ImageFlow - Service Layer Design

Author: Sinem Eissler
Date: July 18, 2025

---

1. Service Architecture Overview

ImageFlow's service layer is built entirely with custom Node.js/Express code, implementing all functionality without relying on third-party services. The API follows RESTful principles with WebSocket support for real-time features.

Base URL: `https://api.imageflow.app/v1`
WebSocket URL: `wss://api.imageflow.app/ws`

Authentication: Custom JWT implementation with refresh tokens

```
Authorization: Bearer <access-token>
X-Refresh-Token: <refresh-token>
```

---

2. Authentication Service Endpoints (Custom JWT Implementation)

POST /auth/register

Purpose: Create new user account with custom validation and hashing.

Custom Implementation Details:

- Password strength validation algorithm
- Custom email verification token generation
- Bcrypt hashing with configurable rounds

Request:

```
json
```

```
{
  "email": "sarah.chen@example.com",
  "password": "SecurePass123!",
  "username": "sarahchen",
  "displayName": "Sarah Chen"
}
```

Success Response (201 Created):

```
json

{
  "user": {
    "userId": "usr_a1b2c3d4e5",
    "username": "sarahchen",
    "displayName": "Sarah Chen",
    "avatarUrl": null,
    "followerCount": 0,
    "followingCount": 0
  },
  "tokens": {
    "accessToken": "eyJhbGciOiJIUzI1NiIs...",
    "refreshToken": "rf_k2l3m4n5o6p7q8r9",
    "expiresIn": 3600
  },
  "verificationRequired": true
}
```

Error Response (400 Bad Request):

```
json

{
  "error": "VALIDATION_ERROR",
  "message": "Password must contain at least one uppercase letter, one number, and one special character",
  "field": "password",
  "code": "WEAK_PASSWORD"
}
```

POST /auth/login

Purpose: Authenticate user with custom session management.

Custom Implementation Details:

- Rate limiting with exponential backoff
- Device fingerprinting for security
- Concurrent session management

Request:

```json
{
  "username": "sarahchen",
  "password": "SecurePass123!",
  "deviceInfo": {
    "userAgent": "Mozilla/5.0...",
    "timezone": "America/Los_Angeles"
  }
}
```

POST /auth/refresh

Purpose: Refresh access token using custom token rotation.

POST /auth/logout

Purpose: Invalidate tokens and clear session data.

---

3. Image Management Endpoints (Custom Processing)

POST /images/upload

Purpose: Handle image upload with custom processing pipeline.

Custom Implementation Details:

- Chunked upload handling
- Custom file validation
- Virus scanning implementation
- Initial processing queue

Request (Multipart Form Data):

```
Content-Type: multipart/form-data
---
file: [binary data]
metadata: {
  "title": "Sunset at Beach",
  "description": "Beautiful sunset captured in Hawaii",
  "tags": ["sunset", "beach", "vacation"],
  "privacy": "public"
}
```

Success Response (202 Accepted):

```
json

{
  "imageId": "img_f6g7h8i9j0",
  "uploadStatus": "processing",
  "processingSteps": [
    {"step": "upload", "status": "complete"},
    {"step": "validation", "status": "complete"},
    {"step": "thumbnail", "status": "pending"},
    {"step": "analysis", "status": "pending"},
    {"step": "indexing", "status": "pending"}
  ],
  "estimatedTime": 15,
  "websocketChannel": "processing_img_f6g7h8i9j0"
}
```

GET /images/{imageId}

Purpose: Retrieve image with edit history and social metadata.

Success Response (200 OK):

```
json
```

```
{
  "image": {
    "imageId": "img_f6g7h8i9j0",
    "ownerId": "usr_a1b2c3d4e5",
    "title": "Sunset at Beach",
    "description": "Beautiful sunset captured in Hawaii",
    "uploadedAt": "2025-07-18T10:30:00Z",
    "lastEditedAt": "2025-07-18T14:20:00Z",
    "dimensions": {
      "width": 3000,
      "height": 2000
    },
    "fileSize": 2048576,
    "format": "jpeg",
    "urls": {
      "original": "/images/img_f6g7h8i9j0/original",
      "display": "/images/img_f6g7h8i9j0/display",
      "thumbnail": "/images/img_f6g7h8i9j0/thumbnail",
      "edit": "/images/img_f6g7h8i9j0/edit"
    },
    "privacy": "public",
    "stats": {
      "views": 156,
      "likes": 23,
      "comments": 7,
      "shares": 3
    },
    "tags": ["sunset", "beach", "vacation", "hawaii"],
    "colorPalette": ["#FF6B35", "#FFA500", "#87CEEB", "#FFD700"],
    "histogram": {
      "red": [/* 256 values */],
      "green": [/* 256 values */],
      "blue": [/* 256 values */]
    }
  },
  "owner": {
    "userId": "usr_a1b2c3d4e5",
    "username": "sarahchen",
    "displayName": "Sarah Chen",
    "avatarUrl": "/users/sarahchen/avatar"
  },
  "editHistory": {
    "totalEdits": 3,
```

```
    "currentVersion": "v3",
    "canEdit": true
  }
}
```

