Code:

1)Models

```
import mongoose from 'mongoose';

const taskSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
    trim: true,
    maxlength: 100
  },
  description: {
    type: String,
    trim: true,
    default: ''
  },
  completed: {
    type: Boolean,
    default: false
  },
  createdAt: {
    type: Date,
    default: Date.now
  },
```

Fig 1.1

```
  // Update completedAt when task is marked as completed
  taskSchema.pre('save', function(next) {
    if (this.isModified('completed')) {
      if (this.completed && !this.completedAt) {
        this.completedAt = new Date();
      } else if (!this.completed) {
        this.completedAt = null;
      }
    }
    next();
  });

  export default mongoose.model('Task', taskSchema);
```

Fig 1.2

2)Routes

```
import express from 'express';
import Task from '../models/Task.js';

const router = express.Router();

// Get all tasks
router.get('/', async (req, res) => {
  try {
    const tasks = await Task.find().sort({ createdAt: -1 });
    res.json(tasks);
  } catch (error) {
    console.error('Error fetching tasks:', error);
    res.status(500).json({ error: 'Failed to fetch tasks' });
  }
});

// Create a new task
router.post('/', async (req, res) => {
  try {
    const { title, description } = req.body;

    if (!title || title.trim().length === 0) {
      return res.status(400).json({ error: 'Task title is required' });
    }
```

Fig 2.1

```
// Update a task (toggle completion or edit)
router.put('/:id', async (req, res) => {
  try {
    const { id } = req.params;
    const updates = req.body;

    const task = await Task.findByIdAndUpdate(
      id,
      updates,
      { new: true, runValidators: true }
    );

    if (!task) {
      return res.status(404).json({ error: 'Task not found' });
    }

    res.json(task);
  } catch (error) {
    console.error('Error updating task:', error);
    res.status(500).json({ error: 'Failed to update task' });
  }
});
```

Fig 2.2

```
// Delete a task
router.delete('/:id', async (req, res) => {
  try {
    const { id } = req.params;
    const task = await Task.findByIdAndDelete(id);

    if (!task) {
      return res.status(404).json({ error: 'Task not found' });
    }

    res.json({ message: 'Task deleted successfully' });
  } catch (error) {
    console.error('Error deleting task:', error);
    res.status(500).json({ error: 'Failed to delete task' });
  }
});
```

Fig 2.3

```
// Get task statistics
router.get('/stats', async (req, res) => {
  try {
    const total = await Task.countDocuments();
    const completed = await Task.countDocuments({ completed: true });
    const pending = total - completed;
    const completionRate = total > 0 ? (completed / total) * 100 : 0;

    res.json({
      total,
      completed,
      pending,
      completionRate
    });
  } catch (error) {
    console.error('Error fetching stats:', error);
    res.status(500).json({ error: 'Failed to fetch statistics' });
  }
});

export default router;
```

Fig 2.4

3)index.js(entry point)

```
import express from 'express';
import mongoose from 'mongoose';
import cors from 'cors';
import dotenv from 'dotenv';
import taskRoutes from './routes/tasks.js';

dotenv.config();

const app = express();
const PORT = process.env.PORT || 3001;

// Middleware
app.use(cors());
app.use(express.json());

// Connect to MongoDB
mongoose.connect(process.env.MONGODB_URI || 'mongodb://localhost:27017/react-hooks-demo')
  .then(() => {
    console.log('Connected to MongoDB');
  })
  .catch((error) => {
    console.error('MongoDB connection error:', error);
    process.exit(1);
  });
```

Fig 3.1

```
// Routes
app.use('/api/tasks', taskRoutes);

// Health check endpoint
app.get('/api/health', (req, res) => {
  res.json({ status: 'OK', timestamp: new Date().toISOString() });
});

// Error handling middleware
app.use((error, req, res, next) => {
  console.error('Server error:', error);
  res.status(500).json({ error: 'Internal server error' });
});

app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

Fig 3.2

Output: