Nexgate Social Media service Overview

Nexgate's social media platform is a hybrid of Instagram and Twitter functionality, integrated deeply with the existing eCommerce and event management systems. The platform excludes stories but includes comprehensive posting, interaction, messaging, and discovery features.

Key Differentiators

- **Commerce Integration**: Users can link products, shops, and events directly in posts
- **Shop Mentions**: Uses `$shopname` syntax for shop mentions/tags
- **Product-Context Messaging**: Messages can include product banners with details
- **Collaborative Commerce**: Social discovery engine for products and events

---

Core Features

1. User Relationships

Follow System

- **Follow User**: User A follows User B
- **Unfollow User**: User A unfollows User B
- **Get Followers**: Retrieve list of users following a specific user
- **Get Following**: Retrieve list of users a specific user follows
- **Follow Counts**: Display follower and following counts on profiles

Privacy Controls

- **Block User**: Prevent interaction with specific users
- **Unblock User**: Remove block restriction
- **Mute User**: Hide user's content without unfollowing
- **Unmute User**: Restore muted user's content visibility

Account Privacy

- **Public Account**: Anyone can see posts and follow
- **Private Account**: Followers must be approved before seeing content

2. Posts

Post Creation

Posts are the central content type in the platform.

Post Types:

- Text-only posts
- Image posts (single or multiple images)
- Video posts
- Poll posts
- Mixed media posts

Post States:

- **Draft**: Saved but not published
- **Published**: Live and visible based on privacy settings
- **Deleted**: Soft-deleted (retained for audit)

Creation Features:

- Rich text content
- Image uploads (multiple)
- Video uploads
- Add polls with multiple options
- Link products using product IDs
- Link buy-together groups (to boost group participation)
- Link installment plans (specific payment plans for products)
- Link shops using `$shopname` syntax
- Link events using event IDs
- Tag users using `@username` syntax
- Tag shops using `$shopname` syntax
- Tag products on images (coordinates-based like Instagram)
- Tag users on images (coordinates-based)
- Add hashtags using `#hashtag` syntax
- Set as collaborative post (multiple co-authors)
- Save as draft for later publishing

Post Management

- **Edit Post**: Modify post content (with edit history)
- **Delete Post**: Soft delete post
- **Publish Draft**: Convert draft to published post
- **View Post**: Display single post with all details

- **Get User Posts**: Retrieve all posts by specific user
- **Get Feed**: Get posts from followed users (chronological)

Image Tagging System

Similar to Instagram's tag functionality:

- Click/tap on image at specific coordinates
- Tag can be: User (`@username`), Product (product_id), Shop (`$shopname`)
- Multiple tags per image
- Tags display as interactive overlay on image
- Clicking tag navigates to tagged entity

Collaborative Posts

Collaborative posts allow multiple users to co-author a single post, with all authors displayed and the post appearing on all collaborators' profiles.

Features:

- **Invite Collaborators**: During post creation or editing, invite other users as co-authors
- **Multiple Authors**: Support up to 5-10 co-authors per post
- **Accept/Decline**: Invited collaborators receive notification and can accept or decline
- **Pending Status**: Post shows "Pending collaboration with @username" until accepted
- **Profile Display**: Accepted collaborative posts appear on all co-authors' profiles
- **Byline**: Post displays "By @user1, @user2, and @user3"
- **Equal Rights**: All collaborators can edit or delete the post (with consent logic)
- **Attribution**: All engagement (likes, comments) benefits all collaborators

Use Cases:

```
Fashion Brand + Influencer Collaboration:
- Brand creates post about new collection
- Adds influencer as co-author
- Influencer accepts
- Post appears on both profiles
- Both audiences see the content
- Both get engagement metrics

Joint Product Launch:
- Two shops collaborate on bundle deal
- Create one post together
- Shows on both shop pages
- Shared promotion effort
```

Workflow:

1. User A creates post
2. User A adds User B, User C as collaborators during creation
3. Post saved with status = "pending_collaboration"
4. Notifications sent to User B and User C: "User A invited you to collaborate on a post"
5. User B accepts → PostCollaborator status = "accepted"
6. User C declines → PostCollaborator deleted, User A notified
7. Once all pending invites resolved (accepted/declined), post can be published
8. Published collaborative post appears on User A and User B profiles
9. Byline shows: "By @userA and @userB"

Rules:

- Post creator is always primary author (cannot be removed)
- Collaborators can decline invitation without penalty
- If any collaborator leaves (removes themselves), post remains but their name removed
- Post creator can remove collaborators anytime
- Deleted collaborative posts removed from all profiles

