CiviFix API Contract

Application Overview

CiviFix is a civic engagement platform where citizens can report local issues, track their status, and engage with their community and local government.

Core Features

- User registration and authentication
- Report civic issues (potholes, broken streetlights, etc.)
- View and search issues
- Comment on issues
- Vote/support issues
- Track issue status updates
- User profiles and dashboards

Data Models

User Model

json

```
{
  "id": "integer",
  "username": "string",
  "email": "string",
  "full_name": "string",
  "phone": "string (optional)",
  "address": "string (optional)",
  "created_at": "datetime",
  "updated_at": "datetime"
}
```

Issue Model

json

```
{
  "id": "integer",
  "title": "string",
  "description": "string",
  "category": "string",
  "status": "string (open/in_progress/resolved/closed)",
  "priority": "string (low/medium/high/urgent)",
  "location": {
    "address": "string",
    "latitude": "float",
    "longitude": "float"
  },
  "images": ["array of image URLs"],
  "author_id": "integer",
  "assigned_to": "integer (optional)",
  "vote_count": "integer",
  "created_at": "datetime",
  "updated_at": "datetime"
}
```

Comment Model

json

```
{
  "id": "integer",
  "content": "string",
  "issue_id": "integer",
  "user_id": "integer",
  "created_at": "datetime",
  "updated_at": "datetime"
}
```

Vote Model

json

```
{
  "id": "integer",
  "user_id": "integer",
  "issue_id": "integer",
  "created_at": "datetime"
}
```

API Endpoints

1. User Registration

Feature: Register a new user

HTTP Method: POST

Endpoint Path: `/api/auth/register`

Description: Creates a new user account

Request Body:

```
json

{
  "username": "johndoe",
  "email": "john@example.com",
  "password": "securepassword123",
  "full_name": "John Doe",
  "phone": "+1234567890"
}
```

Success Response (201 Created):

```
json

{
  "message": "User registered successfully",
  "user": {
    "id": 1,
    "username": "johndoe",
    "email": "john@example.com",
    "full_name": "John Doe",
    "created_at": "2024-01-15T10:30:00Z"
  },
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Error Response (400 Bad Request):

```
json

{
  "error": "Registration failed",
  "message": "Email already exists"
}
```

---

2. User Login

Feature: Authenticate user

HTTP Method: POST

Endpoint Path: `/api/auth/login`

Description: Authenticates user and returns access token

Request Body:

```json
{
  "email": "john@example.com",
  "password": "securepassword123"
}
```

Success Response (200 OK):

```json
{
  "message": "Login successful",
  "user": {
    "id": 1,
    "username": "johndoe",
    "email": "john@example.com",
    "full_name": "John Doe"
  },
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Error Response (401 Unauthorized):

```json
{
  "error": "Authentication failed",
  "message": "Invalid email or password"
}
```

---

3. Create Issue

Feature: Report a new civic issue

HTTP Method: POST

Endpoint Path: `/api/issues`

Description: Creates a new issue report

Request Body:

```json
{
  "title": "Pothole on Main Street",
  "description": "Large pothole causing traffic issues",
  "category": "road_maintenance",
  "priority": "medium",
  "location": {
    "address": "123 Main St, City, State",
    "latitude": 40.7128,
    "longitude": -74.0060
  },
  "images": ["image1.jpg", "image2.jpg"]
}
```

Success Response (201 Created):

```json
{
  "message": "Issue created successfully",
  "issue": {
    "id": 1,
    "title": "Pothole on Main Street",
    "description": "Large pothole causing traffic issues",
    "category": "road_maintenance",
    "status": "open",
    "priority": "medium",
    "location": {
      "address": "123 Main St, City, State",
      "latitude": 40.7128,
      "longitude": -74.0060
    },
    "images": ["image1.jpg", "image2.jpg"],
    "author_id": 1,
    "vote_count": 0,
    "created_at": "2024-01-15T10:30:00Z"
  }
}
```

Error Response (400 Bad Request):

```json
```

```
{
  "error": "Issue creation failed",
  "message": "Title and description are required"
}
```

---

4. Get All Issues

Feature: Retrieve list of all issues
HTTP Method: GET
Endpoint Path: `/api/issues`
Description: Retrieves a paginated list of all issues

