Task Manager REST API

Developer:

Vipulsinh Parmar
Academic Program: B.Tech – Artificial Intelligence and Data Science

Objective:

The primary goal of this project is to build a full-stack Task Management System that allows users to create, read, update, and delete tasks, along with tracking progress via statistics. The backend exposes a RESTful API built with Node.js and Express.js, and the frontend provides a dynamic and styled interface using React.js, consuming the API to offer a seamless user experience.

Key Goals:

- Keep track of multiple tasks
- Monitor task progress using status tracking
- Visually manage task lifecycles with simple interactions
- Implement modern web development practices
- Create a scalable and maintainable codebase

Technology Stack:

Backend - Node.js, Express.js (REST API server development)
Frontend - React.js (User interface and state management)
Styling - CSS3 (Modern UI design and responsive layout)
Testing - Postman (API endpoint testing and validation)
Middleware - CORS, Body-parser (Cross-origin requests and JSON parsing)

Core Functionalities:

Task Management Features:

1. Add New Task - Create tasks with title, description, and status
2. View All Tasks - Display complete task list with filtering
3. Update Task Status - Change task progress (pending → in-progress → completed)
4. Edit Task Content - Modify task title and description
5. Delete Task - Remove tasks from the system
6. Task Statistics Panel - Real-time overview (Total, Pending, In Progress, Completed)

Task Properties:
javascript
```
{
  id: "unique_identifier",           // Auto-generated unique ID
  title: "Task Title",               // Required field
  description: "Task Description",    // Optional field
  status: "pending",                 // pending | in-progress | completed
  createdAt: "2024-01-15T10:30:00Z", // Timestamp
  updatedAt: "2024-01-15T14:45:00Z"  // Last modified timestamp
}
```

API Documentation:

Base URL: `http://localhost:3000/api`

Endpoints Overview:

Method	Endpoint	Description	Request Body
GET	/tasks	Get all tasks	None
GET	/tasks/:id	Get task by ID	None
POST	/tasks	Create new task	{title, description, status}
PUT	/tasks/:id	Update entire task	{title, description, status}
PATCH	/tasks/:id/status	Update status only	{status}
DELETE	/tasks/:id	Delete a task	None
GET	/stats	Get task statistics	None

Detailed API Specifications:

1. Create New Task
http
```
POST /api/tasks
Content-Type: application/json
{
  "title": "Learn React.js",
  "description": "Complete React fundamentals course",
  "status": "pending"
}
```

Response (201 Created):

```
{
  "id": "1",
  "title": "Learn React.js",
  "description": "Complete React fundamentals course",
  "status": "pending",
  "createdAt": "2024-01-15T10:30:00Z",
  "updatedAt": "2024-01-15T10:30:00Z"
}
```


2. Get All Tasks

http
GET /api/tasks
Query Parameters: ?status=pending (optional filtering)

Response (200 OK):


```
[
  {
    "id": "1",
    "title": "Learn React.js",
    "description": "Complete React fundamentals course",
    "status": "pending",
    "createdAt": "2024-01-15T10:30:00Z",
    "updatedAt": "2024-01-15T10:30:00Z"
  }
]
```


3. Update Task Status

http
PATCH /api/tasks/1/status
Content-Type: application/json


```
{
  "status": "completed"
}
```


Response (200 OK):


```
{
  "id": "1",
  "title": "Learn React.js",
  "description": "Complete React fundamentals course",
  "status": "completed",
  "createdAt": "2024-01-15T10:30:00Z",
  "updatedAt": "2024-01-15T14:45:00Z"
```

}

4. Get Task Statistics
http
GET /api/stats

Response (200 OK):
{
  "total": 10,
  "pending": 4,
  "in-progress": 3,
  "completed": 3
}

Frontend Implementation:

React Components Architecture:

Main App Component (App.js):
- State Management: Uses React hooks (`useState`, `useEffect`)
- API Integration: Handles all HTTP requests to backend
- Error Handling: Displays user-friendly error messages
- Loading States: Shows loading spinners during API calls

Component Breakdown:
1. TaskForm.js - Handles task creation with form validation
2. TaskList.js - Renders task collection with filtering options
3. TaskItem.js - Individual task display with action buttons

UI/UX Design:

Design Principles:
- Minimalistic Design: Clean, distraction-free interface
- Responsive Layout: Works seamlessly on desktop and mobile
- Visual Feedback: Color-coded status indicators
- Intuitive Navigation: Clear action buttons and form controls

Testing Strategy:

API Testing with Postman:

Test Cases Executed:

Create valid task
POST
/api/tasks
201 Created with task data
✅ Pass

Create task without title
POST
/api/tasks
400 Bad Request
✅ Pass

Get all tasks
GET
/api/tasks
200 OK with task array
✅ Pass

Get non-existent task
GET
/api/tasks/999
404 Not Found
✅ Pass

Update task status
PATCH
/api/tasks/1/status
200 OK with updated task
✅ Pass

Update with invalid status
PATCH
/api/tasks/1/status
400 Bad Request
✅ Pass

Delete existing task
DELETE
/api/tasks/1
200 OK with confirmation
🟩 Pass

Delete non-existent task
DELETE
/api/tasks/999
404 Not Found
🟩 Pass

Get task statistics
GET
/api/stats
200 OK with statistics object
🟩 Pass

Frontend Testing:
- Manual Testing:All UI interactions tested across different browsers
- Responsive Testing: Verified layout on various screen sizes
- Error Handling: Tested offline scenarios and API failures

