Follow System APIs

Author: Josh, Lead Backend Developer
Last Updated: 2025-12-20
Version: v1.1

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

Short Description: The Follow System API manages user relationships including following/unfollowing users, handling follow requests for private accounts, retrieving followers and following lists, and discovering featured users. It supports both immediate follows (public accounts) and pending follow requests (private accounts). The system intelligently tracks mutual relationships and provides contextual follow/unfollow buttons based on the authenticated user's relationship with other users.

Hints:

- All endpoints require authentication via Bearer token
- Following a private account creates a PENDING follow request that requires acceptance
- Following a public account creates an ACCEPTED follow immediately
- Featured users are sorted by follower count and exclude users you already follow
- Pagination is 1-indexed (page=1 for first page)
- Default page size is 20 items
- Follower/Following lists include relationship indicators (`followsYou`, `youFollow`) for smart button rendering

---

How The Follow System Works

Relationship Context Logic

The Follow System provides intelligent relationship tracking between the **authenticated user** (you) and **listed users** across different profile views:

1. Your Own Followers List (`/followers/{yourUserId}`)

- Shows users who follow you
- Each follower includes:
  - `followsYou`: Always `true` (they follow you)
  - `youFollow`: Whether you follow them back
- **Button Logic:**

- `youFollow: false` → Show **"Follow Back"** button
- `youFollow: true` → Show **"Following"** button

2. Your Own Following List (`/following/{yourUserId}`)

- Shows users you follow
- Each user includes:
  - `followsYou`: Whether they follow you back (mutual relationship)
  - `youFollow`: Always `true` (you follow them)
- **UI Logic:**
  - `followsYou: true` → Display **"Follows You"** badge
  - Always show **"Following"** button (can unfollow)

3. Friend's Followers List (`/followers/{friendUserId}`)

- Shows users who follow your friend
- Each follower includes:
  - `followsYou`: Whether this follower follows **you** (authenticated user)
  - `youFollow`: Whether **you** follow this follower
- **Button Logic:**
  - `followsYou: true` AND `youFollow: false` → Show **"Follow Back"**
  - `youFollow: true` → Show **"Following"**
  - Both `false` → Show **"Follow"**

4. Friend's Following List (`/following/{friendUserId}`)

- Shows users your friend follows
- Each user includes:
  - `followsYou`: Whether this user follows **you** (authenticated user)
  - `youFollow`: Whether **you** follow this user
- **Button Logic:**
  - `followsYou: true` AND `youFollow: false` → Show **"Follow Back"**
  - `youFollow: true` → Show **"Following"**
  - Both `false` → Show **"Follow"**

Privacy & Status Rules

Private Accounts:

- Follow requests enter **PENDING** status
- Require manual acceptance by the account owner
- Can be accepted or declined

Public Accounts:

- Follows are immediately **ACCEPTED**

- No manual approval needed

Blocking Behavior:

- Blocking a user removes all follow relationships (bidirectional)
- Blocked users cannot follow you or see your content

Key Principle: All relationship indicators (`followsYou`, `youFollow`) are always calculated relative to the **authenticated user**, regardless of whose profile is being viewed.

UI Implementation Best Practices

Button State Management

The follow system uses a **single button** that changes state based on the relationship context. This follows industry standards (Twitter/X, Instagram, LinkedIn).