Query Parameters:

- `page` (optional): Page number (default: 1)
- `limit` (optional): Items per page (default: 10)
- `category` (optional): Filter by category
- `status` (optional): Filter by status

Success Response (200 OK):

```
json
```

```
{
  "issues": [
    {
      "id": 1,
      "title": "Pothole on Main Street",
      "description": "Large pothole causing traffic issues",
      "category": "road_maintenance",
      "status": "open",
      "priority": "medium",
      "location": {
        "address": "123 Main St, City, State",
        "latitude": 40.7128,
        "longitude": -74.0060
      },
      "vote_count": 5,
      "author_id": 1,
      "created_at": "2024-01-15T10:30:00Z"
    }
  ],
  "pagination": {
    "current_page": 1,
    "total_pages": 10,
    "total_items": 100,
    "items_per_page": 10
  }
}
```


Error Response (500 Internal Server Error):


```
json

{
  "error": "Server error",
  "message": "Unable to retrieve issues"
}
```


---

5. Get Single Issue

Feature: Get detailed information about a specific issue

HTTP Method: GET

Endpoint Path: `/api/issues/{id}`

Description: Retrieves detailed information about a single issue

Success Response (200 OK):

```json
{
  "issue": {
    "id": 1,
    "title": "Pothole on Main Street",
    "description": "Large pothole causing traffic issues",
    "category": "road_maintenance",
    "status": "open",
    "priority": "medium",
    "location": {
      "address": "123 Main St, City, State",
      "latitude": 40.7128,
      "longitude": -74.0060
    },
    "images": ["image1.jpg", "image2.jpg"],
    "author_id": 1,
    "vote_count": 5,
    "created_at": "2024-01-15T10:30:00Z",
    "updated_at": "2024-01-15T10:30:00Z"
  }
}
```

Error Response (404 Not Found):

```json
{
  "error": "Issue not found",
  "message": "Issue with ID 1 does not exist"
}
```

---

6. Update Issue Status

Feature: Update the status of an issue (admin/government users)
HTTP Method: PUT
Endpoint Path: `/api/issues/{id}/status`
Description: Updates the status of an existing issue

Request Body:

```json
```

```
{
  "status": "in_progress",
  "update_message": "Work has begun on this issue"
}
```

Success Response (200 OK):

```
json

{
  "message": "Issue status updated successfully",
  "issue": {
    "id": 1,
    "status": "in_progress",
    "updated_at": "2024-01-16T09:15:00Z"
  }
}
```

Error Response (403 Forbidden):

```
json

{
  "error": "Access denied",
  "message": "You don't have permission to update this issue"
}
```

---

7. Vote on Issue

Feature: Support an issue by voting

HTTP Method: POST

Endpoint Path: `/api/issues/{id}/vote`

Description: Allows users to vote/support an issue

Request Body:

```
json

{}
```

Success Response (201 Created):

```
json
```

```
{
  "message": "Vote added successfully",
  "vote_count": 6
}
```

Error Response (409 Conflict):

```
json

{
  "error": "Vote failed",
  "message": "You have already voted on this issue"
}
```

---

8. Remove Vote

Feature: Remove vote from an issue

HTTP Method: DELETE

Endpoint Path: `/api/issues/{id}/vote`

Description: Removes user's vote from an issue

Success Response (200 OK):

```
json

{
  "message": "Vote removed successfully",
  "vote_count": 5
}
```

Error Response (404 Not Found):

```
json

{
  "error": "Vote removal failed",
  "message": "You haven't voted on this issue"
}
```

---

9. Add Comment to Issue

Feature: Comment on an issue

HTTP Method: POST

Endpoint Path: `/api/issues/{id}/comments`
Description: Adds a comment to an existing issue

Request Body:

```
json

{
  "content": "I've also noticed this issue. It's getting worse daily."
}
```

Success Response (201 Created):

```
json

{
  "message": "Comment added successfully",
  "comment": {
    "id": 1,
    "content": "I've also noticed this issue. It's getting worse daily.",
    "issue_id": 1,
    "user_id": 2,
    "created_at": "2024-01-15T11:00:00Z"
  }
}
```

Error Response (400 Bad Request):

```
json

{
  "error": "Comment creation failed",
  "message": "Comment content cannot be empty"
}
```

