# TaskFlow: Task Management System Report

Uday Sharma
uday.sharma@gmail.com

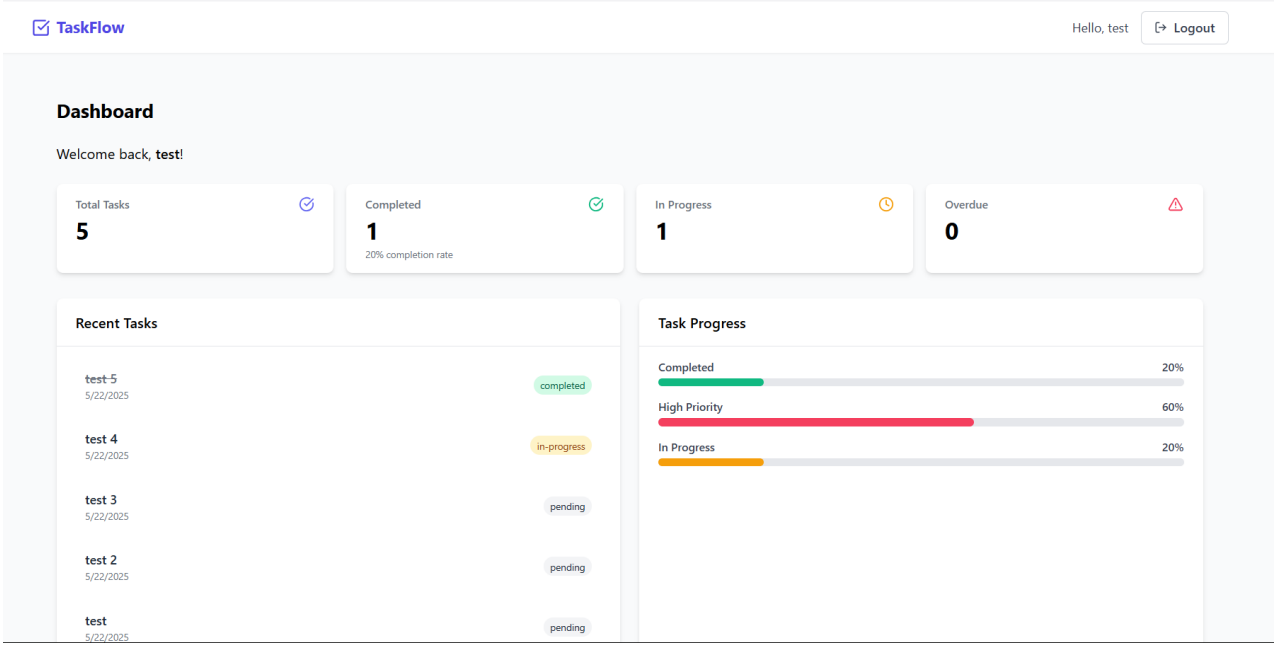
May 22, 2025

Version 1.0

# Contents

# 1 Project Overview

TaskFlow is a secure, scalable task management system built using the MERN stack (MongoDB, Express.js, React, Node.js). Designed for individual users, it of- fers robust JWT-based authentication, real-time task tracking, and an intuitive UI/UX. The system leverages modern development practices, including TypeScript for type safety, RESTful APIs for communication, and cloud-native deployment on Render.com with MongoDB Atlas for data storage. TaskFlow ensures efficient task management with features like filtering, progress tracking, and secure data handling.




# 2 System Architecture

TaskFlow follows a client-server architecture with distinct frontend and backend components communicating via RESTful APIs.

## 2.1 Technology Stack

| Component | Technology |
|---|---|
| Frontend | React 18.3.1, TypeScript, Tailwind CSS, React Router v6, Axios, Lucide Reac |
| Backend | Node.js 18, Express.js 4.18.2, MongoDB with Mongoose ORM Authentication JWT (jsonwebtoken), Bcryptjs, HTTP-only Cookies |
| Deployment | Render.com, MongoDB Atlas |