Button Logic by Relationship State:

```
if (youFollow) {
  // Button: "Following"
  // Action: Click to unfollow (with confirmation)
  // Style: Secondary/outlined button
} else if (followsYou) {
  // Button: "Follow Back"
  // Action: Click to follow immediately (no confirmation)
  // Style: Primary/solid button
} else {
  // Button: "Follow"
  // Action: Click to follow immediately (no confirmation)
  // Style: Primary/solid button
}
```

Why Single Button Design:

 Advantages:

- Clean, uncluttered interface
- Clear current state indication
- Industry-standard pattern
- Saves screen space

- Intuitive user experience

 Avoid:

- Separate "Unfollow" button (clutters UI)
- Multiple buttons for same relationship
- Ambiguous action labels

Confirmation Dialog Best Practices

Show confirmation ONLY for destructive actions:

Unfollow Action (Destructive):

```
User clicks "Following" button
↓
Show confirmation modal:
"Unfollow @username?"
[Cancel] [Unfollow]
↓
On confirm: Call DELETE /e-social/follows/{userId}
```

Follow/Follow Back Actions (Constructive):

```
User clicks "Follow" or "Follow Back" button
↓
Immediately call POST /e-social/follows/{userId}
No confirmation needed
```

Rationale:

- Unfollowing removes an existing connection (requires confirmation)
- Following creates a new connection (no barrier needed)
- Matches user expectations from major platforms

Visual States & Feedback

Button States:

State	Label	Style	On Click
Not Following	"Follow"	Primary/Solid	Follow immediately
Mutual Follow Possible	"Follow Back"	Primary/Solid	Follow immediately

State	Label	Style	On Click
Currently Following	"Following"	Secondary/Outlined	Show unfollow confirmation
Pending Request	"Requested"	Secondary/Disabled	Cancel request (optional)

Loading States:

- Show spinner on button during API call
- Disable button to prevent double-clicks
- Optimistic UI update (change button immediately, rollback on error)

Error Handling:

- Show toast/snackbar on API errors
- Revert button state on failure
- Clear error messages (e.g., "Failed to follow user. Try again.")

Additional UI Elements

Badges:

- Show "Follows You" badge when `followsYou: true` in Following list
- Display badge near username, not on button
- Use subtle styling (gray text or small pill)

Private Account Indicators:

- Show lock icon for private accounts
- Update button to "Request" for private accounts
- After clicking, change to "Requested" state

Standard Response Format

All API responses follow a consistent structure using our Globe Response Builder pattern:

Success Response Structure

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

```
  "action_time": "2025-09-23T10:30:45",
  "data": {
    // Actual response data goes here
  }
}
```

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	Always `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 operation result
`action_time`	string	ISO 8601 timestamp of when the response was generated
`data`	object/string	Response payload for success, error details for failures

---

HTTP Method Badge Standards

For better visual clarity, all endpoints use colored badges for HTTP methods with the following standard colors:

- **GET** - Green (Safe, read-only operations)
- **POST** - Blue (Create new resources)
- **DELETE** - Red (Remove resources)

Endpoints

1. Follow User

Purpose: Follow a user or create a follow request if the target account is private

Endpoint: **POST** `{base_url}/e-social/follows/{userId}`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
userId	string	Yes	UUID of the user to follow	Must be valid UUID format

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "User followed successfully",
  "action_time": "2025-12-11T10:30:45",
  "data": {
    "id": "660e8400-e29b-41d4-a716-446655440011",
    "followerId": "123e4567-e89b-12d3-a456-426614174000",
    "followingId": "987e6543-e21b-12d3-a456-426614174999",
    "follower": {
      "id": "123e4567-e89b-12d3-a456-426614174000",
```

