TaskManage - User Guide & API Documentation

Table of Contents

Overview

TaskManage is a comprehensive task management application that helps users organize, track, and complete their tasks efficiently. The application provides both a user-friendly interface and a robust API for developers to integrate task management functionality into their applications.

Key Features

- Create, read, update, and delete tasks
- Task categorization and priority management
- User authentication and authorization
- Task assignment and collaboration
- Progress tracking and reporting
- RESTful API with JSON responses

Getting Started

Prerequisites

- Node.js (v14.0 or higher)
- npm or yarn package manager
- Database (MongoDB)

Installation

1. **Clone the repository:**
2. `git clone https://github.com/gontsejnr/taskManage.git`

3. `cd taskManage`
4. **Install dependencies:**
5. `npm install`
6. **Environment Setup:** Create a `.env` file in the root directory:
7. `PORT=3000`
8. `DATABASE_URL=your_database_connection_string`
9. `NODE_ENV=development`
10. **Start the application:**
11. `npm start`

The application will be available at `http://localhost:3000`

User Guide

Dashboard Overview

The dashboard provides a comprehensive view of all your tasks, organized by status and priority.

Task Status Types

- **Pending**: Tasks that haven't been started
- **In Progress**: Tasks currently being worked on
- **Completed**: Finished tasks
- **Cancelled**: Tasks that were cancelled

Priority Levels

- **High**: Urgent tasks requiring immediate attention
- **Medium**: Important tasks with moderate urgency
- **Low**: Tasks that can be completed when time permits

Creating Tasks

1. **Navigate to the "New Task" section**
2. **Fill in the required fields:**
    - **Title**: Brief description of the task
    - **Description**: Detailed information about the task
    - **Priority**: Select from High, Medium, or Low
    - **Category**: Organize tasks by category
    - **Due Date**: Set a deadline for completion
    - **Assignee**: Select who will be responsible for the task
3. **Click "Create Task"** to save

Managing Tasks

Editing Tasks

- Click on any task to open the edit modal
- Modify the necessary fields
- Click "Save Changes" to update

Deleting Tasks

- Click the delete icon next to the task
- Confirm the deletion in the popup modal

Task Filters

Use the filter options to view specific task subsets:

- Filter by status
- Filter by priority
- Filter by assignee

API Documentation

Base URL

`https://task-manage-blue.vercel.app/`

Task Management Endpoints

GET /tasks

Retrieve all tasks for the authenticated user.

Query Parameters:

- `status` (optional): Filter by task status
- `priority` (optional): Filter by priority level

Response:

```
{
  "success": true,
  "data": {
    "tasks": [
      {
        "id": "string",
        "title": "string",
        "description": "string",
        "status": "pending|in_progress|completed ",
        "priority": "high|medium|low",
        "category": "string",
        "assigneeId": "string",
        "createdAt": "ISO8601 date",
        "updatedAt": "ISO8601 date",
        "dueDate": "ISO8601 date"
      }
    ],

  }
}
```

GET /tasks/:id

Retrieve a specific task by ID.

Response:

```
{
  "success": true,
  "data": {
    "task": {
      "id": "string",
      "title": "string",
      "description": "string",
      "status": "string",
      "priority": "string",
      "category": "string",
      "createdAt": "ISO8601 date",
      "updatedAt": "ISO8601 date",
      "dueDate": "ISO8601 date"
    }
  }
}
```

POST /tasks

Create a new task.

Request Body:

```
{
  "title": "string",
  "description": "string",
  "priority": "high|medium|low",
  "category": "string",
  "dueDate": "ISO8601 date"
}
```

Response:

```
{
  "success": true,
  "message": "Task created successfully",
  "data": {
    "task": {
      "id": "string",
      "title": "string",
      "description": "string",
      "status": "pending",
      "priority": "string",
      "category": "string",
      "createdAt": "ISO8601 date",
      "updatedAt": "ISO8601 date",
      "dueDate": "ISO8601 date"
    }
  }
}
```

PUT /tasks/:id

Update an existing task.

Request Body:

```
{
  "title": "string",
  "description": "string",
  "status": "pending|in_progress|completed|cancelled",
  "priority": "high|medium|low",
  "category": "string",
  "dueDate": "ISO8601 date"
}
```

Response:

```
{
  "success": true,
  "message": "Task updated successfully",
  "data": {
    "task": {
      "id": "string",
      "title": "string",
      "description": "string",
      "status": "string",
      "priority": "string",
      "category": "string",
      "createdAt": "ISO8601 date",
      "updatedAt": "ISO8601 date",
      "dueDate": "ISO8601 date"
    }
  }
}
```

DELETE /tasks/:id

Delete a task.

Response:

```
{
  "success": true,
  "message": "Task deleted successfully"
}
```

Category Management Endpoints

GET /categories

Get all available task categories.

Response:

```
{
  "success": true,
  "data": {
    "categories": [
      {
        "id": "string",
        "name": "string",
        "description": "string",
        "color": "string"
      }
    ]
  }
}
```

POST /categories

Create a new task category.