POST /images/{imageId}/edit

Purpose: Create new edit session with custom canvas state management.

Request:

```
json

{
  "editType": "collaborative",
  "invitedUsers": ["marcusrodriguez", "emmathompson"],
  "permissions": {
    "allowDownload": true,
    "allowFork": true,
    "expiresIn": 3600
  }
}
```

Success Response (201 Created):

```
json

{
  "sessionId": "edit_k1l2m3n4o5",
  "sessionUrl": "/edit/edit_k1l2m3n4o5",
  "websocketChannel": "edit_session_k1l2m3n4o5",
  "participants": [],
  "expiresAt": "2025-07-18T15:30:00Z"
}
```

---

4. Canvas Editor Endpoints (Custom Implementation)

GET /images/{imageId}/canvas

Purpose: Load canvas state with layers and edit history.

Success Response (200 OK):

json


```
{
  "canvas": {
    "width": 3000,
    "height": 2000,
    "backgroundColor": "#FFFFFF",
    "layers": [
      {
        "layerId": "layer_base",
        "type": "image",
        "name": "Original",
        "visible": true,
        "opacity": 1.0,
        "blendMode": "normal",
        "locked": true,
        "data": "base64_encoded_image_data"
      },
      {
        "layerId": "layer_adj_1",
        "type": "adjustment",
        "name": "Brightness/Contrast",
        "visible": true,
        "opacity": 0.8,
        "adjustments": {
          "brightness": 10,
          "contrast": 15
        }
      }
    ],
    "history": {
      "currentIndex": 5,
      "maxIndex": 8,
      "canUndo": true,
      "canRedo": true
    }
  }
}
```


POST /images/{imageId}/canvas/layers

Purpose: Add new layer with custom rendering.

Request:

```json
{
  "layerType": "drawing",
  "name": "Annotations",
  "blendMode": "multiply",
  "opacity": 0.7
}
```

PUT /images/{imageId}/canvas/layers/{layerId}

Purpose: Update layer properties or data.

POST /images/{imageId}/canvas/transform

Purpose: Apply custom transformation matrix.

Request:

```json
{
  "layerId": "layer_1",
  "transformation": {
    "type": "rotate",
    "angle": 45,
    "center": {"x": 1500, "y": 1000},
    "interpolation": "bilinear"
  }
}
```

POST /images/{imageId}/canvas/filters

Purpose: Apply custom filter implementation.

Request:

```json
```

```
{
  "layerId": "layer_1",
  "filter": {
    "type": "convolution",
    "kernel": [
      [-1, -1, -1],
      [-1,  9, -1],
      [-1, -1, -1]
    ],
    "divisor": 1,
    "offset": 0
  }
}
```

---

5. Social Networking Endpoints (Custom Graph Implementation)

POST /users/{username}/follow

Purpose: Create follow relationship in custom social graph.

Custom Implementation Details:

- Bidirectional relationship management
- Activity feed population
- Notification generation

Success Response (201 Created):

```
json
```

```
{
  "relationship": {
    "following": true,
    "followedBy": false,
    "followedAt": "2025-07-18T10:30:00Z",
    "notificationsEnabled": true
  },
  "user": {
    "username": "emmathompson",
    "displayName": "Dr. Emma Thompson",
    "followerCount": 542,
    "isVerified": true
  }
}
```


GET /users/{username}/feed

Purpose: Get personalized activity feed using custom algorithm.