---

3. Interactions

Likes

- **Like Post**: User likes a post
- **Unlike Post**: User removes like
- **Like Comment**: User likes a comment
- **Unlike Comment**: User removes comment like
- **View Likes**: See list of users who liked post/comment
- **Like Count**: Display total likes on post/comment

Comments

- **Comment on Post**: Add text comment to post
- **Edit Comment**: Modify own comment
- **Delete Comment**: Remove own comment (soft delete)
- **Reply to Comment**: Nested comment system (up to N levels)
- **Tag in Comments**: Tag users using `@username` in comments
- **Mention Products**: Reference products in comments
- **Comment Count**: Display total comments on post

Reposting/Sharing

- **Repost**: Share another user's post without adding commentary (like Twitter retweet)
- **Quote Post**: Share post with your own added text/commentary

- **Unrepost**: Remove repost or quote post
- **Share via DM**: Send post link through direct message
- **Repost Count**: Display total reposts

Bookmarks

- **Bookmark Post**: Save post for later viewing
- **Remove Bookmark**: Unsave post
- **View Bookmarks**: Access all saved posts
- **Collections**: Organize bookmarks into named collections/folders
  - Create collection
  - Add post to collection
  - Remove post from collection
  - Delete collection

---

4. Polls

Polls are embedded within posts.

Poll Features:

- Multiple choice options (2-10 options)
- Single vote or multiple votes allowed
- Vote duration/expiry time
- Vote anonymity settings
- Real-time result updates

Poll Actions:

- **Create Poll**: Define options and settings during post creation
- **Vote**: Cast vote on poll option(s)
- **Change Vote**: Modify vote before poll expires
- **View Results**: See voting statistics
  - Percentage per option
  - Total votes
  - Time remaining
  - Voter list (if not anonymous)

---

5. Direct Messaging (DMs)

Basic Messaging

- **Send Message**: Text, images, videos, links

- **Edit Message**: Modify sent message (within time window)
- **Delete Message**: Remove message (for self or everyone)
- **React to Message**: Emoji reactions
- **Reply to Message**: Quote/reply to specific message in thread
- **Forward Message**: Send message to another conversation

Conversation Management

- **One-on-One Chat**: Private conversation between two users
- **Mark as Read/Unread**: Message read status
- **Search Messages**: Full-text search within conversations
- **Mute Conversation**: Disable notifications for conversation
- **Block in DMs**: Prevent messaging from specific user
- **Delete Conversation**: Remove conversation thread

Product-Context Messaging

Unique feature for commerce integration:

- **Contact from Product**: Initiate message from product page
- **Product Banner**: Message includes product image, name, price
- **Product Context**: Conversation maintains product reference
- **Quick Actions**: Direct links to product, shop, or seller profile

Use Case Example:

```
User sees a product → Clicks "Contact Seller"
→ DM opens with product banner at top
→ Conversation starts with product context
→ Easy reference throughout discussion
```

---

6. Commerce Integration

Product Linking

- **Link Product in Post**: Attach product(s) during post creation
- **Product Card Display**: Show product preview (image, name, price)
- **Click-through**: Navigate to product page from post
- **Multiple Products**: Support multiple product links per post

Shop Linking & Tagging

- **Shop Mention**: Use `$shopname` syntax in post text
- **Tag Shop**: Explicitly tag shop in post metadata

- **Shop Card Display**: Show shop preview when linked
- **Navigate to Shop**: Click-through to shop page

Event Linking

- **Link Event in Post**: Attach event during post creation
- **Event Card Display**: Show event preview (image, title, date, location)
- **Event Status**: Display ticket availability, price range
- **Click-through**: Navigate to event page from post

Buy-Together Group Linking

Users can share buy-together groups in posts to boost participation and fill group spots.

Features:

- **Link Group in Post**: Attach active buy-together group during post creation
- **Group Card Display**: Show group preview with:
  - Product image and name
  - Group target price (discounted price when full)
  - Current participants count / Total spots
  - Time remaining to fill group
  - Savings amount vs individual purchase
  - "Join Group" CTA button
- **Group Status Indicators**:
  - Open (spots available)
  - Almost Full (1-2 spots left)
  - Full (no spots available)
  - Expired (time ran out)
- **Click-through**: Navigate to group detail page
- **Real-time Updates**: Group status updates as people join

Use Case:

```
User creates buy-together group for iPhone 15
→ User shares group in post: "Join my iPhone group! Only 2 spots left, save $100!"
→ Followers see post with group card showing discount and spots
→ Click "Join Group" → Navigate to group page to complete joining
→ Group fills up → Post updates to show "Group Full"
```

Installment Plan Linking

Users can showcase flexible payment options for products in their posts.

