User Account

- Account Mng

Account Mng

Account API Documentation

Author: Josh S. Sakweli, Backend Lead Team
Last Updated: 2026-05-19
Version: v1.0

Base URL: `{base_url}/api/v1/account`

Short Description: The Account API manages everything related to a user's identity, credentials, and security. This includes password management, two-factor authentication, email verification, session control, trusted devices, and linking external auth providers. All operations in this API are private — only the authenticated account owner can act on their own account.

Why Account and Profile Are Separate:

> **Account** = who you *are* in the system (credentials, security, identity verification). **Profile** = how you *appear* to others (display name, bio, location, picture).
>
> This separation exists for a clear reason: account operations carry security implications and are always private — only you can change your password, revoke a session, or deactivate your account. Profile operations are social and can be partially visible to others. Mixing them creates confusion about what is private vs public, and makes it harder to apply the right access control. The Account API is your security dashboard; the Profile API is your public face.

Hints:

- All endpoints except `GET /username/check`, `GET /username/search`, and `GET /username/{username}` require a valid Bearer JWT token
- `X-Session-Id` header (UUID) is required for session-specific operations
- Username checks are case-insensitive — all usernames are lowercased automatically
- OTP codes expire in 10 minutes unless otherwise stated
- User-selectable OTP channels are: `SMS`, `WHATSAPP`, `EMAIL`, `SMS_AND_WHATSAPP`

Standard Response Format

Success Response Structure

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Operation completed successfully",
  "action_time": "2025-09-23T10:30:45",
  "data": {}
}
```

Error Response Structure

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Error description",
  "action_time": "2025-09-23T10:30:45",
  "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 describing the result
`action_time`	string	ISO 8601 timestamp of when the response was generated
`data`	object/string	Response payload on success, error details on failure

Standard Error Types

- `400 BAD_REQUEST`: Invalid request data, business rule violation, or item already exists
- `401 UNAUTHORIZED`: Missing, expired, or invalid JWT token
- `403 FORBIDDEN`: Verification failure (wrong password, wrong OTP)
- `404 NOT_FOUND`: Account or resource not found
- `422 UNPROCESSABLE_ENTITY`: Validation errors with field-level detail
- `500 INTERNAL_SERVER_ERROR`: Unexpected server error

Security

1. Get Security Info

Purpose: Returns the authenticated user's account security overview including verification status, 2FA state, and a computed security strength score.

Endpoint: `GET` `{base_url}/api/v1/account/security-info`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Security information retrieved successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "isEmailVerified": true,
    "isPhoneVerified": true,
    "isTwoFactorEnabled": false,
    "isAccountLocked": false,
    "lastPasswordChange": "2026-04-10T08:22:00",
    "accountCreatedAt": "2025-12-01T14:00:00",
```

```
    "roles": ["ROLE_USER"],
    "securityStrength": {
      "score": 50,
      "level": "MEDIUM",
      "description": "Your account security is good but can be improved",
      "recommendations": ["Enable two-factor authentication"]
    }
  }
}
```

Success Response Fields:

Field	Description
`isEmailVerified`	Whether the account's email has been verified
`isPhoneVerified`	Whether the account's phone number has been verified
`isTwoFactorEnabled`	Whether 2FA is currently active
`isAccountLocked`	Whether the account is locked or deactivated
`lastPasswordChange`	ISO 8601 timestamp of the last account edit
`accountCreatedAt`	ISO 8601 timestamp of account creation
`roles`	Array of role strings assigned to the account
`securityStrength.score`	Integer 0–100 representing security health
`securityStrength.level`	`VERY_WEAK`, `WEAK`, `MEDIUM`, or `STRONG`
`securityStrength.description`	Human-readable summary of the score
`securityStrength.recommendations`	List of actionable steps to improve score

Error Response JSON Sample:

```
{
  "success": false,
  "httpStatus": "UNAUTHORIZED",
  "message": "Token has expired",
  "action_time": "2026-05-19T10:30:45",
  "data": "Token has expired"
}
```

---

2. Enable Two-Factor Authentication

Purpose: Enables 2FA on the authenticated user's account. Requires password confirmation as a security gate.