```
      "userName": "john_doe",
      "firstName": "John",
      "lastName": "Doe",
      "profilePictureUrls": ["https://cdn.nexgate.com/profiles/john_doe.jpg"],
      "isVerified": true
    },
    "following": {
      "id": "987e6543-e21b-12d3-a456-426614174999",
      "userName": "jane_smith",
      "firstName": "Jane",
      "lastName": "Smith",
      "profilePictureUrls": ["https://cdn.nexgate.com/profiles/jane_smith.jpg"],
      "isVerified": false
    },
    "status": "ACCEPTED",
    "createdAt": "2025-12-11T10:30:45"
  }
}
```


Success Response Fields:

Field	Description
id	Unique identifier for the follow relationship (UUID format)
followerId	UUID of the user who initiated the follow
followingId	UUID of the user being followed
follower	Object containing summary information about the follower user
follower.id	UUID of the follower user
follower.userName	Username of the follower
follower.firstName	First name of the follower
follower.lastName	Last name of the follower
follower.profilePictureUrls	Array of profile picture URLs for the follower
follower.isVerified	Boolean indicating if the follower account is verified
following	Object containing summary information about the user being followed
following.id	UUID of the user being followed
following.userName	Username of the user being followed
following.firstName	First name of the user being followed

Field	Description
following.lastName	Last name of the user being followed
following.profilePictureUrls	Array of profile picture URLs for the user being followed
following.isVerified	Boolean indicating if the followed account is verified
status	Follow status - "ACCEPTED" for public accounts, "PENDING" for private accounts
createdAt	Timestamp when the follow relationship was created (ISO 8601 format)

Error Response JSON Sample:

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Cannot follow yourself",
  "action_time": "2025-12-11T10:30:45",
  "data": "Cannot follow yourself"
}
```

Standard Error Types:

Application-Level Exceptions (400-499)

- `400 BAD_REQUEST`: Cannot follow yourself, already following this user, or invalid request data
- `401 UNAUTHORIZED`: Authentication issues (empty, invalid, expired, or malformed tokens)
- `404 NOT_FOUND`: User to follow not found
- `500 INTERNAL_SERVER_ERROR`: Unexpected server errors

Error Response Examples:

Bad Request - Self Follow (400):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Cannot follow yourself",
  "action_time": "2025-12-11T10:30:45",
  "data": "Cannot follow yourself"
}
```

Bad Request - Already Following (400):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Already following this user",
  "action_time": "2025-12-11T10:30:45",
  "data": "Already following this user"
}
```

Not Found (404):

```
{
  "success": false,
  "httpStatus": "NOT_FOUND",
  "message": "User not found",
  "action_time": "2025-12-11T10:30:45",
  "data": "User not found"
}
```

---

2. Unfollow User

Purpose: Remove a follow relationship with another user

Endpoint: **DELETE** `{base_url}/e-social/follows/{userId}`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation

userId	string	Yes	UUID of the user to unfollow	Must be valid UUID format

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "User unfollowed successfully",
  "action_time": "2025-12-11T10:30:45",
  "data": null
}
```

Success Response Fields:

Field	Description
data	Always null for this endpoint

Error Response JSON Sample:

```
{
  "success": false,
  "httpStatus": "NOT_FOUND",
  "message": "User not found",
  "action_time": "2025-12-11T10:30:45",
  "data": "User not found"
}
```

Standard Error Types:

Application-Level Exceptions (400-499)

- `401 UNAUTHORIZED`: Authentication issues (empty, invalid, expired, or malformed tokens)
- `404 NOT_FOUND`: User to unfollow not found
- `500 INTERNAL_SERVER_ERROR`: Unexpected server errors

Error Response Examples:

Not Found (404):

```
{
  "success": false,
```

```
  "httpStatus": "NOT_FOUND",
  "message": "User not found",
  "action_time": "2025-12-11T10:30:45",
  "data": "User not found"
}
```

---

3. Accept Follow Request

Purpose: Accept a pending follow request from another user

Endpoint: **POST** `{base_url}/e-social/follows/requests/{followId}/accept`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
followId	string	Yes	UUID of the follow request to accept	Must be valid UUID format

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Follow request accepted",
  "action_time": "2025-12-11T10:30:45",
  "data": {
    "id": "660e8400-e29b-41d4-a716-446655440011",
    "followerId": "987e6543-e21b-12d3-a456-426614174999",
```

```
    "followingId": "123e4567-e89b-12d3-a456-426614174000",
    "follower": {
      "id": "987e6543-e21b-12d3-a456-426614174999",
      "userName": "jane_smith",
      "firstName": "Jane",
      "lastName": "Smith",
      "profilePictureUrls": ["https://cdn.nexgate.com/profiles/jane_smith.jpg"],
      "isVerified": false
    },
    "following": {
      "id": "123e4567-e89b-12d3-a456-426614174000",
      "userName": "john_doe",
      "firstName": "John",
      "lastName": "Doe",
      "profilePictureUrls": ["https://cdn.nexgate.com/profiles/john_doe.jpg"],
      "isVerified": true
    },
    "status": "ACCEPTED",
    "createdAt": "2025-12-11T09:15:30"
  }
}
```