Features:

- **Link Installment Plan in Post**: Attach specific payment plan(s) during post creation
- **Multiple Plans Support**: Product can have different installment options (3 months, 6 months, 12 months)
- **Plan Card Display**: Show plan preview with:
  - Product image and name
  - Plan duration (e.g., "6 months")
  - Monthly payment amount
  - Total price (may include interest)
  - Interest rate (if applicable)
  - Down payment (if required)
  - "Buy with Plan" CTA button
- **Plan Comparison**: If multiple plans linked, show side-by-side comparison
- **Eligibility Badge**: Show if user pre-qualifies for plan
- **Click-through**: Navigate to product page with plan pre-selected

Use Case:

```
Shop owner selling laptop at $1200
→ Has 3 installment plans: 3mo ($400/mo), 6mo ($200/mo), 12mo ($100/mo)
→ Creates post: "Get this gaming laptop! Pay as low as $100/month □□"
→ Links 6-month and 12-month plans
→ Post shows both plans with monthly amounts
→ User clicks → Taken to checkout with 6mo plan selected
```

Integrated Discovery

Posts with linked products/shops/events/groups/plans appear in:

- Regular feed (from followed users)
- Explore/discover feed
- Search results (filtered by product/shop/event/group/plan)
- Shop pages (posts mentioning the shop)
- Product pages (posts featuring the product)
- Event pages (posts about the event)
- Buy-together group pages (posts promoting the group)
- Installment plan searches (posts showcasing payment options)

---

7. Hashtags & Discovery

Hashtags

- **Create/Use Hashtag**: Add `#hashtag` to posts
- **Search by Hashtag**: View all posts with specific hashtag

- **Trending Hashtags**: Most used hashtags in time period
- **Follow Hashtag**: See posts with hashtag in feed (optional feature)
- **Hashtag Count**: Number of posts using hashtag

Lists (Curated Feeds)

- **Create List**: Named collection of users
- **Add User to List**: Include user in list
- **Remove from List**: Exclude user from list
- **View List Feed**: See posts only from list members
- **Private/Public Lists**: Control list visibility

Search & Explore

- **Search Users**: Find users by username, name
- **Search Posts**: Full-text search in post content
- **Search Hashtags**: Find relevant hashtags
- **Search Products**: Find posts with specific products
- **Search Shops**: Find posts mentioning shops
- **Search Events**: Find posts about events
- **Search Buy-Together Groups**: Find posts promoting active groups
- **Search Installment Plans**: Find posts with payment plans
- **Explore Feed**: Algorithm-free discovery (recent/trending posts)
- **Suggested Users**: Recommended follows based on connections/interests

---

8. User Profile

Profile Information

- **Username**: Unique identifier
- **Display Name**: Public-facing name
- **Bio**: Text description (max 150-250 chars)
- **Profile Picture**: Avatar image
- **Cover Photo**: Header image
- **Location**: City/country (optional)
- **Website**: External link (optional)
- **Join Date**: Account creation date

Profile Stats

- **Posts Count**: Total published posts
- **Followers Count**: Total followers
- **Following Count**: Total following
- **Likes Received**: Aggregate likes across all posts (optional)

Profile Actions

- **View Profile**: Display user's profile and posts
- **Edit Profile**: Update profile information
- **Account Privacy**: Toggle public/private
- **Verification Badge**: For verified accounts (if applicable)

---

9. Notifications

Users receive notifications for various activities:

Social Interactions

- User followed you
- User liked your post
- User commented on your post
- User replied to your comment
- User mentioned you in post
- User mentioned you in comment
- User tagged you in post
- User tagged you in image
- User reposted your post
- User quote posted your post
- User invited you to collaborate on a post
- User accepted your collaboration invite
- User declined your collaboration invite
- Collaborator edited your shared post

Direct Messages

- New message received
- Someone started conversation with product context

Commerce-Related

- User liked post with your product
- User commented on post with your product
- User shared post with your product
- Shop you tagged shared/liked your post

Notification Settings

- **Push Notifications**: Mobile/browser push
- **In-App Notifications**: Notification center
- **Email Notifications**: Email digest (optional)

- **Notification Preferences**: Granular control per notification type

---

10. Moderation & Safety

Reporting

- **Report Post**: Flag inappropriate content
  - Spam
  - Harassment
  - Misinformation
  - Inappropriate content
  - Intellectual property violation
  - Other (with description)
- **Report User**: Flag user account
- **Report Comment**: Flag inappropriate comment

Content Control

- **Hide Post**: Remove post from your feed without reporting
- **Not Interested**: Signal to reduce similar content (if using algorithm)
- **Mute Keywords**: Hide posts containing specific keywords

Account Actions

- **Block User**: Comprehensive blocking
  - Can't see your posts
  - Can't follow you
  - Can't message you
  - Can't tag you
  - Can't mention you
- **Unblock User**: Restore interaction ability
- **Restrict User**: Limited interaction (shadow restriction)

