



TaskFlow: Task & Project Management System

Software Requirements Specification (SRS)

Submitted By:

Abdul Rehman (2023-CS-20)
Hamid Riaz (2023-CS-10)
Ahmed Butt (2023-CS-18)
M. Zaeem (2023-CS-38)

Submitted To:

Sir Amjad Farooq

Department of Computer Science
University of Engineering and Technology, Lahore

January 1, 2026

Contents

1	Introduction	3
1.1	Purpose	3
1.2	Scope	3
1.3	Definitions, Acronyms, and Abbreviations	3
1.4	References	4
1.5	Overview	4
2	Overall Description	4
2.1	Product Perspective	4
2.2	Product Functions	4
2.3	User Classes and Characteristics	5
2.4	Operating Environment	6
2.5	Design and Implementation Constraints	6
2.6	Assumptions and Dependencies	6
3	Specific Requirements	6
3.1	Functional Requirements	6
3.1.1	FR1: User Authentication	6
3.1.2	FR2: Profile Management	7
3.1.3	FR3: Project Management	7
3.1.4	FR4: Task Management	7
3.1.5	FR5: Dashboard	7
3.1.6	FR6: Team Collaboration	8
3.2	Non-Functional Requirements	8
3.2.1	NFR1: Performance	8
3.2.2	NFR2: Security	8
3.2.3	NFR3: Usability	8
3.2.4	NFR4: Reliability	8
3.2.5	NFR5: Maintainability	9
3.2.6	NFR6: Scalability	9
4	System Architecture	9
4.1	Technology Stack	9
4.2	System Components	9
4.2.1	Frontend Application	9
4.2.2	Backend API	10
4.2.3	Database	10
4.3	API Endpoints	10
4.3.1	Authentication Routes	10
4.3.2	Project Routes	10
4.3.3	Task Routes	11
4.3.4	Dashboard Routes	11
5	Data Models	11
5.1	User Schema	11
5.2	Project Schema	12
5.3	Task Schema	12

5.4	Comment Schema	12
6	Deployment	13
6.1	Docker Configuration	13
6.2	Environment Variables	13
6.3	Deployment Commands	13
7	Testing Requirements	13
7.1	Unit Testing	13
7.2	Integration Testing	13
7.3	User Acceptance Testing	14
8	Future Enhancements	14
9	Appendices	14
9.1	Appendix A: Glossary	14
9.2	Appendix B: Use Case Diagrams	14
9.3	Appendix C: Database ERD	14

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a comprehensive description of the TaskFlow system - a modern, full-stack Task and Project Management application built using the MERN (MongoDB, Express.js, React, Node.js) stack. This document is intended for:

- Development team members
- Project stakeholders
- Quality assurance teams
- System administrators
- End users

1.2 Scope

TaskFlow is a web-based collaborative project management system designed to streamline team workflows, task assignments, and project tracking. The system enables:

- Project creation and management
- Task assignment and tracking with Kanban boards
- Team collaboration and member management
- Real-time dashboard analytics
- User authentication and role-based access control
- File uploads and attachments
- Comment and activity tracking

1.3 Definitions, Acronyms, and Abbreviations

- **MERN Stack:** MongoDB, Express.js, React, Node.js
- **SRS:** Software Requirements Specification
- **API:** Application Programming Interface
- **JWT:** JSON Web Token
- **REST:** Representational State Transfer
- **CRUD:** Create, Read, Update, Delete
- **UI:** User Interface
- **UX:** User Experience

1.4 References

- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
- MERN Stack Documentation
- Docker Documentation
- MongoDB Documentation
- React.js Documentation

1.5 Overview

This document is structured into several sections describing functional requirements, non-functional requirements, system architecture, and deployment specifications. Each section provides detailed information about specific aspects of the TaskFlow system.

2 Overall Description

2.1 Product Perspective

TaskFlow is a standalone web application designed to replace traditional project management methods with a modern, digital solution. The system consists of:

- **Frontend:** React-based single-page application with Vite build tool
- **Backend:** Node.js/Express.js RESTful API server
- **Database:** MongoDB for persistent data storage
- **Deployment:** Docker containerization for easy deployment

2.2 Product Functions

The major functions of TaskFlow include:

1. User Management

- User registration and authentication
- Profile management
- Role-based access control
- Password management

2. Project Management

- Create, read, update, delete projects
- Project status tracking
- Priority assignment

- Member management

3. Task Management

- Task creation and assignment
- Status updates via Kanban board
- Priority levels (low, medium, high, critical)
- Subtask management
- Due date tracking

4. Team Collaboration

- Team member viewing
- Task comments and discussions
- Activity tracking
- File attachments

5. Dashboard Analytics

- Project statistics
- Task completion metrics
- Upcoming deadlines
- Recent activity feed

