🚀 MVP Task Platform API

Comprehensive REST API Documentation

MVP Task Collaboration Platform - API Documentation

🚀 Overview

This document provides comprehensive documentation for the MVP Task Collaboration Platform REST API. The API enables task management, user collaboration, and administrative functions.

Base URL: `http://localhost:5002`
API Version: 1.0.0
Authentication: JWT Bearer Token

📋 Table of Contents

🔐 Authentication

All protected endpoints require a JWT token in the Authorization header:

```
Authorization: Bearer
```

Register User

```
POST

/api/auth/register
```

Register a new user account.

Request Body:

```
{
  "name": "John Doe",
  "email": "john@example.com",
  "password": "password123"
}
```

Response (201):

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": 1,
    "name": "John Doe",
    "email": "john@example.com",
    "role": "user"
  }
}
```

Login User

```
POST

/api/auth/login
```

Authenticate user and receive JWT token.

Request Body:

```
{
  "email": "john@example.com",
  "password": "password123"
}
```

Response (200):

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": 1,
    "name": "John Doe",
    "email": "john@example.com",
    "role": "user"
  }
}
```

Get Current User

```
GET

/api/auth/me
```

Get current authenticated user profile.

Headers: `Authorization: Bearer`

Response (200):

```
{
  "user": {
    "id": 1,
    "name": "John Doe",
    "email": "john@example.com",
    "role": "user"
  }
}
```

Forgot Password

POST

/api/auth/forgot-password

Request password reset email.

Request Body:

```
{
  "email": "john@example.com"
}
```

Response (200):

```
{
  "message": "Email sent"
}
```

Reset Password

```
PUT

/api/auth/reset-password/:token
```

Reset password using token from email.

Parameters:

- `token` (path) - Password reset token

Request Body:

```
{
  "password": "newpassword123"
}
```

Response (200):

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": 1,
    "name": "John Doe",
    "email": "john@example.com"
  }
}
```

📋 Tasks Management

Get Users for Assignment

```
GET

/api/tasks/users
```

Get list of all users for task assignment dropdown.

Headers: `Authorization: Bearer`

Response (200):

```
[
  {
    "id": 1,
    "name": "John Doe",
    "email": "john@example.com"
  },
  {
    "id": 2,
    "name": "Jane Smith",
    "email": "jane@example.com"
  }
]
```

Create Task

```
POST

/api/tasks
```

Create a new task.

Headers: `Authorization: Bearer`

Request Body:

```
{
  "title": "Complete project documentation",
  "description": "Write comprehensive API documentation",
  "priority": "High",
  "assignedTo": 2
}
```

Response (201):

```
{
  "id": 1,
  "title": "Complete project documentation",
  "description": "Write comprehensive API documentation",
  "status": "Todo",
  "priority": "High",
  "createdBy": 1,
  "assignedTo": 2,
  "creatorName": "John Doe",
  "assigneeName": "Jane Smith",
  "createdAt": "2024-01-15T10:30:00Z",
  "updatedAt": "2024-01-15T10:30:00Z"
}
```


Get My Tasks

```
GET

/api/tasks/my
```

Get current user's tasks with pagination.

Headers: `Authorization: Bearer`

Query Parameters:

- `page` (optional) - Page number (default: 1)
- `limit` (optional) - Items per page (default: 10)

Response (200):

```
{
  "tasks": [
    {
      "id": 1,
      "title": "Complete project documentation",
      "description": "Write comprehensive API documentation",
      "status": "Todo",
      "priority": "High",
      "createdBy": 1,
      "assignedTo": 2,
      "creatorName": "John Doe",
      "assigneeName": "Jane Smith",
      "createdAt": "2024-01-15T10:30:00Z",
      "updatedAt": "2024-01-15T10:30:00Z"
    }
  ],
  "page": 1,
  "totalPages": 5,
  "total": 42
}
```


Get Task by ID

```
GET

/api/tasks/:id
```

Get specific task details.

Headers: `Authorization: Bearer`

Parameters:

- `id` (path) - Task ID

Response (200):

```
{
  "id": 1,
  "title": "Complete project documentation",
  "description": "Write comprehensive API documentation",
  "status": "Todo",
  "priority": "High",
  "createdBy": 1,
  "assignedTo": 2,
  "creatorName": "John Doe",
  "assigneeName": "Jane Smith",
  "createdAt": "2024-01-15T10:30:00Z",
  "updatedAt": "2024-01-15T10:30:00Z"
}
```


Update Task

```
PUT

/api/tasks/:id
```

Update task details. Only task creator or assignee can update.

Headers: `Authorization: Bearer`

Parameters:

- `id` (path) - Task ID

Request Body:


```
{
  "title": "Updated task title",
  "description": "Updated description",
  "priority": "Medium",
  "status": "In Progress",
  "assignedTo": 3
}
```


Response (200):

```
{
  "message": "Task updated successfully"
}
```

Delete Task

```
DELETE

/api/tasks/:id
```

Delete task. Only task creator can delete.
Headers: `Authorization: Bearer`
Parameters:

- `id` (path) - Task ID

Response (200):

```
{
  "message": "Task deleted successfully"
}
```

💬 Comments System

Get Task Comments

```
GET

/api/comments/task/:taskId
```

Get all comments for a specific task.

Headers: `Authorization: Bearer`

Parameters:

- `taskId` (path) - Task ID

Response (200):

