BountyHunter API Documentation

This document outlines all the backend API routes implemented in the BountyHunter application.

Authentication Routes

POST /api/signup

Creates a new user account with crypto wallet authentication.

Request Body:

```
{
  "hotkey": "string (required) - Crypto wallet address",
  "username": "string (optional) - Display name",
  "password": "string (required) - Account password"
}
```

Response:

```
{
  "message": "User created successfully",
  "user": {
    "id": "string",
    "hotkey": "string",
    "username": "string",
    "createdAt": "datetime"
  }
}
```

POST /api/auth/[...nextauth]

NextAuth.js authentication endpoints for login/logout using credentials provider.

Credentials:
- `hotkey` : Crypto wallet address
- `password` : Account password

Bounty Management Routes

GET /api/bounties

Retrieves all bounties with optional filtering.

Query Parameters:
- `status` (optional): Filter by bounty status (ACTIVE, COMPLETED, PAUSED, CANCELLED)
- `category` (optional): Filter by category name
- `limit` (optional): Number of results (default: 20)
- `offset` (optional): Pagination offset (default: 0)

Response:

```
{
  "bounties": [
    {
      "id": "string",
      "title": "string",
      "description": "string",
      "requirements": "string",
      "alphaReward": "string",
      "alphaRewardCap": "string",
      "rewardDistribution": "ALL_AT_ONCE | OVER_TIME",
      "winningSpots": "number",
      "status": "string",
      "deadline": "datetime",
      "creator": {
        "id": "string",
        "username": "string",
        "hotkey": "string"
      },
      "categories": [
        {
          "id": "string",
          "name": "string",
          "color": "string"
        }
      ],
      "submissionCount": "number"
    }
  ]
}
```

POST /api/bounties

Creates a new bounty (requires authentication).

Request Body:

```
{
  "title": "string (required)",
  "description": "string (required)",
  "requirements": "string (required)",
  "alphaReward": "number (required)",
  "alphaRewardCap": "number (required)",
  "rewardDistribution": "ALL_AT_ONCE | OVER_TIME",
  "winningSpots": "number",
  "deadline": "datetime (optional)"
}
```

GET /api/bounties/[id]

Retrieves a specific bounty with submissions and detailed information.

Response:

```
{
  "bounty": {
    "id": "string",
    "title": "string",
    "description": "string",
    "requirements": "string",
    "alphaReward": "string",
    "alphaRewardCap": "string",
    "rewardDistribution": "string",
    "winningSpots": "number",
    "status": "string",
    "deadline": "datetime",
    "creator": {
      "id": "string",
      "username": "string",
      "hotkey": "string"
    },
    "submissions": [
      {
        "id": "string",
        "title": "string",
        "description": "string",
        "status": "string",
        "score": "string",
        "submitter": {
          "id": "string",
          "username": "string",
          "hotkey": "string"
        },
        "files": [
          {
            "id": "string",
            "originalName": "string",
            "filepath": "string",
            "filesize": "string",
            "mimeType": "string",
            "fileType": "string"
          }
        ],
        "voteCount": "number",
        "createdAt": "datetime"
      }
    ]
  }
}
```


PUT /api/bounties/[id]

Updates a bounty (requires authentication and ownership).

Request Body: Same as POST /api/bounties

Submission Routes

GET /api/bounties/[id]/submissions

Retrieves all submissions for a specific bounty.

Query Parameters:
- `status` (optional): Filter by submission status

- `sort` (optional): Sort field (createdAt, score, etc.)
- `order` (optional): Sort order (asc, desc)

POST /api/bounties/[id]/submissions

Creates a new submission for a bounty (requires authentication).

Request Body:

```
{
  "title": "string (required)",
  "description": "string (required)"
}
```

GET /api/submissions/[id]

Retrieves detailed information about a specific submission.

PUT /api/submissions/[id]

Updates a submission (requires authentication and ownership).

POST /api/submissions/[id]/vote

Vote on a submission (requires authentication).

Request Body:

```
{
  "type": "UPVOTE | DOWNVOTE"
}
```

File Management Routes

POST /api/upload

Handles file uploads for submissions (requires authentication).

Request: Multipart form data
- `files` : File array
- `submissionId` : Target submission ID

Response:

```
{
  "message": "X files uploaded successfully",
  "files": [
    {
      "id": "string",
      "originalName": "string",
      "filename": "string",
      "filepath": "string",
      "filesize": "string",
      "mimeType": "string",
      "fileType": "string",
      "uploadedAt": "datetime"
    }
  ]
}
```

GET /api/files/[...filename]

Serves uploaded files with proper content-type headers and security validation.

User Dashboard Route

GET /api/dashboard

Retrieves user dashboard data including user's bounties, submissions, and recent activity (requires authentication).

Response:

```
{
  "user": {
    "id": "string",
    "hotkey": "string",
    "username": "string"
  },
  "stats": {
    "totalBounties": "number",
    "totalSubmissions": "number"
  },
  "bounties": [], // User's created bounties
  "submissions": [], // User's submissions
  "recentActivity": [] // Recent activity on user's bounties
}
```

Error Responses

All API routes return consistent error responses:

```
{
  "error": "Error message description"
}
```

Common HTTP Status Codes:
- 200: Success
- 201: Created successfully

- 400: Bad request/validation error
- 401: Authentication required
- 403: Forbidden/not authorized
- 404: Resource not found
- 409: Conflict (e.g., user already exists)
- 500: Internal server error

Authentication

Most routes require authentication via NextAuth.js session. The session includes:
- User ID
- Crypto wallet address (hotkey)
- Username
- Active status

Authenticated routes check for valid session and appropriate permissions before processing requests.

Database Schema

The application uses Prisma with PostgreSQL and includes the following main models:
- User (with crypto wallet authentication)
- Bounty (with reward system and categories)
- Submission (with validation and voting)
- SubmissionFile (file upload handling)
- Vote (community voting system)
- ValidationLog (submission validation tracking)

File Upload System

- Files are stored in `/uploads` directory
- Supports unlimited file types and sizes
- Automatic file type detection
- Security validation for file paths
- Files served via `/api/files/` endpoint

Future Implementation Notes

These API routes provide the foundation for a full bounty hunting platform. Future enhancements could include:
- Webhook integration for payment processing
- Advanced validation algorithms
- Real-time notifications
- Enhanced search and filtering
- Analytics and reporting endpoints