Standard Error Types:

Application-Level Exceptions (400-499)

- `400 BAD_REQUEST`: Follow request is not pending
- `401 UNAUTHORIZED`: Authentication issues
- `404 NOT_FOUND`: Follow request not found
- `500 INTERNAL_SERVER_ERROR`: Unexpected server errors

---

4. Decline Follow Request

Purpose: Decline a pending follow request from another user

Endpoint: **DELETE** `{base_url}/e-social/follows/requests/{followId}/decline`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
followId	string	Yes	UUID of the follow request to decline	Must be valid UUID format

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Follow request declined",
  "action_time": "2025-12-11T10:30:45",
  "data": null
}
```

Standard Error Types:

Application-Level Exceptions (400-499)

- `400 BAD_REQUEST`: Follow request is not pending
- `401 UNAUTHORIZED`: Authentication issues
- `404 NOT_FOUND`: Follow request not found
- `500 INTERNAL_SERVER_ERROR`: Unexpected server errors

---

5. Get Followers

Purpose: Retrieve the complete list of users following a specified user with relationship context

Endpoint: **GET** `{base_url}/e-social/follows/followers/{userId}`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
userId	string	Yes	UUID of the user whose followers to retrieve	Must be valid UUID format

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Followers retrieved successfully",
  "action_time": "2025-12-11T10:30:45",
  "data": [
    {
      "id": "660e8400-e29b-41d4-a716-446655440011",
      "user": {
        "id": "987e6543-e21b-12d3-a456-426614174999",
        "userName": "jane_smith",
        "firstName": "Jane",
        "lastName": "Smith",
        "profilePictureUrls": ["https://cdn.nexgate.com/profiles/jane_smith.jpg"],
        "isVerified": false
      },
      "status": "ACCEPTED",
      "createdAt": "2025-12-10T14:20:15",
      "followsYou": true,
      "youFollow": false
    },
    {
      "id": "770e8400-e29b-41d4-a716-446655440022",
```

```
      "user": {
        "id": "111e1111-e11b-11d1-a111-111111111111",
        "userName": "bob_wilson",
        "firstName": "Bob",
        "lastName": "Wilson",
        "profilePictureUrls": ["https://cdn.nexgate.com/profiles/bob_wilson.jpg"],
        "isVerified": true
      },
      "status": "ACCEPTED",
      "createdAt": "2025-12-09T08:45:30",
      "followsYou": true,
      "youFollow": true
    }
  ]
}
```


Success Response Fields:

Field	Description
data	Array of follower objects
data[].id	Unique identifier for the follow relationship (UUID format)
data[].user	Object containing information about the follower user
data[].user.id	UUID of the follower user
data[].user.userName	Username of the follower
data[].user.firstName	First name of the follower
data[].user.lastName	Last name of the follower
data[].user.profilePictureUrls	Array of profile picture URLs for the follower
data[].user.isVerified	Boolean indicating if the follower account is verified
data[].status	Follow status - always "ACCEPTED" for this endpoint
data[].createdAt	Timestamp when the follow relationship was created (ISO 8601 format)
data[].followsYou	**NEW:** Does this follower follow the authenticated user?
data[].youFollow	**NEW:** Does the authenticated user follow this follower back?

