SOURCE CODE:

BACKEND:

Task.js:

```
const mongoose = require('mongoose');
  const TaskSchema = new mongoose.Schema({
    title: { type: String, required: true },
    description: { type: String },
    priority: { type: String, enum: ['Low', 'Medium', 'High'], default: 'Medium' },
    category: { type: String, enum: ['Work', 'Personal', 'Urgent'], default: 'Work' },
    completed: { type: Boolean, default: false },
    createdAt: { type: Date, default: Date.now },
  });
  module.exports = mongoose.model('Task', TaskSchema);
```

taskRoutes.js:

```
const express = require('express');
  const Task = require('../models/Task');
  const router = express.Router();
  router.post('/', async (req, res) => {
    try {
      const task = new Task(req.body);
      await task.save();
      res.status(201).json(task);
    } catch (error) {
      res.status(500).json({ error: error.message });
    }
  });
  router.get('/', async (req, res) => {
    try {
```

```
    const tasks = await Task.find();
    res.status(200).json(tasks);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
router.put('/:id', async (req, res) => {
  try {
    const task = await Task.findByIdAndUpdate(req.params.id, req.body, { new: true });
    res.status(200).json(task);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
router.delete('/:id', async (req, res) => {
  try {
    await Task.findByIdAndDelete(req.params.id);
    res.status(200).send('Task deleted');
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
module.exports = router;
```

index.js:

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const dotenv = require('dotenv');
```

```
dotenv.config();
const app = express();
const PORT = process.env.PORT || 5000;
app.use(cors());
app.use(express.json());
mongoose
  .connect(process.env.MONGO_URI, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => app.listen(PORT, () => console.log(`Server running on port ${PORT}`)))
  .catch((error) => console.log(error.message));
const taskRoutes = require('./routes/taskRoutes');
app.use('/tasks', taskRoutes);
```

.env:

```
MONGO_URI=your-mongodb-connection-string
```

FRONTEND:

TaskForm.js:

```
import React, { useState } from 'react';
import axios from 'axios';
const TaskForm = ({ fetchTasks }) => {
  const [title, setTitle] = useState('');
  const [description, setDescription] = useState('');
  const [priority, setPriority] = useState('Medium');
  const [category, setCategory] = useState('Work');
  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.post('http://localhost:5000/tasks', {
        title,
```

```
      description,
      priority,
      category,
    });
    fetchTasks();
    setTitle('');
    setDescription('');
    setPriority('Medium');
    setCategory('Work');
  } catch (error) {
    console.error('Error adding task:', error);
  }
};
return (
  <form className="task-form" onSubmit={handleSubmit}>
    <input
      type="text"
      placeholder="Task Title"
      value={title}
      onChange={(e) => setTitle(e.target.value)}
      required
    />
    <textarea
      placeholder="Description"
      value={description}
      onChange={(e) => setDescription(e.target.value)}
    />
    <select value={priority} onChange={(e) => setPriority(e.target.value)}>
```

```
          <option value="Low">Low</option>
          <option value="Medium">Medium</option>
          <option value="High">High</option>
        </select>
        <select value={category} onChange={(e) => setCategory(e.target.value)}>
          <option value="Work">Work</option>
          <option value="Personal">Personal</option>
          <option value="Urgent">Urgent</option>
        </select>
        <button type="submit">Add Task</button>
      </form>
    );
  };
  export default TaskForm;
```

TaskItem.js:

```
import React from 'react';
  import axios from 'axios';
  const TaskItem = ({ task, fetchTasks }) => {
    const handleDelete = async () => {
      try {
        await axios.delete(`http://localhost:5000/tasks/${task._id}`);
        fetchTasks();
      } catch (error) {
        console.error('Error deleting task:', error);
      }
    };
    const toggleComplete = async () => {
      try {
```

```
      await axios.put(`http://localhost:5000/tasks/${task._id}`, {
        ...task,
        completed: !task.completed,
      });
      fetchTasks();
    } catch (error) {
      console.error('Error updating task:', error);
    }
  };
  return (
    <div className={`task-item ${task.completed ? 'completed' : ''}`}>
      <h3>{task.title}</h3>
      <p>{task.description}</p>
      <p>Priority: {task.priority}</p>
      <p>Category: {task.category}</p>
      <button onClick={toggleComplete}>
        {task.completed ? 'Mark Incomplete' : 'Mark Complete'}
      </button>
      <button onClick={handleDelete}>Delete</button>
    </div>
  );
};
export default TaskItem;
```

TaskList.js:

```
import React from 'react';
  import TaskItem from './TaskItem';
  const TaskList = ({ tasks, fetchTasks }) => {
    return (
```

```
      <div className="task-list">
        {tasks.map((task) => (
          <TaskItem key={task._id} task={task} fetchTasks={fetchTasks} />
        ))}
      </div>
    );
  };
  export default TaskList;
```

App.js:

```
import React, { useState, useEffect } from 'react';
  import axios from 'axios';
  import TaskList from './components/TaskList';
  import TaskForm from './components/TaskForm';
  import './App.css';
  const App = () => {
    const [tasks, setTasks] = useState([]);
    const fetchTasks = async () => {
      try {
        const response = await axios.get('http://localhost:5000/tasks');
        setTasks(response.data);
      } catch (error) {
        console.error('Error fetching tasks:', error);
      }
    };
    useEffect(() => {
      fetchTasks();
    }, []);
    return (
```

```
      <div className="app">
        <h1>Task Manager</h1>
        <TaskForm fetchTasks={fetchTasks} />
        <TaskList tasks={tasks} fetchTasks={fetchTasks} />
      </div>
    );
  };
  export default App;
```

App.css:

```
.App {
  text-align: center;
}
.App-logo {
  height: 40vmin;
  pointer-events: none;
}
@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}
.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
```

```
  font-size: calc(10px + 2vmin);
  color: white;
}
.App-link {
  color: #61dafb;
}
@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

OUTPUT: