••••••.com",
    "expiresIn": 600,
    "message": "Verification code sent"
  }
}
```

30. Link Phone

Purpose: Add a new phone number to the account

Endpoint: POST `{base_url}/auth/linked-accounts/phone/link`

Access Level: □□ Protected

Authentication: Bearer Token

Request JSON Sample:

```
{
  "phone": "+255787654321"
}
```

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Verification SMS sent",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "tempToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "phone": "••• ••• ••21",
    "expiresIn": 600,
    "message": "Verification code sent"
  }
}
```

---

31. Link OAuth Provider

Purpose: Link a new OAuth provider to existing account

Endpoint: POST `{base_url}/auth/linked-accounts/oauth/link`

Access Level: □□ Protected

Authentication: Bearer Token

Request JSON Sample:

```
{
  "provider": "APPLE",
  "code": "authorization_code_from_apple",
  "redirectUri": "https://app.nextgate.com/auth/callback"
}
```

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "OAuth provider linked",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "provider": "APPLE",
    "email": "john.doe@icloud.com",
    "linkedAt": "2025-01-24T10:30:45",
    "message": "Apple account linked successfully"
  }
}
```


---

32. Unlink OAuth Provider

Purpose: Remove an OAuth provider from the account

Endpoint: DELETE `{base_url}/auth/linked-accounts/oauth/{provider}`

Access Level: □□ Protected

Authentication: Bearer Token

Path Parameters:

Parameter	Type	Required	Description
provider	string	Yes	`GOOGLE` or `APPLE`

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "OAuth provider unlinked",
  "action_time": "2025-01-24T10:30:45",
  "data": {
    "provider": "APPLE",
    "message": "Apple account unlinked"
  }
}
```

Error Response (Last Login Method):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Cannot unlink: This is your only login method.",
  "action_time": "2025-01-24T10:30:45",
  "data": "Cannot unlink: This is your only login method."
}
```

Error Reference

Standard Error Codes

HTTP Code	Status	Common Causes
400	BAD_REQUEST	Invalid request data, item already exists
401	UNAUTHORIZED	Missing/invalid/expired token
403	FORBIDDEN	Invalid OTP, max attempts exceeded, blocked
404	NOT_FOUND	Resource doesn't exist
422	UNPROCESSABLE_ENTITY	Validation errors
429	TOO_MANY_REQUESTS	Rate limit exceeded
500	INTERNAL_SERVER_ERROR	Server error

Authentication-Specific Errors

Error Message	HTTP Code	Resolution
"Token has expired"	401	Refresh token or re-authenticate
"Invalid OTP code"	403	Re-enter correct code
"Maximum verification attempts exceeded"	403	Request new OTP
"This phone is blocked"	400	Contact support

Error Message	HTTP Code	Resolution
"Account not found"	404	Check identifier
"Security alert: Token reuse detected"	401	Login again
"Login blocked: Too many failed attempts"	400	Wait or reset password
"You must be at least 13 years old"	N/A	Cannot use service

Frontend Integration Checklist

Before Starting

- ☐ Install FingerprintJS: `npm install @fingerprintjs/fingerprintjs`
- ☐ Set up secure token storage (httpOnly cookies or secure storage)
- ☐ Configure API base URL

Authentication Flow

- ☐ Implement device fingerprint generation on app load
- ☐ Handle all response types from `/auth/start`
- ☐ Implement OTP input with 6-digit validation
- ☐ Handle device verification flow
- ☐ Store tokens securely after successful auth

Onboarding Flow

- ☐ Pre-fill forms using `nextStepMetadata` when available
- ☐ Display username suggestions as clickable chips
- ☐ Implement real-time username availability check (debounced)
- ☐ Load interest categories from API
- ☐ Require minimum 3 interests
- ☐ Handle profile picture upload

Token Management

- ☐ Implement automatic token refresh before expiry
- ☐ Handle 401 responses globally (redirect to login)
- ☐ Store new refresh token after each rotation
- ☐ Clear all tokens on logout

Security Best Practices

1. **Never store tokens in localStorage** - Use httpOnly cookies or secure native storage
2. **Always send device fingerprint** - Required for device trust tracking
3. **Handle token rotation** - Always save new refresh token after refresh
4. **Validate OTP client-side** - Only allow 6 digits before API call
5. **Rate limit on frontend** - Disable resend button during cooldown
6. **Clear tokens on security events** - Token reuse detection, password reset

---

End of Documentation

Revision #11
Created 23 September 2025 06:35:10 by Admin Qbit
Updated 3 April 2026 10:38:53 by Admin Qbit