API Endpoints

Authentication

```
POST   /api/auth/register          - Register new user
POST   /api/auth/login             - Login user
POST   /api/auth/logout            - Logout user
```

```
POST   /api/auth/refresh-token     - Refresh access token
POST   /api/auth/forgot-password   - Request password reset
POST   /api/auth/reset-password    - Reset password
```

User Profile

```
GET    /api/users/:username        - Get user profile
GET    /api/users/:id              - Get user by ID
PATCH  /api/users/:id              - Update user profile
GET    /api/users/:id/posts        - Get user's posts
GET    /api/users/:id/followers    - Get followers list
GET    /api/users/:id/following    - Get following list
GET    /api/users/:id/stats        - Get user statistics
```

Follow System

```
POST   /api/users/:id/follow       - Follow user
DELETE /api/users/:id/unfollow     - Unfollow user
GET    /api/follows/requests       - Get pending follow requests (private accounts)
POST   /api/follows/:id/accept     - Accept follow request
POST   /api/follows/:id/decline    - Decline follow request
```

Block & Mute

```
POST   /api/users/:id/block        - Block user
DELETE /api/users/:id/unblock      - Unblock user
POST   /api/users/:id/mute         - Mute user
DELETE /api/users/:id/unmute       - Unmute user
GET    /api/users/blocked          - Get blocked users list
GET    /api/users/muted            - Get muted users list
```

Posts

```
POST   /api/posts                  - Create post
GET    /api/posts/:id              - Get single post
PATCH  /api/posts/:id              - Edit post
DELETE /api/posts/:id              - Delete post
```

```
GET    /api/posts/:id/likes        - Get post likes
GET    /api/posts/:id/comments     - Get post comments
GET    /api/posts/:id/reposts      - Get post reposts

GET    /api/posts                  - Get feed (from followed users)
GET    /api/posts/explore          - Get explore/discover feed
GET    /api/posts/drafts           - Get user's draft posts
POST   /api/posts/:id/publish      - Publish draft post
```

Post Interactions

```
POST   /api/posts/:id/like         - Like post
DELETE /api/posts/:id/unlike       - Unlike post
POST   /api/posts/:id/repost       - Repost (simple repost)
POST   /api/posts/:id/quote        - Quote post (with text)
DELETE /api/posts/:id/unrepost     - Remove repost/quote
POST   /api/posts/:id/bookmark     - Bookmark post
DELETE /api/posts/:id/unbookmark   - Remove bookmark
POST   /api/posts/:id/hide         - Hide post from feed
POST   /api/posts/:id/report       - Report post
```

Image Tags

```
POST   /api/posts/:id/media/:mediaId/tags     - Add tag to image
DELETE /api/posts/:id/media/:mediaId/tags/:tagId - Remove image tag
GET    /api/posts/:id/media/:mediaId/tags     - Get all tags on image
```

Comments

```
POST   /api/posts/:id/comments           - Add comment to post
GET    /api/comments/:id                 - Get single comment
PATCH  /api/comments/:id                 - Edit comment
DELETE /api/comments/:id                 - Delete comment
POST   /api/comments/:id/like            - Like comment
DELETE /api/comments/:id/unlike          - Unlike comment
POST   /api/comments/:id/reply           - Reply to comment
GET    /api/comments/:id/replies         - Get comment replies
POST   /api/comments/:id/report          - Report comment
```

Polls

```
POST   /api/polls/:id/vote              - Vote on poll
PATCH  /api/polls/:id/vote              - Change vote
GET    /api/polls/:id/results           - Get poll results
POST   /api/polls/:id/close             - Close poll early (creator only)
```

Bookmarks & Collections

```
GET    /api/bookmarks                   - Get all bookmarks
POST   /api/bookmarks/collections       - Create bookmark collection
GET    /api/bookmarks/collections       - Get all collections
GET    /api/bookmarks/collections/:id   - Get collection bookmarks
PATCH  /api/bookmarks/collections/:id   - Update collection
DELETE /api/bookmarks/collections/:id   - Delete collection
POST   /api/bookmarks/:id/move          - Move bookmark to collection
```

Hashtags

```
GET    /api/hashtags/:tag               - Get hashtag info
GET    /api/hashtags/:tag/posts         - Get posts with hashtag
GET    /api/hashtags/trending           - Get trending hashtags
POST   /api/hashtags/:tag/follow        - Follow hashtag
DELETE /api/hashtags/:tag/unfollow      - Unfollow hashtag
```