Custom Implementation Details:

- Relevance scoring based on interactions
- Time decay function
- Content diversity optimization

Request Parameters:

```
?limit=20&before=2025-07-18T10:00:00Z&filter=images,edits
```

Success Response (200 OK):

```
json
```

```
{
  "feed": [
    {
      "activityId": "act_p0q1r2s3t4",
      "type": "image_upload",
      "actor": {
        "username": "marcusrodriguez",
        "displayName": "Marcus Rodriguez"
      },
      "action": "uploaded a new image",
      "target": {
        "type": "image",
        "imageId": "img_u5v6w7x8y9",
        "title": "Mountain Landscape",
        "thumbnailUrl": "/images/img_u5v6w7x8y9/thumbnail"
      },
      "timestamp": "2025-07-18T09:45:00Z",
      "relevanceScore": 0.92
    }
  ],
  "nextCursor": "eyJiZWZvcmUiOiIyMDI1LTA3LTE4VDA5OjAwOjAwWiJ9"
}
```


GET /users/{username}/followers

Purpose: Get followers list with pagination.

GET /users/{username}/following

Purpose: Get following list with pagination.

---

6. Real-time Collaboration Endpoints (WebSocket)

WS /collaboration/join

Purpose: Join collaborative editing session.

Message Format:

```
json
```

```
{
  "type": "join_session",
  "sessionId": "edit_k1l2m3n4o5",
  "token": "Bearer eyJhbGciOiJIUzI1NiIs..."
}
```

Response:

```
json

{
  "type": "session_joined",
  "sessionId": "edit_k1l2m3n4o5",
  "participants": [
    {
      "userId": "usr_a1b2c3d4e5",
      "username": "sarahchen",
      "cursorColor": "#FF6B35",
      "isOwner": true
    }
  ],
  "canvasState": {/* current canvas state */}
}
```

WS /collaboration/cursor

Purpose: Broadcast cursor position for real-time collaboration.

Message Format:

```
json

{
  "type": "cursor_move",
  "position": {"x": 1250, "y": 750},
  "tool": "brush",
  "layerId": "layer_2"
}
```

WS /collaboration/operation

Purpose: Send canvas operation for operational transformation.

Message Format:

```
json

{
  "type": "canvas_operation",
  "operation": {
    "op": "stroke",
    "layerId": "layer_2",
    "path": [
      {"x": 100, "y": 100, "pressure": 0.5},
      {"x": 110, "y": 105, "pressure": 0.7}
    ],
    "brush": {
      "size": 10,
      "color": "#000000",
      "opacity": 0.8
    }
  },
  "revision": 42
}
```

---

7. Search Engine Endpoints (Custom Implementation)

GET /search/images

Purpose: Full-text search with custom ranking algorithm.

Custom Implementation Details:

- TF-IDF scoring
- User preference weighting
- Visual similarity matching
- Faceted search support

Request Parameters:

```
?q=sunset beach&tags=vacation,hawaii&color=orange&user=sarahchen&sort=relevance
```

Success Response (200 OK):

```
json
```

```
{
  "query": {
    "original": "sunset beach",
    "parsed": {
      "terms": ["sunset", "beach"],
      "filters": {
        "tags": ["vacation", "hawaii"],
        "dominantColor": "orange",
        "owner": "sarahchen"
      }
    }
  },
  "results": [
    {
      "imageId": "img_f6g7h8i9j0",
      "score": 0.95,
      "highlights": {
        "title": "<em>Sunset</em> at <em>Beach</em>",
        "tags": ["<em>sunset</em>", "<em>beach</em>", "vacation", "hawaii"]
      },
      "thumbnail": "/images/img_f6g7h8i9j0/thumbnail",
      "owner": {
        "username": "sarahchen",
        "displayName": "Sarah Chen"
      }
    }
  ],
  "facets": {
    "tags": [
      {"value": "sunset", "count": 45},
      {"value": "beach", "count": 38}
    ],
    "colors": [
      {"value": "orange", "count": 23},
      {"value": "blue", "count": 19}
    ]
  },
  "totalResults": 127,
  "processingTime": 0.045
}
```


POST /search/index

Purpose: Manually trigger re-indexing for search optimization.

---

8. Version Control Endpoints (Custom Implementation)

GET /images/{imageId}/versions

Purpose: Get edit history with branching visualization.

