User-profile-nexgate-service(5)

- User Profile Management Service

User Profile Management Service

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

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

Short Description: The Profile API manages a user's social presence — what others see when they visit your page. This includes viewing your own full profile, looking up other users by ID or username, and updating your display information such as name, bio, location, and profile pictures.

Why Account and Profile Are Separate:

> **Profile** = how you *appear* to others (display name, bio, location, picture, social stats). **Account** = who you *are* in the system (credentials, security, identity verification).
>
> Profile operations are social in nature — some endpoints (`/id/{userId}`, `/u/{username}`) are intentionally viewable by any authenticated user, not just the owner. Account operations carry security implications and are always strictly private. By separating the two, we can apply the right access rules to each: profile data can be shared, account data cannot. Anything that touches credentials (password, 2FA, email, phone) belongs in the Account API, not here.

Hints:

- `GET /me` requires authentication; it returns private fields like `email`, `phoneNumber`, and `securityStrength` only visible to the account owner
- `GET /id/{userId}` and `GET /u/{username}` work for both authenticated and unauthenticated callers — authenticated callers additionally receive `relationship` info (isFollowing, isBlocked, etc.)
- `PUT /update` only accepts social/display fields — to change username use `POST /api/v1/account/username/change`; to change email or phone use the Account Linking API
- Profile picture URLs must be publicly accessible `https://` image URLs ending in `.jpg`, `.jpeg`, `.png`, `.gif`, or `.webp`
- Maximum 5 profile picture URLs per account

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 or business rule violation
- `401 UNAUTHORIZED`: Missing, expired, or invalid JWT token
- `404 NOT_FOUND`: User not found
- `422 UNPROCESSABLE_ENTITY`: Validation errors with field-level detail
- `500 INTERNAL_SERVER_ERROR`: Unexpected server error

Endpoints

1. Get My Profile

Purpose: Returns the full profile of the currently authenticated user, including private fields (email, phone, verification status, security strength) and social stats (followers, posts, shops, events).

Endpoint: `GET` `{base_url}/api/v1/profile/me`

Access Level: □□ Protected (Requires valid JWT — own profile only)

Authentication: Bearer Token

Request Headers:

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

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Profile retrieved successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "userName": "john_doe",
    "firstName": "John",
    "lastName": "Doe",
    "middleName": null,
    "email": "john@example.com",
    "phoneNumber": "+254712345678",
```

```
    "bio": "Building cool things.",
    "location": "Nairobi, Kenya",
    "profilePictureUrls": [
      "https://cdn.example.com/profiles/john_main.jpg"
    ],
    "isVerified": true,
    "isEmailVerified": true,
    "isPhoneVerified": true,
    "isTwoFactorEnabled": false,
    "isAccountLocked": false,
    "followersCount": 1420,
    "followingCount": 310,
    "postsCount": 42,
    "shopsCount": 1,
    "eventsCount": 3,
    "createdAt": "2025-12-01T14:00:00",
    "editedAt": "2026-05-10T08:22:00",
    "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
`id`	Account UUID
`userName`	Public username
`firstName`	First name
`lastName`	Last name
`middleName`	Middle name (may be null)
`email`	Linked email address (private — only visible to owner)
`phoneNumber`	Linked phone number (private — only visible to owner)
`bio`	Profile bio text

Field	Description
`location`	Location string
`profilePictureUrls`	Array of profile picture URLs (first is primary)
`isVerified`	Whether the account has been verified
`isEmailVerified`	Whether the email has been verified
`isPhoneVerified`	Whether the phone has been verified
`isTwoFactorEnabled`	Whether 2FA is active
`isAccountLocked`	Whether the account is deactivated or locked
`followersCount`	Number of accepted followers
`followingCount`	Number of accounts the user is following
`postsCount`	Number of published posts
`shopsCount`	Number of active shops
`eventsCount`	Number of published events
`createdAt`	ISO 8601 account creation timestamp
`editedAt`	ISO 8601 last profile update timestamp
`roles`	Array of assigned role strings
`securityStrength.score`	Security health score 0–100
`securityStrength.level`	`VERY_WEAK`, `WEAK`, `MEDIUM`, or `STRONG`
`securityStrength.recommendations`	Actionable steps to improve security 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. Get Profile by User ID

Purpose: Returns the public profile summary of any user by their UUID. Authenticated callers additionally receive relationship context (isFollowing, isBlocked, etc.).