Standard Error Types:

Application-Level Exceptions (400-499)

- `401 UNAUTHORIZED`: Authentication issues
- `404 NOT_FOUND`: User not found
- `500 INTERNAL_SERVER_ERROR`: Unexpected server errors

---

6. Get Following

Purpose: Retrieve the complete list of users that a specified user is following with relationship context

Endpoint: **GET** `{base_url}/e-social/follows/following/{userId}`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
userId	string	Yes	UUID of the user whose following list to retrieve	Must be valid UUID format

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Following retrieved successfully",
  "action_time": "2025-12-11T10:30:45",
  "data": [
    {
      "id": "660e8400-e29b-41d4-a716-446655440011",
      "user": {
```

```
        "id": "222e2222-e22b-22d2-a222-222222222222",
        "userName": "alice_johnson",
        "firstName": "Alice",
        "lastName": "Johnson",
        "profilePictureUrls": ["https://cdn.nexgate.com/profiles/alice_johnson.jpg"],
        "isVerified": true
      },
      "status": "ACCEPTED",
      "createdAt": "2025-12-08T16:30:00",
      "followsYou": true,
      "youFollow": true
    }
  ]
}
```


Success Response Fields:

Field	Description
data	Array of following objects
data[].id	Unique identifier for the follow relationship (UUID format)
data[].user	Object containing information about the followed user
data[].user.id	UUID of the followed user
data[].user.userName	Username of the followed user
data[].user.firstName	First name of the followed user
data[].user.lastName	Last name of the followed user
data[].user.profilePictureUrls	Array of profile picture URLs for the followed user
data[].user.isVerified	Boolean indicating if the followed account is verified
data[].status	Follow status - always "ACCEPTED" for this endpoint
data[].createdAt	Timestamp when the follow relationship was created (ISO 8601 format)
data[].followsYou	**NEW:** Does this user follow the authenticated user back?
data[].youFollow	**NEW:** Does the authenticated user follow this user?

Standard Error Types:

Application-Level Exceptions (400-499)

- `401 UNAUTHORIZED`: Authentication issues

- `404 NOT_FOUND`: User not found
- `500 INTERNAL_SERVER_ERROR`: Unexpected server errors

---

7. Get Followers (Paginated)

Purpose: Retrieve a paginated list of users following a specified user with relationship context

Endpoint: **GET** `{base_url}/e-social/follows/followers/{userId}/paged`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
userId	string	Yes	UUID of the user whose followers to retrieve	Must be valid UUID format

Query Parameters:

Parameter	Type	Required	Description	Validation	Default
page	integer	No	Page number (1-indexed)	Min: 1	1
size	integer	No	Number of items per page	Min: 1, Max: 100	20

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Followers retrieved successfully",
```

```
"action_time": "2025-12-11T10:30:45",
"data": {
  "content": [
    {
      "id": "660e8400-e29b-41d4-a716-446655440011",
      "user": {
        "id": "987e6543-e21b-12d3-a456-426614174999",
        "userName": "jane_smith",
        "firstName": "Jane",
        "lastName": "Smith",
        "profilePictureUrls": ["https://cdn.nexgate.com/profiles/jane_smith.jpg"],
        "isVerified": false
      },
      "status": "ACCEPTED",
      "createdAt": "2025-12-10T14:20:15",
      "followsYou": true,
      "youFollow": false
    }
  ],
  "pageable": {
    "pageNumber": 0,
    "pageSize": 20,
    "sort": {
      "empty": true,
      "sorted": false,
      "unsorted": true
    },
    "offset": 0,
    "paged": true,
    "unpaged": false
  },
  "last": true,
  "totalPages": 1,
  "totalElements": 1,
  "first": true,
  "size": 20,
  "number": 0,
  "sort": {
    "empty": true,
    "sorted": false,
```

```
      "unsorted": true
    },
    "numberOfElements": 1,
    "empty": false
  }
}
```

Success Response Fields:

Field	Description
data.content	Array of follower objects for the current page (includes `followsYou` and `youFollow` fields)
data.content[].followsYou	**NEW:** Does this follower follow the authenticated user?
data.content[].youFollow	**NEW:** Does the authenticated user follow this follower back?
data.pageable	Pagination metadata
data.totalPages	Total number of pages available
data.totalElements	Total number of followers
data.first	Boolean indicating if this is the first page
data.last	Boolean indicating if this is the last page
data.size	Page size
data.number	Current page number (0-indexed)
data.numberOfElements	Number of elements in current page
data.empty	Boolean indicating if the page is empty

---

8. Get Following (Paginated)

Purpose: Retrieve a paginated list of users that a specified user is following with relationship context

Endpoint: **GET** `{base_url}/e-social/follows/following/{userId}/paged`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
userId	string	Yes	UUID of the user whose following list to retrieve	Must be valid UUID format

Query Parameters:

Parameter	Type	Required	Description	Validation	Default
page	integer	No	Page number (1-indexed)	Min: 1	1
size	integer	No	Number of items per page	Min: 1, Max: 100	20

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Following retrieved successfully",
  "action_time": "2025-12-11T10:30:45",
  "data": {
    "content": [
      {
        "id": "660e8400-e29b-41d4-a716-446655440011",
        "user": {
          "id": "222e2222-e22b-22d2-a222-222222222222",
          "userName": "alice_johnson",
          "firstName": "Alice",
          "lastName": "Johnson",
          "profilePictureUrls": ["https://cdn.nexgate.com/profiles/alice_johnson.jpg"],
          "isVerified": true
        },
        "status": "ACCEPTED",
        "createdAt": "2025-12-08T16:30:00",
```

```
        "followsYou": true,
        "youFollow": true
      }
    ],
    "pageable": {
      "pageNumber": 0,
      "pageSize": 20,
      "sort": {
        "empty": true,
        "sorted": false,
        "unsorted": true
      },
      "offset": 0,
      "paged": true,
      "unpaged": false
    },
    "last": true,
    "totalPages": 1,
    "totalElements": 1,
    "first": true,
    "size": 20,
    "number": 0,
    "sort": {
      "empty": true,
      "sorted": false,
      "unsorted": true
    },
    "numberOfElements": 1,
    "empty": false
  }
}
```


Success Response Fields:

Field	Description
data.content	Array of following objects for the current page (includes `followsYou` and `youFollow` fields)
data.content[].followsYou	**NEW:** Does this user follow the authenticated user back?
data.content[].youFollow	**NEW:** Does the authenticated user follow this user?
data.pageable	Pagination metadata

Field	Description
data.totalPages	Total number of pages available
data.totalElements	Total number of users being followed
data.first	Boolean indicating if this is the first page
data.last	Boolean indicating if this is the last page
data.size	Page size
data.number	Current page number (0-indexed)
data.numberOfElements	Number of elements in current page
data.empty	Boolean indicating if the page is empty

---

9. Get Pending Follow Requests

Purpose: Retrieve all pending follow requests for the authenticated user (requests they need to approve)

Endpoint: **GET** `{base_url}/e-social/follows/requests/pending`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Pending requests retrieved successfully",
  "action_time": "2025-12-11T10:30:45",
  "data": [
    {
```

```
      "id": "660e8400-e29b-41d4-a716-446655440011",
      "user": {
        "id": "333e3333-e33b-33d3-a333-333333333333",
        "userName": "charlie_brown",
        "firstName": "Charlie",
        "lastName": "Brown",
        "profilePictureUrls": ["https://cdn.nexgate.com/profiles/charlie_brown.jpg"],
        "isVerified": false
      },
      "status": "PENDING",
      "createdAt": "2025-12-11T09:15:30",
      "followsYou": false,
      "youFollow": false
    }
  ]
}
```


Success Response Fields:

Field	Description
data	Array of pending follow request objects
data[].id	Unique identifier for the follow relationship (UUID format)
data[].user	Object containing information about the user requesting to follow
data[].status	Follow status - always "PENDING" for this endpoint
data[].createdAt	Timestamp when the follow request was created (ISO 8601 format)
data[].followsYou	**NEW:** Does this requester follow the authenticated user?
data[].youFollow	**NEW:** Does the authenticated user follow this requester?