Success Response (200 OK):

```
json
```

```
{
  "versions": [
    {
      "versionId": "ver_a1b2c3",
      "versionNumber": 1,
      "parentVersion": null,
      "createdAt": "2025-07-18T10:30:00Z",
      "createdBy": "sarahchen",
      "description": "Original upload",
      "thumbnail": "/images/img_f6g7h8i9j0/versions/ver_a1b2c3/thumbnail",
      "stats": {
        "layers": 1,
        "fileSize": 2048576
      }
    },
    {
      "versionId": "ver_d4e5f6",
      "versionNumber": 2,
      "parentVersion": "ver_a1b2c3",
      "createdAt": "2025-07-18T11:00:00Z",
      "createdBy": "sarahchen",
      "description": "Applied sunset enhancement filter",
      "changes": [
        {"type": "filter", "description": "Warm tone adjustment"},
        {"type": "adjustment", "description": "Increased contrast by 15%"}
      ]
    }
  ],
  "branches": [
    {
      "branchId": "branch_exp_1",
      "name": "Experimental edits",
      "baseVersion": "ver_d4e5f6",
      "createdBy": "marcusrodriguez",
      "isActive": true
    }
  ]
}
```


POST /images/{imageId}/versions/{versionId}/fork

Purpose: Create new branch from specific version.

POST /images/{imageId}/versions/compare

Purpose: Generate visual diff between versions.

Request:

```json
{
  "versionA": "ver_a1b2c3",
  "versionB": "ver_d4e5f6",
  "diffType": "visual"
}
```

---

9. Notification Service Endpoints

GET /notifications

Purpose: Get user notifications with custom priority sorting.

POST /notifications/{notificationId}/read

Purpose: Mark notification as read.