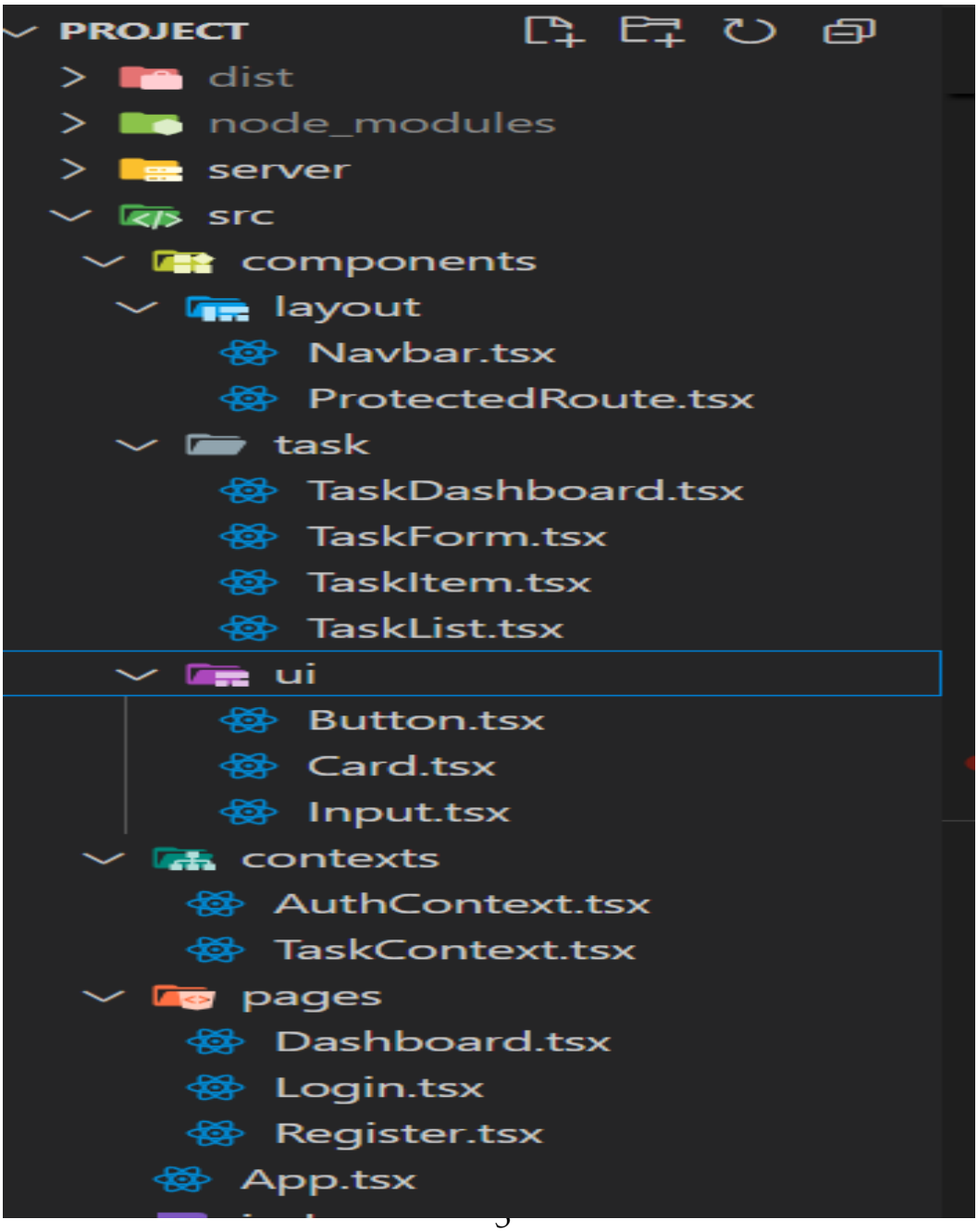
Table 1: Technology Stack

## 2.2 Architectural Diagram

[Frontend] ↔ [REST API] ↔ [MongoDB]
↑ ↑
React Express
Context Middleware

## 2.3 Frontend Components

- **Layout**: Navbar, ProtectedRoute
- **Task** : TaskForm, TaskItem, TaskList, TaskDashboard
- **State Management**: AuthContext (user authentication), TaskContext (task operations)



### 2.4 Backend

- Express.js handles API routing and middleware.
- Mongoose ORM manages MongoDB interactions.
- CORS enabled for secure frontend-backend communication.

## 3 Key Features

- **Authentication & Authorization**:
  - Use of JWT-based authentication with 30-day session persistence.
  - Secure password hashing (bcryptjs).
  - Role-based access control .
- **Task Management**:
  - CRUD operations with real-time updates.
  - Tasks get Filtering by status (pending, in-progress, completed), priority (low, medium, high), and due date.
  - Progress tracking (0–100%).
- **Security**:
  - Input sanitization using Express-validator.
  - Email format and password length validation.

## 4 Database Design

### 4.1 User Model

```
{
  name: { type: String, required: true, trim: true },
  email: { type: String, required: true, unique: true, match: /^[\w
     -\.]+@([\w-]+\.)+[\w-]{2,4}$/ },
  password: { type: String, required: true, minlength: 6 },
  createdAt: { type: Date, default: Date.now }
}
```

### 4.2 Task Model

```
{
  title: { type: String, required: true, trim: true },
  description: { type: String, trim: true },
  status: { type: String, enum: ['pending', 'in-progress', '
     completed'] },
  priority: { type: String, enum: ['low', 'medium', 'high'] },
  dueDate: { type: Date },
  user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  createdAt: { type: Date, default: Date.now }
}
```