```
[
  {
    "id": 1,
    "taskId": 1,
    "userId": 1,
    "content": "This task is progressing well",
    "userName": "John Doe",
    "createdAt": "2024-01-15T11:00:00Z",
    "updatedAt": "2024-01-15T11:00:00Z"
  }
]
```

Create Comment

```
POST

/api/comments
```

Add a new comment to a task.

Headers: `Authorization: Bearer`

Request Body:

```
{
  "taskId": 1,
  "content": "This task is progressing well"
}
```

Response (201):

```
{
  "id": 1,
  "taskId": 1,
  "userId": 1,
  "content": "This task is progressing well",
  "userName": "John Doe",
  "createdAt": "2024-01-15T11:00:00Z",
  "updatedAt": "2024-01-15T11:00:00Z"
}
```


📈 Activities Feed

Get Recent Activities

```
GET

/api/activities
```

Get recent platform activities (public endpoint).

Response (200):

```
[
  {
    "id": 1,
    "type": "task_created",
    "description": "John Doe created task \"Complete documentation\"",
    "userId": 1,
    "taskId": 1,
    "userName": "John Doe",
    "createdAt": "2024-01-15T10:30:00Z"
  },
  {
    "id": 2,
    "type": "comment_added",
    "description": "Jane Smith commented on task \"Complete documentation\"",
    "userId": 2,
    "taskId": 1,
    "userName": "Jane Smith",
    "createdAt": "2024-01-15T11:00:00Z"
  }
]
```


👑 Admin Operations

Note: All admin endpoints require admin role authentication.

Get All Tasks (Admin)

```
GET

/api/admin/tasks
```

Get all tasks in the system with pagination.
Headers: `Authorization: Bearer`
Query Parameters:

- `page` (optional) - Page number (default: 1)
- `limit` (optional) - Items per page (default: 10)

Response (200):


```
{
  "tasks": [
    {
      "id": 1,
      "title": "Complete project documentation",
      "description": "Write comprehensive API documentation",
      "status": "Todo",
      "priority": "High",
      "createdBy": 1,
      "assignedTo": 2,
      "creatorName": "John Doe",
      "assigneeName": "Jane Smith",
      "createdAt": "2024-01-15T10:30:00Z",
      "updatedAt": "2024-01-15T10:30:00Z"
    }
  ],
  "page": 1,
  "totalPages": 10,
  "total": 95
}
```


Get All Users (Admin)

```
GET

/api/admin/users
```

Get all users in the system.
Headers: `Authorization: Bearer`
Response (200):

```
[
  {
    "id": 1,
    "name": "John Doe",
    "email": "john@example.com",
    "role": "admin",
    "createdAt": "2024-01-10T09:00:00Z"
  },
  {
    "id": 2,
    "name": "Jane Smith",
    "email": "jane@example.com",
    "role": "user",
    "createdAt": "2024-01-12T14:30:00Z"
  }
]
```


Get All Comments (Admin)

```
GET

/api/admin/comments
```

Get all comments in the system.
Headers: `Authorization: Bearer`
Response (200):

```
[
  {
    "id": 1,
    "taskId": 1,
    "userId": 1,
    "content": "This task is progressing well",
    "userName": "John Doe",
    "taskTitle": "Complete project documentation",
    "createdAt": "2024-01-15T11:00:00Z",
    "updatedAt": "2024-01-15T11:00:00Z"
  }
]
```


Update User Role (Admin)

```
PUT

/api/admin/users/:id/role
```

Update user role (user/admin).
Headers: `Authorization: Bearer`
Parameters:

- `id` (path) - User ID

Request Body:


```
{
  "role": "admin"
}
```


Response (200):


```
{
  "message": "User role updated successfully"
}
```

Get Platform Statistics (Admin)

```
GET

/api/admin/stats
```

Get platform statistics and metrics.
Headers: `Authorization: Bearer`
Response (200):

```
{
  "totalUsers": 25,
  "totalTasks": 150,
  "totalComments": 300
}
```

⚠️ Error Handling

Error Response Format

All API errors follow a consistent format:

```
{
  "message": "Error description",
  "errors": [
    {
      "field": "email",
      "message": "Please provide a valid email"
    }
  ]
}
```

HTTP Status Codes

- **200** - Success
- **201** - Created
- **400** - Bad Request (validation errors)
- **401** - Unauthorized (invalid/missing token)
- **403** - Forbidden (insufficient permissions)
- **404** - Not Found
- **500** - Internal Server Error

Common Error Scenarios

Validation Error (400):

```
{
  "errors": [
    {
      "field": "title",
      "message": "Title is required"
    },
    {
      "field": "email",
      "message": "Please provide a valid email"
    }
  ]
}
```

Authentication Error (401):

```
{
  "message": "Invalid credentials"
}
```

Authorization Error (403):

```
{
  "message": "Not authorized to update this task"
}
```

Not Found Error (404):

```
{
  "message": "Task not found"
}
```

🔄 Rate Limiting

Currently, no rate limiting is implemented. In production, consider implementing:

- **Authentication endpoints:** 5 requests per minute
- **General API endpoints:** 100 requests per minute
- **Admin endpoints:** 200 requests per minute

🔧 Development Notes

- **Database:** SQLite (auto-created on first run)
- **Authentication:** JWT with 7-day expiration
- **Password Hashing:** bcryptjs with salt rounds
- **Validation:** express-validator for input validation
- **CORS:** Enabled for localhost:3000 and localhost:3001

Last Updated: December 2025
API Version: 1.0.0