---

10. Get Comments for Issue

Feature: Retrieve all comments for an issue
HTTP Method: GET
Endpoint Path: `/api/issues/{id}/comments`
Description: Retrieves all comments for a specific issue

Success Response (200 OK):

```
json
```

```
{
  "comments": [
    {
      "id": 1,
      "content": "I've also noticed this issue. It's getting worse daily.",
      "issue_id": 1,
      "user_id": 2,
      "user": {
        "username": "janedoe",
        "full_name": "Jane Doe"
      },
      "created_at": "2024-01-15T11:00:00Z"
    }
  ]
}
```


Error Response (404 Not Found):


```
json

{
  "error": "Issue not found",
  "message": "Issue with ID 1 does not exist"
}
```


---

11. Get User Profile

Feature: Retrieve user profile information

HTTP Method: GET

Endpoint Path: `/api/users/{id}`

Description: Retrieves public profile information for a user

Success Response (200 OK):


```
json
```

```
{
  "user": {
    "id": 1,
    "username": "johndoe",
    "full_name": "John Doe",
    "created_at": "2024-01-15T10:30:00Z",
    "issues_created": 5,
    "issues_voted": 12
  }
}
```


Error Response (404 Not Found):


```
json

{
  "error": "User not found",
  "message": "User with ID 1 does not exist"
}
```


---

12. Update User Profile

Feature: Update user's own profile

HTTP Method: PUT

Endpoint Path: `/api/users/profile`

Description: Updates the authenticated user's profile

Request Body:


```
json

{
  "full_name": "John Smith",
  "phone": "+1234567890",
  "address": "456 Oak St, City, State"
}
```


Success Response (200 OK):


```
json
```

```
{
  "message": "Profile updated successfully",
  "user": {
    "id": 1,
    "username": "johndoe",
    "email": "john@example.com",
    "full_name": "John Smith",
    "phone": "+1234567890",
    "address": "456 Oak St, City, State",
    "updated_at": "2024-01-16T09:30:00Z"
  }
}
```


Error Response (400 Bad Request):

json


```
{
  "error": "Profile update failed",
  "message": "Invalid phone number format"
}
```


---

13. Search Issues

Feature: Search issues by keyword

HTTP Method: GET

Endpoint Path: `/api/issues/search`

Description: Searches issues based on title, description, or location

Query Parameters:

- `q`: Search query (required)
- `category` (optional): Filter by category
- `status` (optional): Filter by status

Success Response (200 OK):

json

```
{
  "results": [
    {
      "id": 1,
      "title": "Pothole on Main Street",
      "description": "Large pothole causing traffic issues",
      "category": "road_maintenance",
      "status": "open",
      "location": {
        "address": "123 Main St, City, State"
      },
      "vote_count": 5,
      "created_at": "2024-01-15T10:30:00Z"
    }
  ],
  "total_results": 1
}
```


Error Response (400 Bad Request):


```
json

{
  "error": "Search failed",
  "message": "Search query parameter 'q' is required"
}
```

Authentication

Most endpoints require authentication via JWT token in the Authorization header:

```
Authorization: Bearer {your-jwt-token}
```

Error Handling

All error responses follow this structure:


```
json

{
  "error": "Error category",
  "message": "Detailed error message"
}
```

Status Codes Used

- `200 OK`: Successful GET, PUT, DELETE requests
- `201 Created`: Successful POST requests
- `400 Bad Request`: Invalid request data
- `401 Unauthorized`: Authentication required or failed
- `403 Forbidden`: Insufficient permissions
- `404 Not Found`: Resource not found
- `409 Conflict`: Resource conflict (e.g., duplicate vote)
- `500 Internal Server Error`: Server-side error

Issue Categories

- `road_maintenance`
- `street_lighting`
- `waste_management`
- `water_drainage`
- `public_safety`
- `parks_recreation`
- `traffic_signals`
- `sidewalks`
- `other`

Issue Statuses

- `open`: Newly reported, not yet reviewed
- `in_progress`: Being worked on by authorities
- `resolved`: Issue has been fixed
- `closed`: Issue closed without resolution (duplicate, invalid, etc.)

This API contract serves as the single source of truth for all frontend-backend communication in the CiviFix application.