Endpoint: `GET` `{base_url}/api/v1/profile/id/{userId}`

Access Level: □□ Public (Relationship info available when authenticated)

Authentication: Bearer Token (optional)

Request Headers:

Header	Type	Required	Description
`Authorization`	string	No	`Bearer <jwt_token>` — include to get relationship data

Path Parameters:

Parameter	Type	Required	Description	Validation
`userId`	UUID	Yes	The UUID of the user whose profile to fetch	Valid UUID format

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Profile retrieved successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "userId": "550e8400-e29b-41d4-a716-446655440000",
    "userName": "john_doe",
    "firstName": "John",
    "lastName": "Doe",
    "profilePictureUrl": "https://cdn.example.com/profiles/john_main.jpg",
    "bio": "Building cool things.",
    "location": "Nairobi, Kenya",
    "isVerified": true,
    "followersCount": 1420,
    "followingCount": 310,
    "postsCount": 42,
    "shopsCount": 1,
    "eventsCount": 3,
    "createdAt": "2025-12-01T14:00:00",
    "isOwnProfile": false,
```

```
    "relationship": {
      "isFollowing": true,
      "isFollowedBy": false,
      "isBlocked": false,
      "isBlockedBy": false
    }
  }
}
```

Success Response Fields:

Field	Description
`userId`	Account UUID
`userName`	Public username
`firstName`	First name
`lastName`	Last name
`profilePictureUrl`	Primary profile picture URL (first in the list, may be null)
`bio`	Profile bio
`location`	Location string
`isVerified`	Whether the account has been platform-verified
`followersCount`	Number of accepted followers
`followingCount`	Number of accounts the user follows
`postsCount`	Number of published posts
`shopsCount`	Number of active shops
`eventsCount`	Number of published events
`createdAt`	ISO 8601 account creation timestamp
`isOwnProfile`	`true` if the authenticated caller is viewing their own profile
`relationship`	Present only when the caller is authenticated and viewing someone else's profile
`relationship.isFollowing`	`true` if the caller follows this user
`relationship.isFollowedBy`	`true` if this user follows the caller
`relationship.isBlocked`	`true` if the caller has blocked this user
`relationship.isBlockedBy`	`true` if this user has blocked the caller

Error Response JSON Sample:

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


---

3. Get Profile by Username

Purpose: Returns the public profile summary of any user by their username. Authenticated callers additionally receive relationship context. Functionally identical to endpoint 2 but accessed via username instead of UUID.

Endpoint: `GET` `{base_url}/api/v1/profile/u/{username}`

Access Level: □□ Public (Relationship info available when authenticated)

Authentication: Bearer Token (optional)

Request Headers:

Header	Type	Required	Description
`Authorization`	string	No	`Bearer <jwt_token>` — include to get relationship data

Path Parameters:

Parameter	Type	Required	Description	Validation
`username`	string	Yes	The public username to look up	Supports `@username` format

Success Response JSON Sample:


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

```
    "userId": "550e8400-e29b-41d4-a716-446655440000",
    "userName": "john_doe",
    "firstName": "John",
    "lastName": "Doe",
    "profilePictureUrl": "https://cdn.example.com/profiles/john_main.jpg",
    "bio": "Building cool things.",
    "location": "Nairobi, Kenya",
    "isVerified": true,
    "followersCount": 1420,
    "followingCount": 310,
    "postsCount": 42,
    "shopsCount": 1,
    "eventsCount": 3,
    "createdAt": "2025-12-01T14:00:00",
    "isOwnProfile": false,
    "relationship": {
      "isFollowing": false,
      "isFollowedBy": false,
      "isBlocked": false,
      "isBlockedBy": false
    }
  }
}
```


Success Response Fields: Same as endpoint 2 — see Get Profile by User ID.

Error Response JSON Sample:


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


---

4. Update Profile

Purpose: Updates the authenticated user's social/display information. Only accepts social fields — username changes, email changes, and phone changes are handled by the Account API. All fields are optional; only fields provided will be updated.

Endpoint: **PUT** `{base_url}/api/v1/profile/update`

Access Level: □□ Protected (Requires valid JWT — own profile only)

Authentication: Bearer Token

Request Headers:

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

Request JSON Sample:

```
{
  "firstName": "John",
  "lastName": "Doe",
  "middleName": "K",
  "bio": "Building cool things one commit at a time.",
  "location": "Nairobi, Kenya",
  "profilePictureUrls": [
    "https://cdn.example.com/profiles/john_main.jpg",
    "https://cdn.example.com/profiles/john_alt.jpg"
  ]
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`firstName`	string	No	First name	1–30 characters
`lastName`	string	No	Last name	1–30 characters
`middleName`	string	No	Middle name	Max 30 characters
`bio`	string	No	Short bio shown on profile	No length constraint enforced here
`location`	string	No	Location string shown on profile	No length constraint enforced here

Parameter	Type	Required	Description	Validation
`profilePictureUrls`	array of strings	No	Ordered list of image URLs	Max 5 items; each URL must be a valid `https://` image URL ending in `.jpg`, `.jpeg`, `.png`, `.gif`, or `.webp`; max 500 characters per URL

> **Note**: To change your username, use `POST /api/v1/account/username/change`. To change your email or phone number, use the Account Linking API (`/api/v1/account/link`).

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Profile updated successfully",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "userName": "john_doe",
    "firstName": "John",
    "lastName": "Doe",
    "middleName": "K",
    "email": "john@example.com",
    "phoneNumber": "+254712345678",
    "bio": "Building cool things one commit at a time.",
    "location": "Nairobi, Kenya",
    "profilePictureUrls": [
      "https://cdn.example.com/profiles/john_main.jpg",
      "https://cdn.example.com/profiles/john_alt.jpg"
    ],
    "isVerified": true,
    "isEmailVerified": true,
    "isPhoneVerified": true,
    "isTwoFactorEnabled": false,
    "isAccountLocked": false,
    "createdAt": "2025-12-01T14:00:00",
```

```
    "editedAt": "2026-05-19T10:30:45",
    "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
`id`	Account UUID
`userName`	Public username (unchanged by this endpoint)
`firstName`	Updated first name
`lastName`	Updated last name
`middleName`	Updated middle name (may be null)
`email`	Current email (unchanged by this endpoint)
`phoneNumber`	Current phone number (unchanged by this endpoint)
`bio`	Updated bio
`location`	Updated location
`profilePictureUrls`	Updated list of profile picture URLs
`isVerified`	Account verification status
`isEmailVerified`	Email verification status
`isPhoneVerified`	Phone verification status
`isTwoFactorEnabled`	2FA status
`isAccountLocked`	Account lock status
`createdAt`	Account creation timestamp
`editedAt`	Updated timestamp reflecting this change
`roles`	Assigned role strings
`securityStrength`	Current security score and recommendations

Error Response JSON Sample:

```
{
  "success": false,
  "httpStatus": "UNPROCESSABLE_ENTITY",
  "message": "Validation failed",
  "action_time": "2026-05-19T10:30:45",
  "data": {
    "firstName": "First name must be between 1 and 30 characters",
    "profilePictureUrls": "Maximum 5 profile pictures allowed"
  }
}
```

Quick Reference — All Profile Endpoints

#	Method	Endpoint	Auth	Description
1	GET	`/profile/me`	◫ Required	Full profile — own account only
2	GET	`/profile/id/{userId}`	◫ Optional	Public profile by UUID
3	GET	`/profile/u/{username}`	◫ Optional	Public profile by username
4	PUT	`/profile/update`	◫ Required	Update display info

What Belongs in Profile vs Account

I want to...	Use
View my full profile	`GET /profile/me`
View someone else's profile	`GET /profile/u/{username}` or `GET /profile/id/{id}`
Update my name, bio, location, or picture	`PUT /profile/update`
Change my username	`POST /api/v1/account/username/change`
Change my email address	`POST /api/v1/account/link/email/initiate`
Change my phone number	Phone OTP verification flow
Change my password	`POST /api/v1/account/password/change`
Enable or disable 2FA	`POST /api/v1/account/2fa/enable` or `/disable`

I want to…	Use
Verify my email	`POST /api/v1/account/email/request-verification` → `/email/verify`
View my security status	`GET /api/v1/account/security-info`
Deactivate my account	`DELETE /api/v1/account/deactivate`