Lists

```
POST   /api/lists                       - Create list
GET    /api/lists                       - Get user's lists
GET    /api/lists/:id                   - Get single list
PATCH  /api/lists/:id                   - Update list
DELETE /api/lists/:id                   - Delete list
POST   /api/lists/:id/members           - Add user to list
DELETE /api/lists/:id/members/:userId   - Remove user from list
GET    /api/lists/:id/feed              - Get list feed (posts from members)
```

Direct Messages

```
GET    /api/conversations                    - Get user's conversations
POST   /api/conversations                    - Create new conversation
GET    /api/conversations/:id                - Get conversation details
DELETE /api/conversations/:id                - Delete conversation
POST   /api/conversations/:id/mute           - Mute conversation
DELETE /api/conversations/:id/unmute         - Unmute conversation

GET    /api/conversations/:id/messages       - Get conversation messages
POST   /api/conversations/:id/messages       - Send message
PATCH  /api/messages/:id                     - Edit message
DELETE /api/messages/:id                     - Delete message
POST   /api/messages/:id/reactions           - Add reaction to message
DELETE /api/messages/:id/reactions/:emoji    - Remove reaction
POST   /api/conversations/:id/read           - Mark conversation as read
```

Product Context Messaging

```
POST   /api/products/:id/contact             - Initiate message with product context
GET    /api/messages/:id/product-context     - Get product context for message
```

Notifications

```
GET    /api/notifications                    - Get user notifications
GET    /api/notifications/unread-count       - Get unread count
PATCH  /api/notifications/:id/read           - Mark notification as read
PATCH  /api/notifications/read-all           - Mark all as read
DELETE /api/notifications/:id                - Delete notification
PATCH  /api/notifications/settings           - Update notification preferences
```

Search

```
GET    /api/search/users?q=:query            - Search users
GET    /api/search/posts?q=:query            - Search posts
GET    /api/search/hashtags?q=:query         - Search hashtags
GET    /api/search/products?q=:query         - Search posts with products
GET    /api/search/shops?q=:query            - Search posts with shops
GET    /api/search/events?q=:query           - Search posts with events
```

```
GET    /api/search/groups?q=:query             - Search posts with buy-together groups
GET    /api/search/installment-plans?q=:query  - Search posts with installment plans
GET    /api/search/all?q=:query                - Search across all types
```

Reports & Moderation

```
POST   /api/reports                            - Submit report
GET    /api/reports                            - Get user's reports (admin only)
PATCH  /api/reports/:id                        - Update report status (admin only)
```

Business Logic

Feed Generation

Personal Feed (Home Feed)

```
Algorithm: Reverse Chronological (no complex algorithm for v1)

Logic:
1. Get list of users current user follows
2. Exclude blocked users
3. Exclude muted users
4. Fetch posts from followed users
5. Include reposts from followed users
6. Sort by published_at DESC
7. Apply pagination (cursor-based recommended)
8. Return posts with engagement counts

Optimizations:
- Cache follower list
- Use database indexes on (author_id, published_at)
- Consider read replicas for heavy read operations
```

Explore Feed

```
Logic:
1. Fetch recent posts from public accounts
```

```
2. Exclude posts from users you follow (already in home feed)
3. Exclude blocked/muted users
4. Optionally: boost posts with higher engagement
5. Optionally: filter by interests/hashtags
6. Sort by published_at DESC or trending_score
7. Apply pagination

v1 Recommendation: Simple recent public posts
v2: Add trending/engagement scoring
```

List Feed

```
Logic:
1. Get list members
2. Fetch posts from list members only
3. Sort by published_at DESC
4. Apply pagination
```

Hashtag Feed

```
Logic:
1. Fetch posts containing specific hashtag
2. Sort by published_at DESC
3. Apply pagination
```

Privacy & Visibility Rules

Post Visibility

```
Rules:
1. Public posts: Visible to everyone
2. Followers-only posts: Visible only to followers
3. Private posts: Visible only to mentioned users (if implemented)

Checks before showing post:
- Is viewer blocked by author? → Hide
- Is author blocked by viewer? → Hide
- Is author muted by viewer? → Hide (optional, depends on UX)
- Is account private and viewer not a follower? → Hide
- Is post status = published? → Show
```

Follow Request Logic (Private Accounts)

```
Flow:
1. User A clicks "Follow" on User B (private account)
2. Create Follow record with status = "pending"
3. Send notification to User B
4. User B approves/declines request
5. If approved: status = "accepted", User A sees User B's posts
6. If declined: Delete Follow record
```