2.3 User Classes and Characteristics

1. Regular User

- Can view assigned tasks
- Can update task status
- Can view projects they're members of
- Can add comments

2. Project Manager

- All user permissions
- Can create projects
- Can assign tasks
- Can manage team members

3. Administrator

- All manager permissions
- Can manage all users
- Can access all projects
- Can view team performance metrics

2.4 Operating Environment

- **Client Side:** Modern web browsers (Chrome, Firefox, Safari, Edge)
- **Server Side:** Node.js v20.x runtime environment
- **Database:** MongoDB 6.x or higher
- **Container Platform:** Docker and Docker Compose
- **Operating System:** Linux, Windows, macOS (via Docker)

2.5 Design and Implementation Constraints

- Must use MERN stack technologies
- Must be containerized using Docker
- Must follow RESTful API design principles
- Must implement JWT-based authentication
- Must be responsive for mobile and desktop devices
- Must support file uploads up to 5MB

2.6 Assumptions and Dependencies

- Users have stable internet connection
- Docker is installed for deployment
- MongoDB database is accessible
- Users have modern web browsers
- Email service is configured for notifications (future enhancement)

3 Specific Requirements

3.1 Functional Requirements

3.1.1 FR1: User Authentication

- **FR1.1:** System shall allow users to register with name, email, password, and department
- **FR1.2:** System shall validate email format and password strength
- **FR1.3:** System shall authenticate users using email and password
- **FR1.4:** System shall generate JWT tokens for authenticated sessions
- **FR1.5:** System shall allow users to log out

3.1.2 FR2: Profile Management

- **FR2.1:** Users shall be able to view their profile information
- **FR2.2:** Users shall be able to update profile details
- **FR2.3:** Users shall be able to change their password
- **FR2.4:** System shall validate password changes

3.1.3 FR3: Project Management

- **FR3.1:** Users shall be able to create new projects
- **FR3.2:** Users shall be able to view all projects they own or are members of
- **FR3.3:** Users shall be able to update project details
- **FR3.4:** Users shall be able to delete projects they own
- **FR3.5:** System shall support project status: planning, in-progress, on-hold, completed, cancelled
- **FR3.6:** System shall support priority levels: low, medium, high, critical
- **FR3.7:** Users shall be able to add/remove team members to projects

3.1.4 FR4: Task Management

- **FR4.1:** Users shall be able to create tasks within projects
- **FR4.2:** Users shall be able to assign tasks to team members
- **FR4.3:** Users shall be able to update task details
- **FR4.4:** Users shall be able to change task status via Kanban board
- **FR4.5:** System shall support task statuses: todo, in-progress, in-review, completed
- **FR4.6:** Users shall be able to set task due dates
- **FR4.7:** Users shall be able to add subtasks
- **FR4.8:** Users shall be able to attach files to tasks

3.1.5 FR5: Dashboard

- **FR5.1:** System shall display total project count
- **FR5.2:** System shall display total task count
- **FR5.3:** System shall display completed tasks count
- **FR5.4:** System shall show recent activity feed
- **FR5.5:** System shall show upcoming deadlines
- **FR5.6:** Managers shall be able to view team performance metrics

3.1.6 FR6: Team Collaboration

- **FR6.1:** Users shall be able to view all team members
- **FR6.2:** Users shall be able to search team members
- **FR6.3:** Users shall be able to add comments to tasks
- **FR6.4:** Users shall be able to edit/delete their own comments

3.2 Non-Functional Requirements

3.2.1 NFR1: Performance

- System shall respond to user actions within 2 seconds
- API endpoints shall respond within 500ms under normal load
- System shall support at least 100 concurrent users
- Database queries shall be optimized with proper indexing

3.2.2 NFR2: Security

- All passwords shall be hashed using bcrypt
- Authentication shall use JWT tokens
- API endpoints shall be protected with authentication middleware
- File uploads shall be validated for type and size
- System shall prevent SQL injection and XSS attacks

3.2.3 NFR3: Usability

- UI shall be intuitive and user-friendly
- System shall be responsive for mobile and desktop devices
- Error messages shall be clear and helpful
- Navigation shall be consistent across all pages

3.2.4 NFR4: Reliability

- System shall have 99% uptime
- Database connections shall have retry logic
- System shall handle errors gracefully
- Data shall be validated before storage

3.2.5 NFR5: Maintainability

- Code shall be well-documented with comments
- System shall follow MVC architecture pattern
- API shall follow RESTful conventions
- Docker containers shall enable easy deployment

3.2.6 NFR6: Scalability

- System architecture shall support horizontal scaling
- Database shall support replication
- Stateless API design shall enable load balancing