---

10. Get User Stats

Purpose: Retrieve follower, following, and pending request counts for a specified user

Endpoint: **GET** `{base_url}/e-social/follows/stats/{userId}`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
userId	string	Yes	UUID of the user whose stats to retrieve	Must be valid UUID format

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "User stats retrieved successfully",
  "action_time": "2025-12-11T10:30:45",
  "data": {
    "userId": "123e4567-e89b-12d3-a456-426614174000",
    "followersCount": 1250,
    "followingCount": 487,
    "pendingRequestsCount": 15
  }
}
```

Success Response Fields:

Field	Description
userId	UUID of the user these stats belong to
followersCount	Total number of accepted followers for this user
followingCount	Total number of users this user is following (accepted follows only)
pendingRequestsCount	Total number of pending follow requests for this user

---

11. Check Follow Status

Purpose: Check if the authenticated user is following a specified user

Endpoint: **GET** `{base_url}/e-social/follows/check/{userId}`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
userId	string	Yes	UUID of the user to check follow status for	Must be valid UUID format

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Follow status checked",
  "action_time": "2025-12-11T10:30:45",
  "data": {
    "userId": "987e6543-e21b-12d3-a456-426614174999",
    "isFollowing": true
  }
}
```

Success Response Fields:

Field	Description
userId	UUID of the user being checked

Field	Description
isFollowing	Boolean indicating if the authenticated user is following this user (includes pending and accepted follows)

---

12. Get Featured Users

Purpose: Retrieve a list of suggested users to follow, sorted by follower count, excluding users already followed by the authenticated user

Endpoint: **GET** `{base_url}/e-social/follows/featured`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Query Parameters:

Parameter	Type	Required	Description	Validation	Default
limit	integer	No	Maximum number of featured users to return	Min: 1, Max: 100	20

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Featured users retrieved successfully",
  "action_time": "2025-12-11T10:30:45",
  "data": [
    {
      "id": "555e5555-e55b-55d5-a555-555555555555",
```

```
      "userName": "tech_guru",
      "firstName": "Sarah",
      "lastName": "Martinez",
      "profilePictureUrls": ["https://cdn.nexgate.com/profiles/tech_guru.jpg"],
      "isVerified": true,
      "followersCount": 125000,
      "followsMe": false
    },
    {
      "id": "666e6666-e66b-66d6-a666-666666666666",
      "userName": "travel_explorer",
      "firstName": "Mike",
      "lastName": "Chen",
      "profilePictureUrls": ["https://cdn.nexgate.com/profiles/travel_explorer.jpg"],
      "isVerified": true,
      "followersCount": 98500,
      "followsMe": true
    }
  ]
}
```


Success Response Fields:

Field	Description
data	Array of featured user objects
data[].id	UUID of the featured user
data[].userName	Username of the featured user
data[].firstName	First name of the featured user
data[].lastName	Last name of the featured user
data[].profilePictureUrls	Array of profile picture URLs for the featured user
data[].isVerified	Boolean indicating if the featured account is verified
data[].followersCount	Total number of followers the featured user has
data[].followsMe	Boolean indicating if this featured user follows the authenticated user

---

13. Get Featured Users (Paginated)

Purpose: Retrieve a paginated list of suggested users to follow, sorted by follower count, excluding users already followed by the authenticated user

Endpoint: **GET** `{base_url}/e-social/follows/featured/paged`

Access Level: □□ Protected (Requires Bearer Token Authentication)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication (format: `Bearer <token>`)
Content-Type	string	Yes	Must be `application/json`

Query Parameters:

Parameter	Type	Required	Description	Validation	Default
page	integer	No	Page number (1-indexed)	Min: 1	1
size	integer	No	Number of items per page	Min: 1, Max: 100	20

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Featured users retrieved successfully",
  "action_time": "2025-12-11T10:30:45",
  "data": {
    "content": [
      {
        "id": "555e5555-e55b-55d5-a555-555555555555",
        "userName": "tech_guru",
        "firstName": "Sarah",
        "lastName": "Martinez",
        "profilePictureUrls": ["https://cdn.nexgate.com/profiles/tech_guru.jpg"],
        "isVerified": true,
        "followersCount": 125000,
        "followsMe": false
```

```
      }
    ],
    "pageable": {
      "pageNumber": 0,
      "pageSize": 20,
      "sort": {
        "empty": true,
        "sorted": false,
        "unsorted": true
      },
      "offset": 0,
      "paged": true,
      "unpaged": false
    },
    "last": false,
    "totalPages": 5,
    "totalElements": 87,
    "first": true,
    "size": 20,
    "number": 0,
    "sort": {
      "empty": true,
      "sorted": false,
      "unsorted": true
    },
    "numberOfElements": 20,
    "empty": false
  }
}
```