Endpoint: `POST` `{base_url}/api/v1/account/2fa/enable`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:

```
{
  "password": "MyStr0ngP@ss!"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`password`	string	Yes	Current account password	Not blank

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Two-factor authentication enabled successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": null
}
```

Error Response JSON Sample:

```
{
  "success": false,
  "httpStatus": "FORBIDDEN",
  "message": "Password is incorrect",
```

```
  "action_time": "2026-05-19T10:30:45",
  "data": "Password is incorrect"
}
```

---

3. Disable Two-Factor Authentication

Purpose: Disables 2FA on the authenticated user's account. Requires password confirmation.

Endpoint: `POST` `{base_url}/api/v1/account/2fa/disable`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:

```
{
  "password": "MyStr0ngP@ss!"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`password`	string	Yes	Current account password	Not blank

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Two-factor authentication disabled successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": null
```

```
}
```

Error Response JSON Sample:

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Two-factor authentication is not enabled",
  "action_time": "2026-05-19T10:30:45",
  "data": "Two-factor authentication is not enabled"
}
```

---

4. Deactivate Account

Purpose: Locks the authenticated user's account (soft deactivation). The account is not deleted — data is preserved and support can reactivate it. Requires password confirmation.

Endpoint: **DELETE** `{base_url}/api/v1/account/deactivate`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:

```
{
  "password": "MyStr0ngP@ss!",
  "reason": "Taking a break from the platform"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation

`password`	string	Yes	Current account password	Not blank
`reason`	string	No	Optional reason for deactivation	Max 500 characters

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Account deactivated successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": null
}
```

Error Response JSON Sample:

```
{
  "success": false,
  "httpStatus": "FORBIDDEN",
  "message": "Password is incorrect",
  "action_time": "2026-05-19T10:30:45",
  "data": "Password is incorrect"
}
```

Password Management

5. Change Password

Purpose: Changes the account password. Supports two paths: provide `currentPassword` if the account already has one, or provide `otp` for passwordless accounts or if the user forgot their current password.

Endpoint: **POST** `{base_url}/api/v1/account/password/change`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:

```
{
  "currentPassword": "OldP@ss123!",
  "newPassword": "NewStr0ng@Pass!",
  "confirmPassword": "NewStr0ng@Pass!"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`currentPassword`	string	Conditional	Required if account has a password	One of `currentPassword` or `otp` must be provided
`otp`	string	Conditional	OTP alternative for passwordless accounts	One of `currentPassword` or `otp` must be provided
`newPassword`	string	Yes	The new password	Min 8 characters, not blank
`confirmPassword`	string	Yes	Must match `newPassword`	Not blank

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Password changed",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "success": true,
    "hadPassword": true,
    "message": "Password changed successfully"
  }
```

```
}
```

Success Response Fields:

Field	Description
`success`	Always `true` on success
`hadPassword`	`true` if this was a change, `false` if it was a first-time set
`message`	Human-readable confirmation

Error Response JSON Sample:

```
{
  "success": false,
  "httpStatus": "UNPROCESSABLE_ENTITY",
  "message": "Validation failed",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "newPassword": "Password must be at least 8 characters"
  }
}
```

---

6. Send OTP for Password Change

Purpose: Sends an OTP to the user's verified phone/email to enable password change without knowing the current password. Returns a temp token for the verification step.

Endpoint: POST `{base_url}/api/v1/account/password/change-with-otp/send-otp`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:

```
{
  "channel": "SMS"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`channel`	string	Yes	Where to send the OTP	enum: `SMS`, `WHATSAPP`, `EMAIL`, `SMS_AND_WHATSAPP`

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "OTP sent",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "tempToken": "eyJhbGciOiJIUzI1NiJ9...",
    "maskedValue": "+254***7890",
    "expiresIn": 600
  }
}
```

Success Response Fields:

Field	Description
`tempToken`	Short-lived token to submit in the verify step
`maskedValue`	Masked destination (phone or email) the OTP was sent to
`expiresIn`	Seconds until the OTP expires

---

7. Change Password with OTP

Purpose: Completes the OTP-based password change flow using the temp token and OTP from the previous step.

Endpoint: POST `{base_url}/api/v1/account/password/change-with-otp/verify`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:

```
{
  "tempToken": "eyJhbGciOiJIUzI1NiJ9...",
  "otp": "482910",
  "newPassword": "NewStr0ng@Pass!",
  "confirmPassword": "NewStr0ng@Pass!"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`tempToken`	string	Yes	Token from the send-otp step	Not blank
`otp`	string	Yes	6-digit OTP received	Exactly 6 characters
`newPassword`	string	Yes	New password	Min 8 characters, not blank
`confirmPassword`	string	Yes	Must match `newPassword`	Not blank

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Password changed successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "success": true,
    "hadPassword": true,
    "message": "Password changed successfully"
  }
}
```

8. Check If Password Can Be Set

Purpose: Checks whether the authenticated user is eligible to set a password. Useful for passwordless accounts (OAuth-only) who want to add a password.

Endpoint: `GET` `{base_url}/api/v1/account/password/can-set`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "You can set a password",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "canSetPassword": true,
    "authProvider": "GOOGLE"
  }
}
```

Success Response Fields:

Field	Description
`canSetPassword`	`true` if the account does not yet have a password
`authProvider`	The current primary auth provider (EMAIL, GOOGLE, APPLE, etc.)

9. Set Password (First Time)

Purpose: Sets a password for the first time on an account that currently has no password (e.g., a Google OAuth account). Use `/password/change` for subsequent changes.

Endpoint: POST `{base_url}/api/v1/account/password/set`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:

```
{
  "newPassword": "MyStr0ngP@ss!",
  "confirmPassword": "MyStr0ngP@ss!"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`newPassword`	string	Yes	Password to set	Min 8 characters, not blank
`confirmPassword`	string	Yes	Must match `newPassword`	Not blank

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Password set successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "success": true,
    "hadPassword": false,
    "message": "Password set successfully"
  }
}
```

Error Response JSON Sample:

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Password already set",
  "action_time": "2026-05-19T10:30:45",
  "data": "Password already set"
}
```

Username Management

10. Check Username Availability

Purpose: Checks whether a given username is valid and available. Public endpoint — no auth required. Returns alternative suggestions when the username is taken.

Endpoint: `GET` `{base_url}/api/v1/account/username/check`

Access Level: □□ Public

Authentication: None

Query Parameters:

Parameter	Type	Required	Description	Validation
`username`	string	Yes	Username to check	Auto-lowercased

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Username available",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "username": "john_doe",
    "available": true,
    "suggestions": null
```

```
  }
}
```

When Unavailable:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Username not available",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "username": "john_doe",
    "available": false,
    "suggestions": ["john_doe1", "john_doe99", "johndoe_"]
  }
}
```

Success Response Fields:

Field	Description
`username`	The lowercased username that was checked
`available`	`true` if the username is both valid and not taken
`suggestions`	List of alternative usernames (only present when `available` is `false`)

---

11. Check If Username Can Be Changed

Purpose: Checks whether the authenticated user is currently allowed to change their username (based on change frequency limits).

Endpoint: `GET` `{base_url}/api/v1/account/username/can-change`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "You can change your username",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "canChange": true,
    "currentUsername": "john_doe"
  }
}
```

Success Response Fields:

Field	Description
`canChange`	`true` if the user is allowed to change username now
`currentUsername`	The user's current public username

---

12. Change Username

Purpose: Changes the authenticated user's public username. Subject to rate/frequency limits — check `/can-change` first. The change is recorded in the username history.

Endpoint: POST `{base_url}/api/v1/account/username/change`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:

```
{
  "username": "new_username"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`username`	string	Yes	Desired new username	Auto-lowercased; must be valid and available

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Username changed successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "oldUsername": "john_doe",
    "newUsername": "new_username"
  }
}
```

Success Response Fields:

Field	Description
`oldUsername`	The previous username
`newUsername`	The newly assigned username

Error Response JSON Sample:

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Username change limit reached",
  "action_time": "2026-05-19T10:30:45",
  "data": "Username change limit reached"
}
```

---

13. Search Users

Purpose: Full-text search for accounts by username or display name. Public endpoint. Supports pagination.

Endpoint: `GET` `{base_url}/api/v1/account/username/search`

Access Level: Public

Authentication: None

Query Parameters:

Parameter	Type	Required	Description	Validation	Default
`q`	string	Yes	Search query (supports `@username` format)	Min 2 characters	—
`page`	integer	No	Zero-based page number	Min: 0	0
`size`	integer	No	Results per page	Max: 20	4

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Users found",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "users": [
      {
        "id": "550e8400-e29b-41d4-a716-446655440000",
        "userName": "john_doe",
        "displayName": "John Doe",
        "avatarUrl": "https://cdn.example.com/avatar.jpg"
      }
    ],
    "totalCount": 1,
    "hasMore": false
  }
}
```