# 5 API Endpoints

## 5.1 Authentication Endpoints

| Endpoint | Method | Description |
|---|---|---|
| /api/users/register POST | | Create new user account |
| /api/users/login | POST | Generate JWT token |
| /api/users/profile | GET | Get authenticated user data |

Table 2: Authentication Endpoints

**Request/Response**:

- POST /api/users/register: Request: { name, email, password }, Re- sponse: { _id, name, email, token }
- POST /api/users/login: Request: { email, password }, Response: { _id, name, email, token }
- GET /api/users/profile: Headers: Authorization: Bearer <token>, Re- sponse: { _id, name, email }

## 5.2 Task Endpoints

| Endpoint | Method | Parameters |
|---|---|---|
| /api/tasks | GET | ?status=completed&sort=-dueDate |
| /api/tasks | POST | { title, description?, status?, priority?, dueDa |
| /api/tasks/:id | PUT | { title?, description?, status?, priority?, dueD |

Table 3: Task Endpoints

# 6 Version Control and Git Workflow

TaskFlow uses Git for version control, hosted on GitHub. The repository follows a workflow to ensure organized development and maintainable code.

- **Main Branch**: Stable, production-ready code deployed to Render.com.
- **Commits**: Descriptive messages (e.g., Add task creation endpoint, Fix CORS configuration).
- **Pull Requests**: Used for code reviews before merging into main.

**Git Commands**:

```
# Clone the repository
git clone https://github.com/udaysharma9171/Task-Flow.git

# Create and switch to a feature branch
git checkout -b feature/add-task-filtering

# Commit changes
git commit -m "Implement task filtering by status and priority"

# Push branch to GitHub
git push origin feature/add-task-filtering

# Create pull request via GitHub UI
```

# 7 Deployment

TaskFlow is deployed on Render.com with MongoDB Atlas for database hosting.

## 7.1 Render.com Configuration

**Frontend Service:**

- **Build Command:** npm run build
- **Publish Directory:** dist/
- **Environment Variable:**

```
VITE_API_URL=https://task-flow-backend.onrender.com
```

**Backend Service:**

- **Node Version:** 18.x
- **Environment Variables:**

```
MONGODB_URI=
mongodb+srv://udaysharma9171:udaysharma9171@cluster0.cvzbpda.mongo
db.net/taskmanager?retryWrites=true&w=majority&appName=Cluster0
PORT=5001
```

### 7.2 CORS Policy

```
app.use(cors({
    origin: 'https://task-flow-frontend.onrender.com', methods: ['GET', 'POST', 'PUT',
    'DELETE'],
    allowedHeaders:['Content-Type','Authorization'], credentials: true
}));
```

# 8 Testing

- **Postman Collection**:
  - 42 test cases covering all API endpoints.
  - Environment templates for staging and production.

- **Frontend Tests**:

```
describe('Task Workflow', () => {
    it('Creates, updates, and deletes task', () => {
        cy.login('test@taskflow.com', 'password123');
        cy.createTask('Test Task'); cy.updateTaskStatus('TestTask',
        'in-progress'); cy.deleteTask('Test Task');
    });
```

# 9 Future Roadmap

### 9.1

- Real-time collaboration (Socket.IO)

| Challenge | Solution |
|---|---|
| Render.com IP whitelisting | MongoDB Atlas network access rules |
| JWT token persistence | HTTP-only cookies with Secure flag State |
| synchronization | React Context API + Axios interceptors |

Table 4: Challenges and Solutions

- File attachments
- Team management module

# 10 Maintenance and Support

- **Monitoring**: UptimeRobot and MongoDB Atlas alerts
- **Logging**: Render.com log streaming

- **Updates**: Semantic versioning for API endpoints

# 11 Conclusion

TaskFlow represents a pinnacle of modern full-stack development, delivering a secure, efficient, and user-centric task management system tailored for individual users. Built on the MERN stack with TypeScript, it leverages robust JWT-based authentication, real-time task tracking, and an intuitive UI/UX, ensuring a seamless experience. The project's modular architecture, underpinned by a well-structured Git workflow on GitHub (https://github.com/udaysharma9171/Task-Flow.git), facilitates maintainability and collaborative development. Its cloud-native deployment on Render.com (https://task-flow-frontend.onrender.com/) with MongoDB Atlas ensures scalability and reliability, meeting the demands of modern web applications. Comprehensive testing with Postman and Cypress, along with security measures like input sanitization and CSRF protection, underscores its production-ready quality. Looking ahead, TaskFlow is well-positioned for future enhancements, such as real-time collaboration, mobile app integration, and advanced features like Kanban boards, making it a versatile foundation for both personal and potential team-based task management solutions.

# 12 Links and Resources

- **Hosted Site**: https://task-flow-frontend.onrender.com/
- **Repository**: https://github.com/udaysharma9171/Task-Flow.git

**Prepared by:**
Uday Sharma
**github** : https://github.com/udaysharma9171
**Portfolio :** https://udaysharma9171.github.io/Portfolio/