Search API

Author: Josh S. Sakweli, Backend Lead Team
Last Updated: 2025-12-23
Version: v1.0

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

Short Description: Public search endpoints for discovering users and shops on the Nexgate platform. These endpoints power the @mention and $shop autocomplete features across the application.

Hints:

- Use `@` prefix for user search (automatically stripped)
- Use `$` prefix for shop search (automatically stripped)
- Results are ordered by relevance: exact match → starts with → contains
- All endpoints are public and do not require authentication
- Rate limiting applies to prevent abuse

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": "2025-12-23T10:30:45",
  "data": {
    // Actual response data goes here
  }
}
```

Error Response Structure

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Error description",
  "action_time": "2025-12-23T10:30:45",
  "data": "Error description"
}
```

Endpoints

1. Search Users

Purpose: Search for users by username, first name, or last name for @mentions and user discovery.

Endpoint: `GET` `{base_url}/users/search`

Access Level: □□ Public (Rate limited)

Authentication: None

Query Parameters:

Parameter	Type	Required	Description	Validation	Default
q	string	No	Search query (username, first name, or last name). `@` prefix is automatically removed	Min: 1 character after trimming	-
page	integer	No	Page number (1-based)	Min: 1	1
size	integer	No	Number of results per page	Min: 1, Max: 50	10

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
```

```
"message": "Users retrieved successfully",
"action_time": "2025-12-23T11:14:05.293524449",
"data": {
  "content": [
    {
      "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
      "userName": "john_doe",
      "displayName": "John Doe",
      "avatarUrl": "https://cdn.nextgate.com/avatars/john_doe.jpg",
      "isVerified": true
    },
    {
      "id": "b2c3d4e5-f6a7-8901-bcde-f12345678901",
      "userName": "johnny_b",
      "displayName": "Johnny Brown",
      "avatarUrl": null,
      "isVerified": false
    }
  ],
  "pageable": {
    "pageNumber": 0,
    "pageSize": 10,
    "sort": {
      "empty": true,
      "sorted": false,
      "unsorted": true
    },
    "offset": 0,
    "paged": true,
    "unpaged": false
  },
  "last": false,
  "totalPages": 3,
  "totalElements": 25,
  "first": true,
  "size": 10,
  "number": 0,
  "numberOfElements": 10,
  "sort": {
    "empty": true,
```

```
      "sorted": false,
      "unsorted": true
    },
    "empty": false
  }
}
```

Success Response Fields:

Field	Description
content	Array of user objects matching the search query
content[].id	Unique identifier of the user
content[].userName	Username of the user
content[].displayName	Full display name (firstName + lastName, or userName if not available)
content[].avatarUrl	URL to user's profile picture (null if not set)
content[].isVerified	Whether the user account is verified
totalElements	Total number of users matching the query
totalPages	Total number of pages available
first	Whether this is the first page
last	Whether this is the last page

Empty Result Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Users retrieved successfully",
  "action_time": "2025-12-23T11:14:05.293524449",
  "data": {
    "content": [],
    "pageable": {
      "pageNumber": 0,
      "pageSize": 10,
      "sort": {
        "empty": true,
        "sorted": false,
        "unsorted": true
      },
```

```
      "offset": 0,
      "paged": true,
      "unpaged": false
    },
    "last": true,
    "totalPages": 0,
    "totalElements": 0,
    "first": true,
    "size": 10,
    "number": 0,
    "numberOfElements": 0,
    "sort": {
      "empty": true,
      "sorted": false,
      "unsorted": true
    },
    "empty": true
  }
}
```


Example Requests:

Use Case	Request
Basic search	`/users/search?q=john`
Search with @ prefix	`/users/search?q=@john`
Paginated search	`/users/search?q=john&page=2&size=20`

---

2. Search Shops

Purpose: Search for shops by name, slug, or tagline for $mentions and shop discovery.

Endpoint: `GET` `{base_url}/shops/search`

Access Level: □□ Public (Rate limited)

Authentication: None

Query Parameters:

Parameter	Type	Required	Description	Validation	Default
q	string	No	Search query (shop name, slug, or tagline). `$` prefix is automatically removed	Min: 1 character after trimming	-
page	integer	No	Page number (1-based)	Min: 1	1
size	integer	No	Number of results per page	Min: 1, Max: 50	10

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Shops retrieved successfully",
  "action_time": "2025-12-23T11:20:30.123456789",
  "data": {
    "content": [
      {
        "shopId": "c3d4e5f6-a7b8-9012-cdef-123456789012",
        "shopName": "Mama's Kitchen",
        "shopSlug": "mamas-kitchen",
        "tagline": "Best local food in town",
        "logoUrl": "https://cdn.nextgate.com/shops/mamas-kitchen/logo.jpg",
        "isVerified": true,
        "verificationBadge": "GOLD",
        "city": "Dar es Salaam"
      },
      {
        "shopId": "d4e5f6a7-b8c9-0123-def0-234567890123",
        "shopName": "Mama Lishe",
        "shopSlug": "mama-lishe",
        "tagline": "Authentic Tanzanian cuisine",
        "logoUrl": null,
        "isVerified": false,
        "verificationBadge": null,
        "city": "Arusha"
      }
```