Performance Optimizations:

Backend Optimizations:
- Efficient Routing: Modular route organization
- Memory Management: In-memory storage with optimized data structures
- Error Handling: Comprehensive error responses with appropriate HTTP status codes

Frontend Optimizations:
- React Hooks: Efficient state management with `useState` and `useEffect`
- API Caching: Prevents unnecessary re-fetching of data
- Conditional Rendering: Optimized component rendering based on state
- CSS Animations: Smooth transitions for better user experience

Deployment Instructions:
Development Environment Setup:

Prerequisites:
- Node.js (v14.0.0 or higher)
- npm (v6.0.0 or higher)
- Git

Backend Setup:
bash
Navigate to backend directory
cd task-manager-api

Install dependencies
npm install

Start development server
npm run dev
Server runs on http://localhost:3000

Frontend Setup:
bash
Navigate to frontend directory
cd task-manager-frontend

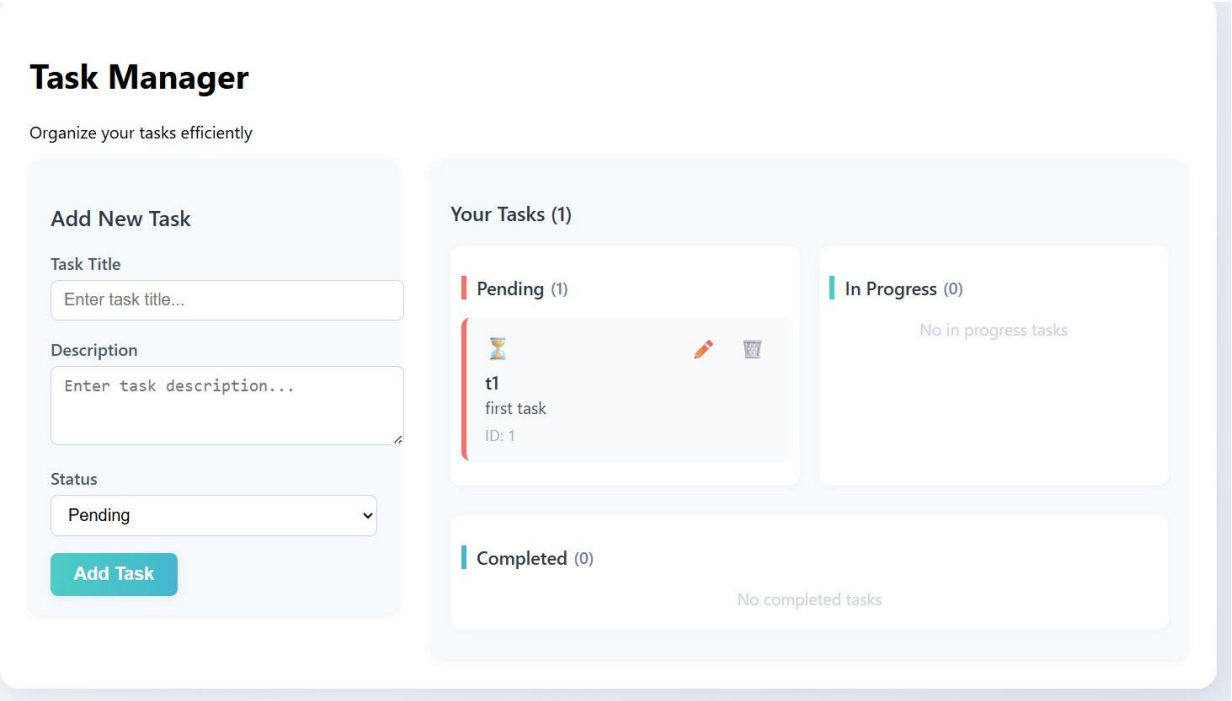
Install dependencies
npm install

Start React development server
npm start
Application runs on http://localhost:3001

Production Deployment:
bash
Backend Production Build
npm start

Frontend Production Build
npm run build

Screenshot:



Conclusion:

The Task Manager REST API project successfully demonstrates a complete full-stack web application development lifecycle. The project showcases modern web development practices, including RESTful API design, React-based frontend development, and responsive user interface design.

Key Achievements:
Functional REST API: Complete CRUD operations with proper HTTP methods
Modern React Frontend: Interactive UI with real-time updates
Responsive Design: Works seamlessly across devices
Comprehensive Testing: Thorough API testing with Postman
Clean Architecture: Modular, maintainable codebase
Error Handling: Robust error management and user feedback

This project serves as a solid foundation for more complex applications and demonstrates proficiency in modern web development technologies. The modular architecture and clean code practices make it highly extensible for future enhancements.

GitHub Repository: [https://github.com/VIPshiv/selkey]