Engagement Counters

Counter Management Strategy

```
Approach: Cached Counters with Periodic Sync

Counters to cache:
- posts.likes_count
- posts.comments_count
- posts.reposts_count
- comments.likes_count
- comments.replies_count
- users.followers_count
- users.following_count
- users.posts_count

Update Strategy:
1. On engagement action (like, comment, etc.):
   - Increment/decrement counter in database (same transaction)
   - Return updated count to client

2. Consistency:
   - Use database triggers or application-level transactions
   - Periodic reconciliation job to fix drift (run nightly)

3. Performance:
   - Index counter columns for sorting/filtering
   - Consider Redis cache for high-traffic posts
```

Notification Logic

Notification Triggers

```
When to create notification:
1. User followed: follower → followee
2. Post liked: liker → post author (exclude self-likes)
3. Post commented: commenter → post author (exclude self-comments)
4. Comment replied: replier → parent comment author
5. User mentioned in post: post author → mentioned user
6. User mentioned in comment: commenter → mentioned user
7. User tagged in post: post author → tagged user
8. User tagged in image: tagger → tagged user
9. Post reposted: reposter → original post author
10. Post quote-posted: quote poster → original post author

Batch Notifications:
- "X and 10 others liked your post" (if multiple users like same post)
- Group similar notifications within time window
```

Notification Delivery

```
Channels:
1. In-app: Store in notifications table, show in notification center
2. Push: Send to mobile/browser if user has push enabled
3. Email: Send digest email based on user preferences

Delivery Logic:
- Check user's notification preferences
- Respect mute/block relationships
- Don't notify for actions on muted content
- Rate limit notifications per user (prevent spam)
```

Message Delivery

Real-time Messaging

```
Technology: WebSockets or Server-Sent Events (SSE)
```

```
Flow:
1. User connects to WebSocket server
2. Authenticate connection
3. Subscribe to user's conversation channels
4. On new message:
   - Persist to database
   - Broadcast to conversation participants via WebSocket
   - Send push notification if recipient offline
   - Update conversation.last_message_at

Message Status:
- Sent: Message persisted to database
- Delivered: Recipient connected and received message
- Read: Recipient opened conversation and viewed message
```

Product Context Messages

```
Flow:
1. User on product page clicks "Contact Seller"
2. Check if conversation exists between user and seller
3. If exists: Open existing conversation
4. If not: Create new conversation
5. Create message with:
   - message_type = "product_context"
   - product_id = current product ID
   - content = optional user text or default template
6. Conversation UI shows product banner at top
7. Product info (image, name, price) displayed persistently

Template:
"Hi, I'm interested in [Product Name]. Is this still available?"
```

Search Implementation

Full-Text Search

```
Technology Options:
1. PostgreSQL Full-Text Search (simple, built-in)
2. Elasticsearch (powerful, scalable, recommended for production)
```

```
3. Algolia (managed, fast, but paid)

Search Scope:
- Users: username, display_name, bio
- Posts: content, hashtags
- Products: name, description (if linked)
- Shops: name, description (if linked)
- Events: title, description (if linked)

Ranking Factors:
- Relevance score
- Recency (for posts)
- Engagement (likes, comments) for popular content
- User's network (prioritize followed users)

Implementation with Elasticsearch:
1. Index documents on create/update
2. Use multi-field search (content, hashtags, etc.)
3. Apply filters (date range, post type, engagement threshold)
4. Return results with highlights
```

Content Moderation

Automated Moderation (Future)

```
Techniques:
1. Keyword filtering (offensive words, spam patterns)
2. Image moderation (AI-based NSFW detection)
3. Rate limiting (prevent spam posting)
4. URL reputation checking (malicious links)

Implementation:
- Run checks on post creation
- Flag suspicious content for review
- Auto-hide or require manual approval
```

Manual Moderation

```
Workflow:
1. User reports content
```

```
2. Report enters moderation queue
3. Moderator reviews report
4. Actions:
   - Dismiss: No violation
   - Warn user: Send warning notification
   - Remove content: Soft delete post/comment
   - Suspend user: Temporary ban
   - Ban user: Permanent account ban

Tools:
- Admin dashboard for reports
- Content review interface
- User management panel
```

Integration Points

eCommerce Integration

Product Linking

```
Data Flow:
1. During post creation, user searches products
2. API call to Product Service: GET /api/products/search?q=:query
3. Return product list with: id, name, image, price, shop_id
4. User selects products to link
5. On post save, create PostProduct records
6. On post display, fetch product details and show product cards

Product Card Display:
- Product image (thumbnail)
- Product name
- Price
- Shop name
- "View Product" CTA button
- Click → Navigate to product detail page
```

Shop Mentions/Tags

```
Shop Mention Parsing ($shopname):
1. On post creation, parse content for $shopname pattern
2. Validate shop exists: Query Shop Service
3. Create PostShop records with mention_type = "mention"
4. On post display, convert $shopname to clickable link

Shop Tagging:
1. User explicitly tags shop (separate from content)
2. Create PostShop record with mention_type = "tag"
3. Shop sees "tagged posts" feed

Shop Profile Integration:
- Show posts where shop is tagged/mentioned
- "Posts about this shop" section on shop page
```