Success Response Fields:

Field	Description
`users`	Array of matching user objects
`users[].id`	Account UUID
`users[].userName`	Public username
`users[].displayName`	Display name (may be null)
`users[].avatarUrl`	First profile picture URL (may be null)
`totalCount`	Total number of matching results
`hasMore`	`true` if more pages exist

---

14. Get User by Username

Purpose: Looks up a minimal user card by exact username. Returns only public-facing identity fields. Only works for active, non-locked accounts that have completed primary onboarding.

Endpoint: `GET` `{base_url}/api/v1/account/username/{username}`

Access Level: □□ Public

Authentication: None

Path Parameters:

Parameter	Type	Required	Description	Validation
`username`	string	Yes	Username to look up (supports `@username` format)	Auto-lowercased

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "User found",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "userName": "john_doe",
```

```
    "displayName": "John Doe",
    "avatarUrl": "https://cdn.example.com/avatar.jpg"
  }
}
```

Error Response JSON Sample:

```
{
  "success": false,
  "httpStatus": "NOT_FOUND",
  "message": "User not found",
  "action_time": "2026-05-19T10:30:45",
  "data": "User not found"
}
```

Sessions

15. Get Active Sessions

Purpose: Returns all active sessions for the authenticated account. Optionally marks the caller's current session when `X-Session-Id` is provided.

Endpoint: `GET` `{base_url}/api/v1/account/sessions`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`X-Session-Id`	string (UUID)	No	Current session UUID to mark it in the response

Success Response JSON Sample:

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

```
  "httpStatus": "OK",
  "message": "Sessions retrieved",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "sessions": [
      {
        "id": "123e4567-e89b-12d3-a456-426614174000",
        "deviceId": "device_abc123",
        "deviceName": "iPhone 15",
        "platform": "IOS",
        "ipAddress": "197.248.1.1",
        "location": "Nairobi, KE",
        "lastActiveAt": "2026-05-19T09:00:00",
        "createdAt": "2026-04-01T12:00:00",
        "currentSession": true
      }
    ],
    "totalCount": 1,
    "currentSession": { ... }
  }
}
```


Success Response Fields:

Field	Description
`sessions`	Array of all active session objects
`sessions[].id`	Session UUID
`sessions[].deviceName`	Human-readable device name
`sessions[].platform`	Device platform (IOS, ANDROID, WEB, etc.)
`sessions[].ipAddress`	IP address the session was created from
`sessions[].location`	Approximate geographic location
`sessions[].lastActiveAt`	When this session last made a request
`sessions[].currentSession`	`true` if this is the caller's current session
`totalCount`	Total number of active sessions
`currentSession`	The caller's current session object (if `X-Session-Id` provided)

---

16. Sign Out (Current Session)

Purpose: Invalidates the session identified by the `X-Session-Id` header.

Endpoint: POST `{base_url}/api/v1/account/sessions/sign-out`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`X-Session-Id`	string (UUID)	Yes	The session to invalidate

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Signed out successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": null
}
```

---

17. Sign Out Other Devices

Purpose: Invalidates all sessions except the current one. Requires password or OTP confirmation as a security gate.

Endpoint: POST `{base_url}/api/v1/account/sessions/sign-out-others`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description

`Authorization`	string	Yes	`Bearer <jwt_token>`
`X-Session-Id`	string (UUID)	Yes	The session to KEEP (current session)
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:

```
{
  "password": "MyStr0ngP@ss!"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`password`	string	Conditional	Account password	One of `password` or `otp` required
`otp`	string	Conditional	OTP alternative	One of `password` or `otp` required

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Signed out of other devices",
  "action_time": "2026-05-19T10:30:45",
  "data": null
}
```

---

18. Sign Out All Devices

Purpose: Invalidates all active sessions including the current one. Requires password or OTP confirmation.

Endpoint: **POST** `{base_url}/api/v1/account/sessions/sign-out-all`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:

```
{
  "password": "MyStr0ngP@ss!"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`password`	string	Conditional	Account password	One of `password` or `otp` required
`otp`	string	Conditional	OTP alternative	One of `password` or `otp` required

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Signed out of all devices",
  "action_time": "2026-05-19T10:30:45",
  "data": null
}
```

---

19. Revoke Specific Session

Purpose: Revokes a specific session by its ID. Used to forcefully end a session from another device.