Request Body:

```
{
  "name": "string",
  "description": "string",
  "color": "string"
}
```

Response:

```
{
  "success": true,
  "message": "Category created successfully",
  "data": {
    "category": {
      "id": "string",
      "name": "string",
      "description": "string",
      "color": "string"
    }
  }
}
```

Error Handling

The API uses standard HTTP status codes and returns consistent error responses:

Error Response Format

```
{
  "success": false,
  "error": {
    "code": "ERROR_CODE",
    "message": "Human-readable error message",
    "details": "Additional error details (optional)"
  }
}
```

Common HTTP Status Codes

- `200` - Success
- `201` - Created successfully
- `400` - Bad Request (validation errors)
- `401` - Unauthorized (authentication required)
- `403` - Forbidden (insufficient permissions)
- `404` - Not Found
- `409` - Conflict (duplicate resource)
- `429` - Too Many Requests (rate limiting)
- `500` - Internal Server Error

Common Error Codes

- `VALIDATION_ERROR` - Request validation failed
- `AUTHENTICATION_REQUIRED` - Valid authentication token required
- `INSUFFICIENT_PERMISSIONS` - User lacks required permissions
- `RESOURCE_NOT_FOUND` - Requested resource doesn't exist
- `DUPLICATE_RESOURCE` - Resource already exists
- `RATE_LIMIT_EXCEEDED` - Too many requests in time window

Examples

JavaScript/Node.js Examples

Create a New Task

```
const axios = require('axios');

const createTask = async () => {
  try {
    const response = await
axios.post('https://api.taskmanage.com/v1/tasks', {
      title: 'Complete project documentation',
      description: 'Write comprehensive API documentation for the project',
      priority: 'high',
      category: 'documentation',
      dueDate: '2024-02-15T18:00:00Z'
    }, {
      headers: {
        'Authorization': 'Bearer your_jwt_token_here',
        'Content-Type': 'application/json'
      }
    });

    console.log('Task created:', response.data);
  } catch (error) {
    console.error('Error creating task:', error.response.data);
  }
};
```

Fetch All Tasks

```
const fetchTasks = async () => {
  try {
    const response = await
axios.get('https://api.taskmanage.com/v1/tasks?status=pending&page=1&limit=
10', {
      headers: {
        'Authorization': 'Bearer your_jwt_token_here'
      }
    });

    console.log('Tasks:', response.data.data.tasks);
  } catch (error) {
    console.error('Error fetching tasks:', error.response.data);
  }
};
```

Update Task Status

```
const updateTaskStatus = async (taskId, newStatus) => {
  try {
    const response = await
axios.put(`https://api.taskmanage.com/v1/tasks/${taskId}`, {
      status: newStatus
    }, {
      headers: {
        'Authorization': 'Bearer your_jwt_token_here',
        'Content-Type': 'application/json'
      }
    });

    console.log('Task updated:', response.data);
  } catch (error) {
    console.error('Error updating task:', error.response.data);
  }
};

// Usage
updateTaskStatus('task_id_here', 'completed');
```

Create Task

```
curl -X POST https://api.taskmanage.com/v1/tasks \
  -H "Content-Type: application/json" \
  -d '{
    "title": "New Task",
    "description": "Task description",
    "priority": "medium",
    "category": "development"
  }'
```

Get Tasks

```
curl -X GET
"https://api.taskmanage.com/v1/tasks?status=pending&page=1&limit=5" \
```

Update Task

```
curl -X PUT https://api.taskmanage.com/v1/tasks/task_id_here \
  -H "Content-Type: application/json" \
  -d '{
    "status": "completed"
  }'
```

Delete Task

```
curl -X DELETE https://api.taskmanage.com/v1/tasks/task_id_here \
```

Troubleshooting

Common Issues and Solutions

Task Creation Fails

Problem: Tasks are not being created **Solutions**:

- Verify all required fields are provided
- Check that field values match the expected format
- Ensure the due date is in ISO8601 format

Performance Issues

Problem: API responses are slow **Solutions**:

- Use pagination for large result sets
- Implement client-side caching for frequently accessed data
- Use appropriate filters to reduce response size
- Check network connectivity

Database Connection Errors

Problem: Getting 500 Internal Server Error **Solutions**:

- Verify database connection string is correct
- Check that database server is running
- Ensure database user has appropriate permissions

- Review server logs for detailed error messages

Debug Mode

Enable debug mode by setting the environment variable:

```
DEBUG=taskmanage:* npm start
```

Log Levels

The application supports different log levels:

- `error`: Error messages only
- `warn`: Warning and error messages
- `info`: Informational, warning, and error messages
- `debug`: All messages including debug information

Set the log level in your `.env` file:

```
LOG_LEVEL=debug
```

Rate Limiting

The API implements rate limiting to prevent abuse:

- **Authentication endpoints**: 5 requests per minute
- **Task operations**: 100 requests per minute
- **General endpoints**: 60 requests per minute

If you exceed these limits, you'll receive a 429 status code. Wait for the time window to reset before making additional requests.

Support and Contributing

For additional support or to contribute to the project:

- **Issues**: Report bugs or request features on GitHub
- **Documentation**: Help improve this documentation
- **Code**: Submit pull requests for bug fixes or new features

Visit the GitHub repository for more information.