```
    ],
    "pageable": {
      "pageNumber": 0,
      "pageSize": 10,
      "sort": {
        "empty": true,
        "sorted": false,
        "unsorted": true
      },
      "offset": 0,
      "paged": true,
      "unpaged": false
    },
    "last": false,
    "totalPages": 2,
    "totalElements": 15,
    "first": true,
    "size": 10,
    "number": 0,
    "numberOfElements": 10,
    "sort": {
      "empty": true,
      "sorted": false,
      "unsorted": true
    },
    "empty": false
  }
}
```


Success Response Fields:

Field	Description
content	Array of shop objects matching the search query
content[].shopId	Unique identifier of the shop
content[].shopName	Display name of the shop
content[].shopSlug	URL-friendly unique identifier
content[].tagline	Short promotional text for the shop
content[].logoUrl	URL to shop's logo image (null if not set)
content[].isVerified	Whether the shop is verified

Field	Description
content[].verificationBadge	Verification badge level (GOLD, SILVER, BRONZE, or null)
content[].city	City where the shop is located
totalElements	Total number of shops matching the query
totalPages	Total number of pages available
first	Whether this is the first page
last	Whether this is the last page

Empty Result Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Shops retrieved successfully",
  "action_time": "2025-12-23T11:20:30.123456789",
  "data": {
    "content": [],
    "pageable": {
      "pageNumber": 0,
      "pageSize": 10,
      "sort": {
        "empty": true,
        "sorted": false,
        "unsorted": true
      },
      "offset": 0,
      "paged": true,
      "unpaged": false
    },
    "last": true,
    "totalPages": 0,
    "totalElements": 0,
    "first": true,
    "size": 10,
    "number": 0,
    "numberOfElements": 0,
    "sort": {
      "empty": true,
      "sorted": false,
```

```
      "unsorted": true
    },
    "empty": true
  }
}
```

Example Requests:

Use Case	Request
Basic search	`/shops/search?q=mama`
Search with $ prefix	`/shops/search?q=$mama`
Paginated search	`/shops/search?q=kitchen&page=2&size=20`

Search Ordering Logic

Both search endpoints use smart ordering to return the most relevant results first:

Priority	Match Type	Example (query: "john")
1	Exact match	`john`
2	Starts with	`johnny`, `john_doe`
3	Contains	`big_john`, `_john_`

Results with the same priority are sorted alphabetically.

Error Responses

Standard Error Types:

Application-Level Exceptions (400-499)

- `400 BAD_REQUEST`: Invalid request parameters
- `429 TOO_MANY_REQUESTS`: Rate limit exceeded

Error Response Examples:

Rate Limit Exceeded (429):

```
{
  "success": false,
  "httpStatus": "TOO_MANY_REQUESTS",
  "message": "Rate limit exceeded. Please try again later.",
  "action_time": "2025-12-23T11:30:45",
  "data": "Rate limit exceeded. Please try again later."
}
```

Quick Reference

Feature	User Search	Shop Search
Endpoint	`/users/search`	`/shops/search`
Prefix	`@` (optional)	`$` (optional)
Searches	userName, firstName, lastName	shopName, shopSlug, tagline
Filters	Non-locked users only	Active, non-deleted shops only
Auth	None	None

Future Scaling Strategy

As Nexgate grows, the search system will evolve to handle millions of users and shops while maintaining fast response times. Here's how we'll scale like Instagram and GitHub.

Phase 1: Current Implementation (MVP)

Scale: Up to 100K users/shops

How it works:

- Direct PostgreSQL queries with pattern matching
- Smart ordering: exact match → starts with → contains
- Simple and effective for early-stage growth

Limitations:

- Slows down with large datasets
- No typo tolerance

- No personalization

Phase 2: Database Optimization

Scale: 100K - 1M users/shops

What we'll add:

- **Trigram Indexing (pg_trgm)**: PostgreSQL extension that enables fast fuzzy search and handles typos (e.g., "jonh" finds "john")
- **Redis Caching**: Cache popular search queries and recent searches per user
- **Optimized Indexes**: GIN indexes for faster text matching

Benefits:

- 3-5x faster queries
- Typo tolerance
- Reduced database load

Phase 3: Dedicated Search Engine

Scale: 1M - 100M users/shops

What we'll add:

- **Elasticsearch Cluster**: Dedicated search infrastructure separate from main database
- **Real-time Sync**: Changes in PostgreSQL automatically sync to Elasticsearch
- **Advanced Features**: Autocomplete suggestions, fuzzy matching, relevance scoring

Benefits:

- Sub-10ms response times
- Horizontal scaling (add more nodes as needed)
- Rich search features without impacting main database

Phase 4: Personalized Search (Instagram/GitHub Style)

Scale: 100M+ users/shops

How Instagram does it: They don't search all 2 billion users. Instead, they search within YOUR world first.

Tiered Search Approach:

Priority	Who Gets Searched First	Why
1st	People you message frequently	Strongest signal of relationship
2nd	People you interact with (likes, comments)	Active engagement
3rd	People you follow	Explicit interest
4th	People who follow you	They know you
5th	Global search	Only if above tiers don't have enough results

What we'll add:

- **Interaction Graph**: Track who talks to whom, who views whose profile, who likes whose posts
- **Pre-computed Candidates**: For each user, maintain a list of likely search targets
- **ML Ranking**: Machine learning model to predict who you're most likely searching for
- **Prefix Caching**: Pre-compute results for common prefixes (typing "j" instantly shows your friend "John")

Relevance Scoring:

- 40% - How often you interact with this person/shop
- 30% - How well the query matches their name
- 20% - How recently you interacted
- 10% - Their popularity (followers, verification)

Performance Targets

Phase	Scale	Average Response Time
Phase 1 (Current)	<100K	~50ms
Phase 2 (Optimized DB)	<1M	~30ms
Phase 3 (Elasticsearch)	<100M	~10ms
Phase 4 (Personalized)	100M+	~5ms

Migration Triggers

When to Move	Action
50K users	Add pg_trgm + Redis caching
500K users	Introduce Elasticsearch
5M users	Build interaction graph + personalization

Key Principle: The API stays the same. Users won't notice any changes except faster, smarter results.

Revision #3
Created 23 December 2025 09:46:21 by Admin Qbit
Updated 23 December 2025 09:54:48 by Admin Qbit