Endpoint: DELETE `{base_url}/api/v1/account/sessions/{sessionId}`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`

Path Parameters:

Parameter	Type	Required	Description	Validation
`sessionId`	UUID	Yes	ID of the session to revoke	Valid UUID format

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Session revoked",
  "action_time": "2026-05-19T10:30:45",
  "data": null
}
```

Devices

20. Verify Device

Purpose: Completes new device verification during login. When a login attempt is made from an unrecognized device, this endpoint finalizes the flow after the user confirms ownership with an OTP. Returns full auth tokens on success.

Endpoint: POST `{base_url}/api/v1/account/device/verify`

Access Level: □□ Public (No JWT — called before full auth is established)

Authentication: None

Request Headers:

Header	Type	Required	Description
`User-Agent`	string	Yes	Device user-agent (auto-sent by clients)

Request JSON Sample:

```
{
  "deviceVerificationToken": "dvt_abc123...",
  "otp": "482910",
  "deviceId": "unique-device-fingerprint-id",
  "deviceName": "iPhone 15 Pro",
  "platform": "IOS"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`deviceVerificationToken`	string	Yes	Token issued during the login challenge step	Not blank
`otp`	string	Yes	6-digit OTP sent to user for device verification	Exactly 6 numeric digits
`deviceId`	string	Yes	Unique client-side device fingerprint	Not blank
`deviceName`	string	No	Human-readable device name	—
`platform`	string	No	Device platform (IOS, ANDROID, WEB, etc.)	—

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Device verified. Login successful.",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiJ9...",
    "onboarding": {
      "isPrimaryComplete": true,
      "hasUsername": true
    },
    "user": {
      "id": "550e8400-e29b-41d4-a716-446655440000",
      "userName": "john_doe",
```

```
      "firstName": "John"
    }
  }
}
```

Success Response Fields:

Field	Description
`accessToken`	JWT access token for subsequent API calls
`refreshToken`	JWT refresh token for getting new access tokens
`onboarding`	Flags indicating what onboarding steps remain
`user`	Basic user info object

---

21. Get Trusted Devices

Purpose: Returns the list of devices that have been verified and trusted on this account.

Endpoint: `GET` `{base_url}/api/v1/account/devices`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`X-Device-Id`	string	No	Current device ID to mark it in the response

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Devices retrieved",
  "action_time": "2026-05-19T10:30:45",
  "data": [
    {
```

```
      "id": "550e8400-e29b-41d4-a716-446655440000",
      "deviceId": "unique-device-fingerprint-id",
      "deviceName": "iPhone 15 Pro",
      "platform": "IOS",
      "lastUsedAt": "2026-05-19T09:00:00",
      "createdAt": "2026-04-01T12:00:00"
    }
  ]
}
```


---

22. Revoke Device

Purpose: Removes a trusted device. The next login from that device will require device verification again.

Endpoint: DELETE `{base_url}/api/v1/account/devices/{deviceKeyId}`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`

Path Parameters:

Parameter	Type	Required	Description	Validation
`deviceKeyId`	UUID	Yes	ID of the device key to revoke	Valid UUID format

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Device revoked",
  "action_time": "2026-05-19T10:30:45",
  "data": null
```

```
}
```

Account Linking

23. Initiate Email Link

Purpose: Starts the process of linking an email address to the account. Sends a verification OTP to the provided email and returns a temp token for the verify step.

Endpoint: `POST` `{base_url}/api/v1/account/link/email/initiate`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:

```
{
  "email": "john@example.com"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`email`	string	Yes	Email address to link	Valid email format, not blank

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Verification code sent",
```

```
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "tempToken": "eyJhbGciOiJIUzI1NiJ9...",
    "maskedValue": "jo***@example.com",
    "expiresIn": 600,
    "linked": false,
    "message": "Verification code sent to jo***@example.com"
  }
}
```


Success Response Fields:

Field	Description
`tempToken`	Short-lived token to submit in the verify step
`maskedValue`	Masked email the OTP was sent to
`expiresIn`	Seconds until the OTP expires

---

24. Verify and Link Email

Purpose: Completes the email linking process by submitting the OTP and temp token from the initiate step.

Endpoint: POST `{base_url}/api/v1/account/link/email/verify`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:


```
{
  "tempToken": "eyJhbGciOiJIUzI1NiJ9...",
  "otp": "482910"
```

```
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`tempToken`	string	Yes	Token from the initiate step	Not blank
`otp`	string	Yes	6-digit OTP sent to the email	Exactly 6 characters

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Email linked successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "linked": true,
    "message": "Email linked successfully"
  }
}
```

---

25. Unlink Email

Purpose: Removes the linked email from the account.

Endpoint: DELETE `{base_url}/api/v1/account/link/email`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Email unlinked",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "unlinked": true,
    "message": "Email unlinked successfully"
  }
}
```


---

26. Link OAuth Provider

Purpose: Links a Google or Apple account to the authenticated account. Useful for adding a social login option to an email/phone account.

Endpoint: `POST` `{base_url}/api/v1/account/link/oauth`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`
`Content-Type`	string	Yes	`application/json`

Request JSON Sample:


```
{
  "provider": "GOOGLE",
  "idToken": "ya29.a0AfH6SMC..."
}
```


Request Body Parameters:

Parameter	Type	Required	Description	Validation
`provider`	string	Yes	OAuth provider to link	enum: `GOOGLE`, `APPLE`

Parameter	Type	Required	Description	Validation
`idToken`	string	Yes	ID token from the provider's SDK	Not blank

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "GOOGLE linked successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "linked": true,
    "message": "GOOGLE linked successfully"
  }
}
```

---

27. Unlink OAuth Provider

Purpose: Removes a linked OAuth provider from the account.

Endpoint: DELETE `{base_url}/api/v1/account/link/oauth/{provider}`

Access Level: □□ Protected (Requires valid JWT)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <jwt_token>`

Path Parameters:

Parameter	Type	Required	Description	Validation
`provider`	string	Yes	OAuth provider to unlink	`GOOGLE` or `APPLE` (case-insensitive)

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "GOOGLE unlinked",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "unlinked": true,
    "message": "GOOGLE unlinked successfully"
  }
}
```


Error Response JSON Sample:


```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Cannot unlink last auth provider",
  "action_time": "2026-05-19T10:30:45",
  "data": "Cannot unlink last auth provider"
}
```

Quick Reference — All Account Endpoints

#	Method	Endpoint	Auth	Description
1	GET	`/account/security-info`	◫◫	Get security overview
2	POST	`/account/2fa/enable`	◫◫	Enable 2FA
3	POST	`/account/2fa/disable`	◫◫	Disable 2FA
4	DELETE	`/account/deactivate`	◫◫	Soft-deactivate account
5	POST	`/account/password/change`	◫◫	Change password
6	POST	`/account/password/change-with-otp/send-otp`	◫◫	Request OTP for password change
7	POST	`/account/password/change-with-otp/verify`	◫◫	Change password via OTP

#	Method	Endpoint	Auth	Description
8	GET	`/account/password/can-set`	◫	Check if password can be set
9	POST	`/account/password/set`	◫	Set password (first time)
10	GET	`/account/username/check`	◫	Check username availability
11	GET	`/account/username/can-change`	◫	Check username change eligibility
12	POST	`/account/username/change`	◫	Change username
13	GET	`/account/username/search`	◫	Search users
14	GET	`/account/username/{username}`	◫	Lookup user by username
15	GET	`/account/sessions`	◫	List active sessions
16	POST	`/account/sessions/sign-out`	◫	Sign out current session
17	POST	`/account/sessions/sign-out-others`	◫	Sign out all other sessions
18	POST	`/account/sessions/sign-out-all`	◫	Sign out all sessions
19	DELETE	`/account/sessions/{sessionId}`	◫	Revoke specific session
20	POST	`/account/device/verify`	◫	Verify new device
21	GET	`/account/devices`	◫	List trusted devices
22	DELETE	`/account/devices/{deviceKeyId}`	◫	Revoke trusted device
23	POST	`/account/link/email/initiate`	◫	Start email linking
24	POST	`/account/link/email/verify`	◫	Complete email linking
25	DELETE	`/account/link/email`	◫	Unlink email
26	POST	`/account/link/oauth`	◫	Link OAuth provider
27	DELETE	`/account/link/oauth/{provider}`	◫	Unlink OAuth provider