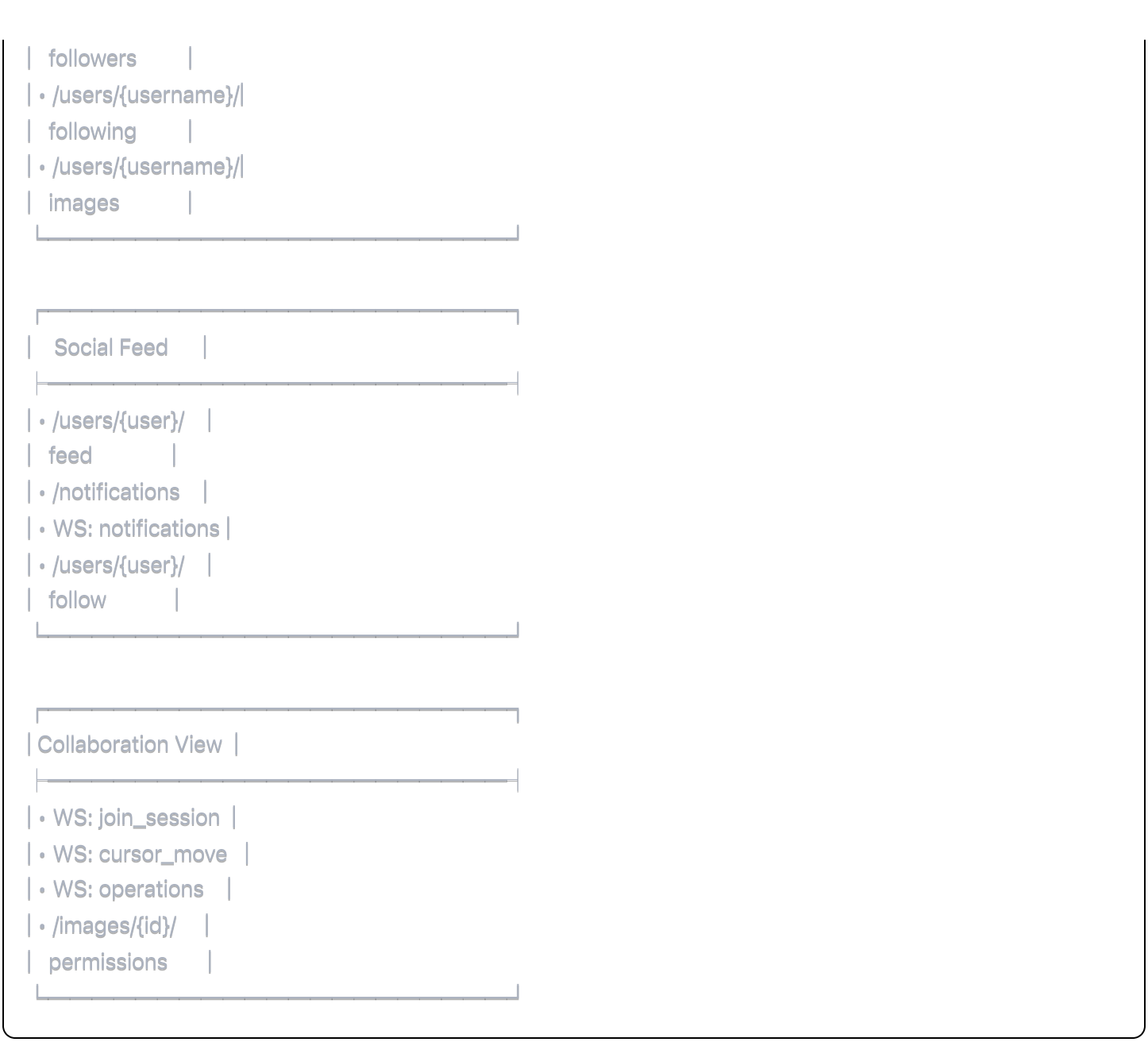
WS /notifications/subscribe

Purpose: Real-time notification delivery via WebSocket.

---

10. Page-to-Endpoint Mapping Diagram


Landing Page
• /auth/register
• /auth/login
• /search/images
• /users/trending
Upload Page
• /images/upload
• WS: processing updates
Gallery View
• /users/{user}/ images
• /images (bulk)
• /search/images
Canvas Editor
• /images/{id}/ canvas
• /images/{id}/ canvas/*
• WS: collaboration
• /images/{id}/ versions
User Profile
• /users/{username}
• /users/{username}/



---

11. Custom Implementation Details

Authentication & Security

- **JWT Library**: Custom implementation with RS256 signing
- **Rate Limiting**: Token bucket algorithm with Redis
- **CSRF Protection**: Double submit cookie pattern
- **Input Validation**: Custom schema validation engine

Image Processing

- **Thumbnail Algorithm**: Smart crop using entropy detection
- **Filter Engine**: Custom convolution matrix implementation

- **Color Analysis**: K-means clustering for palette extraction
- **Compression**: Progressive JPEG encoding

Real-time Features

- **WebSocket Server**: Custom implementation with Socket.io
- **Operational Transform**: Custom OT algorithm for concurrent edits
- **Presence System**: Efficient cursor tracking and broadcasting
- **Message Queue**: Priority queue for notification delivery

Search & Discovery

- **Indexing**: Inverted index with term frequency analysis
- **Ranking**: Custom PageRank-inspired algorithm for images
- **Autocomplete**: Trie data structure implementation
- **Visual Search**: Perceptual hash comparison

Performance Optimizations

- **Caching Layer**: LRU cache implementation
- **Connection Pooling**: Custom database connection manager
- **Image CDN**: Smart routing and caching logic
- **Lazy Loading**: Progressive image loading implementation

---

12. Error Handling

All endpoints follow consistent error response format with custom error codes:

```
json
```

```
{
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid image format. Supported formats: JPEG, PNG, WebP",
    "details": {
      "field": "file",
      "received": "image/bmp",
      "allowed": ["image/jpeg", "image/png", "image/webp"]
    },
    "timestamp": "2025-07-18T10:30:00Z",
    "requestId": "req_x9y8z7w6v5"
  }
}
```


Custom Error Codes

- **AUTH_ERRORS**: Invalid credentials, expired tokens, insufficient permissions
- **VALIDATION_ERRORS**: Input validation failures, schema violations
- **PROCESSING_ERRORS**: Image processing failures, timeout errors
- **COLLABORATION_ERRORS**: Session conflicts, synchronization issues
- **RATE_LIMIT_ERRORS**: Quota exceeded, throttling active

HTTP Status Codes

- **200 OK**: Successful GET/PUT requests
- **201 Created**: Successful POST creating new resources
- **202 Accepted**: Async processing initiated
- **204 No Content**: Successful DELETE operations
- **400 Bad Request**: Validation errors
- **401 Unauthorized**: Authentication required
- **403 Forbidden**: Insufficient permissions
- **404 Not Found**: Resource doesn't exist
- **409 Conflict**: Concurrency conflicts
- **413 Payload Too Large**: File size limit exceeded
- **429 Too Many Requests**: Rate limit exceeded
- **500 Internal Server Error**: Server-side errors

---

13. Implementation Notes

1. **No External Services**: All functionality implemented in custom Node.js code
2. **Database**: PostgreSQL with custom query builders and connection pooling
3. **File Storage**: Local filesystem with custom CDN implementation
4. **Caching**: In-memory caching with custom LRU implementation
5. **Message Queue**: Custom implementation for async processing
6. **Monitoring**: Custom metrics collection and logging
7. **Testing**: Comprehensive unit and integration tests for all endpoints

This service layer provides a complete API surface for the ImageFlow application, with every feature implemented through custom code rather than third-party services.