Event Linking

```
Data Flow:
1. During post creation, user searches events
2. API call to Event Service: GET /api/events/search?q=:query
3. Return event list with: id, title, date, location, image, ticket_status
4. User selects events to link
5. On post save, create PostEvent records
6. On post display, fetch event details and show event cards

Event Card Display:
- Event image
- Event title
- Date & time
- Location
- Ticket availability status
- "View Event" CTA button
- Click → Navigate to event detail page
```

Buy-Together Group Linking

```
Data Flow:
1. During post creation, user searches active buy-together groups (own or others they've
joined)
2. API call to BuyTogether Service: GET /api/buy-together/groups?status=active&user=:userId
```

```
3. Return group list with: id, product_info, current_count, total_slots, target_price,
expires_at
4. User selects group(s) to promote
5. On post save, create PostBuyTogetherGroup records
6. On post display, fetch real-time group status and show group cards

Group Card Display:
- Product image
- Product name with group label "Buy Together □□
- Progress bar: "3/5 spots filled"
- Target price with discount badge: "$800 (Save $100!)"
- Time remaining: "2 days left"
- Status indicator: Open/Almost Full/Full/Expired
- "Join Group" CTA button (disabled if full/expired)
- Click → Navigate to group detail page

Real-Time Updates:
- WebSocket connection for live group status updates
- When spots fill up, button changes to "Group Full"
- When time expires, entire card grays out with "Expired" badge
- Post feed auto-refreshes group status every 30 seconds

Boost Strategy:
User creates group → Shares on social feed → Friends see discount → Join group → Group fills
faster
```

Installment Plan Linking

```
Data Flow:
1. During post creation, user searches products with installment plans
2. API call to Product Service: GET /api/products/:id/installment-plans
3. Return available plans for product: id, duration_months, monthly_amount, total_cost,
interest_rate, down_payment
4. User selects which plan(s) to feature (can select multiple for comparison)
5. On post save, create PostInstallmentPlan records
6. On post display, fetch plan details and show plan cards

Single Plan Display:
- Product image
- Product name
```

```
- "Pay in installments □□ badge
- Monthly payment: "$99/month"
- Plan duration: "6 months"
- Total cost (if different from base price): "$594 total"
- Interest rate (if applicable): "0% interest" or "5% APR"
- "Buy with Plan" CTA button
- Click → Navigate to product page with plan pre-selected

Multiple Plans Comparison Display:
- Product image at top
- Side-by-side plan cards:

  [3 Months]        [6 Months]        [12 Months]
  $200/mo           $100/mo           $50/mo
  $600 total        $600 total        $650 total
  0% interest       0% interest       5% APR
  [Select]          [Select]          [Select]

- Each card clickable to product page with that plan

Marketing Use Cases:
- Shop owner: "Get this laptop for just $99/month! □□
- Influencer: "No need to pay all at once! Check out these flexible plans"
- User sharing deal: "Found this phone with 0% financing!"

Integration with Checkout:
- When user clicks "Buy with Plan"
- Deep link to product page: /products/:id?plan=:planId
- Checkout page has plan pre-selected
- User completes purchase with chosen installment plan
```

Notification Integration

Cross-Service Notifications

```
Scenario 1: User likes a post that has a linked product

Notification Recipients:
1. Post author: "X liked your post"
```

```
2. Product owner (if different from post author): "X liked a post featuring your product"

Scenario 2: User joins a buy-together group from a post

Notification Recipients:
1. Post author: "X joined your buy-together group from your post!"
2. Group creator (if different): "X joined your group"
3. Other group members: "X joined the group - 1 more spot to go!"

Scenario 3: Buy-together group gets filled from post shares

Notification to:
- Post author: "Your group is full! □□Your post helped fill 3 spots"
- All group members: "Group is full! Order will be placed soon"

Scenario 4: User clicks installment plan from post

Analytics Event (not push notification):
- Track which posts drive installment plan conversions
- Shop owner can see: "5 users viewed installment plans from your post"

Implementation:
- Social media service creates Like/Share/Join record
- Triggers notification service
- Notification service:
  - Checks PostProduct for linked products
  - Identifies product owner via Product Service
  - Creates notifications for both post author and product owner
```

Notifications System

Notification Types & Templates

Social Notifications

```
Type: FOLLOW
Template: "{actor_name} started following you"
Action: View {actor_name}'s profile
```

Type: LIKE_POST
Template: "{actor_name} liked your post"
Action: View post

Type: COMMENT_POST
Template: "{actor_name} commented on your post: '{comment_preview}'"
Action: View post/comment

Type: REPLY_COMMENT
Template: "{actor_name} replied to your comment: '{reply_preview}'"
Action: View comment thread

Type: MENTION_POST
Template: "{actor_name} mentioned you in a post"
Action: View post

Type: MENTION_COMMENT
Template: "{actor_name} mentioned you in a comment"
Action: View comment

Type: TAG_POST
Template: "{actor_name} tagged you in a post"
Action: View post

Type: TAG_IMAGE
Template: "{actor_name} tagged you in an image"
Action: View image/post

Type: REPOST
Template: "{actor_name} reposted your post"
Action: View repost

Type: QUOTE_POST
Template: "{actor_name} quoted your post: '{quote_preview}'"
Action: View quote post

Type: COLLABORATION_INVITE
Template: "{actor_name} invited you to collaborate on a post"
Action: View post to accept/decline

```
Type: COLLABORATION_ACCEPTED
Template: "{actor_name} accepted your collaboration invite"
Action: View collaborative post

Type: COLLABORATION_DECLINED
Template: "{actor_name} declined your collaboration invite"
Action: Dismiss notification

Type: COLLABORATOR_EDITED_POST
Template: "{actor_name} edited your collaborative post"
Action: View post changes
```

Messaging Notifications

```
Type: NEW_MESSAGE
Template: "New message from {actor_name}"
Action: View conversation

Type: PRODUCT_MESSAGE
Template: "{actor_name} messaged you about {product_name}"
Action: View conversation with product context
```

Commerce Notifications

```
Type: PRODUCT_POST_LIKE
Template: "{actor_name} liked a post featuring your product '{product_name}'"
Action: View post

Type: PRODUCT_POST_COMMENT
Template: "{actor_name} commented on a post featuring your product"
Action: View post/comment

Type: SHOP_TAGGED
Template: "{actor_name} tagged your shop in a post"
Action: View post
```

Notification Batching

Batching Strategy

```
Rules:
- Same type + same entity within 5 minutes → Batch
- Max 100 actors in batch
- Update existing notification instead of creating new

Example:
Instead of:
- "Alice liked your post"
- "Bob liked your post"
- "Charlie liked your post"

Show:
- "Alice, Bob, Charlie and 10 others liked your post"

Implementation:
- Use transaction to check for recent similar notifications
- Update existing notification's actor list
- Increment count
- Update timestamp to latest
```

Notification Preferences

Preference Options

```
Categories:
1. Social Interactions
   - Follows
   - Likes
   - Comments
   - Mentions
   - Tags
   - Reposts
   - Collaborations

2. Messages
   - Direct messages
   - Message reactions

3. Commerce
```

```
   - Product interactions
   - Shop tags

Settings per category:
- Push: ON/OFF
- In-app: ON/OFF (always ON for critical notifications)
- Email: ON/OFF/DIGEST (daily/weekly)
- Frequency: All / Important only / None

Default:
- All push notifications: ON
- All in-app: ON
- Email: DIGEST (daily)
```

Data Privacy

Sensitive Data Handling

```
Personal Information:
- Email: Never exposed in API responses (except to owner)
- Password: Always hashed (bcrypt, Argon2)
- Profile: Respect account privacy (public/private)

Data Retention:
- Deleted posts: Soft delete, retain for 30 days, then hard delete
- Deleted messages: Soft delete, retain for 90 days for legal compliance
- Deleted accounts: Anonymize after 30 days, hard delete after 90 days

Export & Deletion (GDPR):
- User data export: Provide JSON export of all user data
- Right to be forgotten: Delete all user content and personal info
```

Privacy Controls

Block Functionality

```
When User A blocks User B:
- B cannot see A's profile or posts
```

```
- B cannot follow A
- B cannot message A
- B cannot comment on A's posts
- B cannot tag or mention A
- Existing follows are removed
- B is not notified of the block

Implementation:
- Before showing content, check if viewer is blocked
- Before allowing action, check if actor is blocked
```

Mute Functionality

```
When User A mutes User B:
- B's posts don't appear in A's feed
- B's comments on others' posts may still appear (optional)
- B is not notified of the mute
- A can still see B's profile if directly visited

Implementation:
- Filter out muted users in feed queries
- Store mute relationship in Mute table
```

Conclusion

This documentation provides a complete blueprint for implementing a social media platform integrated with eCommerce and events. The architecture is designed to scale from MVP to millions of users while maintaining code quality and user experience.

Revision #4
Created 3 December 2025 06:23:35 by Admin Qbit
Updated 11 December 2025 08:39:45 by Admin Qbit