Success Response Fields:

Field	Description
data.content	Array of featured user objects for the current page
data.content[].id	UUID of the featured user
data.content[].userName	Username of the featured user
data.content[].firstName	First name of the featured user
data.content[].lastName	Last name of the featured user
data.content[].profilePictureUrls	Array of profile picture URLs

Field	Description
data.content[].isVerified	Boolean indicating if account is verified
data.content[].followersCount	Total number of followers
data.content[].followsMe	Boolean indicating if this user follows the authenticated user
data.pageable	Pagination metadata
data.totalPages	Total number of pages available
data.totalElements	Total number of featured users
data.first	Boolean indicating if this is the first page
data.last	Boolean indicating if this is the last page
data.size	Page size
data.number	Current page number (0-indexed)
data.numberOfElements	Number of elements in current page
data.empty	Boolean indicating if the page is empty

Quick Reference Guide

Common HTTP Status Codes

- `200 OK`: Successful GET/POST/DELETE request
- `400 Bad Request`: Invalid request data or business rule violation
- `401 Unauthorized`: Authentication required/failed
- `404 Not Found`: Resource not found
- `500 Internal Server Error`: Server error

Authentication

- **Bearer Token**: Include `Authorization: Bearer your_token` in headers

Data Format Standards

- **Dates**: Use ISO 8601 format (2025-12-11T14:30:00Z)
- **IDs**: UUID format (e.g., 550e8400-e29b-41d4-a716-446655440000)
- **Pagination**: 1-indexed pages (page=1 for first page), default size=20

Relationship Field Reference

Field	Meaning
`followsYou`	Does this user follow **you** (authenticated user)?
`youFollow`	Do **you** (authenticated user) follow this user?

These fields enable smart UI rendering:

- Show "Follow Back" when `followsYou: true` and `youFollow: false`
- Show "Follows You" badge when `followsYou: true`
- Show "Following" when `youFollow: true`
- Show "Follow" when both are `false`

Version History

v1.1 (2025-12-20)

- Added relationship context fields (`followsYou`, `youFollow`) to all follower/following endpoints
- Added "How The Follow System Works" section with detailed relationship logic
- Added "UI Implementation Best Practices" section with button state management and confirmation patterns
- Updated all response samples to include new relationship fields
- Enhanced documentation with button logic examples and visual state guidelines

v1.0 (2025-12-11)

- Initial release
- All core follow system endpoints
- Basic follower/following management
- Featured users discovery

Revision #3
Created 11 December 2025 09:06:57 by Admin Qbit
Updated 20 December 2025 06:54:24 by Admin Qbit