4 System Architecture

4.1 Technology Stack

- **Frontend:** React 18, Vite, TailwindCSS, React Router, Axios
- **Backend:** Node.js 20, Express.js, Mongoose ODM
- **Database:** MongoDB
- **Authentication:** JWT (jsonwebtoken), bcryptjs
- **Validation:** express-validator
- **File Upload:** multer
- **Containerization:** Docker, Docker Compose
- **Web Server:** Nginx (production frontend)

4.2 System Components

4.2.1 Frontend Application

- Single-page application built with React
- Component-based architecture
- Context API for state management
- Protected and public route handling
- Responsive design with TailwindCSS

4.2.2 Backend API

- RESTful API architecture
- MVC pattern implementation
- JWT-based authentication middleware
- Request validation middleware
- File upload handling
- Error handling and logging

4.2.3 Database

- MongoDB document database
- Mongoose ODM for schema definition
- Collections: Users, Projects, Tasks, Comments
- Indexed fields for performance
- Data relationships and population

4.3 API Endpoints

4.3.1 Authentication Routes

- POST /api/auth/register - User registration
- POST /api/auth/login - User login
- GET /api/auth/me - Get current user
- PUT /api/auth/profile - Update profile
- PUT /api/auth/password - Change password

4.3.2 Project Routes

- GET /api/projects - Get all projects
- GET /api/projects/:id - Get single project
- POST /api/projects - Create project
- PUT /api/projects/:id - Update project
- DELETE /api/projects/:id - Delete project
- POST /api/projects/:id/members - Add member
- DELETE /api/projects/:id/members/:userId - Remove member

4.3.3 Task Routes

- GET /api/tasks - Get all tasks
- GET /api/tasks/my-tasks - Get user's tasks
- GET /api/tasks/project/:projectId - Get project tasks
- GET /api/tasks/:id - Get single task
- POST /api/tasks - Create task
- PUT /api/tasks/:id - Update task
- PATCH /api/tasks/:id/status - Update task status
- DELETE /api/tasks/:id - Delete task

4.3.4 Dashboard Routes

- GET /api/dashboard/stats - Get statistics
- GET /api/dashboard/activity - Get recent activity
- GET /api/dashboard/deadlines - Get upcoming deadlines
- GET /api/dashboard/team-performance - Get team metrics

5 Data Models

5.1 User Schema

- name: String (required)
- email: String (required, unique)
- password: String (required, hashed)
- avatar: String
- role: String (user, manager, admin)
- department: String
- isActive: Boolean
- lastLogin: Date

5.2 Project Schema

- name: String (required)
- description: String
- status: String (planning, in-progress, on-hold, completed, cancelled)
- priority: String (low, medium, high, critical)
- owner: ObjectId (ref: User)
- members: Array of ObjectId (ref: User)
- startDate: Date
- dueDate: Date
- color: String
- tags: Array of String

5.3 Task Schema

- title: String (required)
- description: String
- status: String (todo, in-progress, in-review, completed)
- priority: String (low, medium, high, critical)
- project: ObjectId (ref: Project)
- assignee: ObjectId (ref: User)
- creator: ObjectId (ref: User)
- dueDate: Date
- subtasks: Array of Subtask objects
- attachments: Array of Attachment objects

5.4 Comment Schema

- content: String (required)
- task: ObjectId (ref: Task)
- author: ObjectId (ref: User)
- mentions: Array of ObjectId (ref: User)
- parentComment: ObjectId (ref: Comment)
- isEdited: Boolean
- attachments: Array of objects

6 Deployment

6.1 Docker Configuration

The system uses Docker Compose to orchestrate three services:

- **MongoDB:** Database service with persistent volume
- **Backend:** Node.js API server
- **Frontend:** React application served by Nginx

6.2 Environment Variables

- `NODE_ENV`: development/production
- `MONGO_URI`: MongoDB connection string
- `JWT_SECRET`: Secret key for JWT signing
- `PORT`: Server port (default: 5000)
- `MAX_FILE_SIZE`: Maximum upload size

6.3 Deployment Commands

- Development: `docker-compose up --build`
- Production: `docker-compose -f docker-compose.yml -f docker-compose.prod.yml up`
- Stop: `docker-compose down`
- Clean: `docker-compose down -v`

7 Testing Requirements

7.1 Unit Testing

- Test individual controllers and middleware
- Test API endpoints with mock data
- Test React components

7.2 Integration Testing

- Test API and database integration
- Test authentication flow
- Test file upload functionality

7.3 User Acceptance Testing

- Test all user workflows
- Test on different browsers and devices
- Verify all functional requirements

8 Future Enhancements

- Real-time notifications using WebSockets
- Email notifications for task assignments
- Calendar view for task deadlines
- Gantt chart visualization
- Time tracking functionality
- File version control
- Advanced reporting and analytics
- Mobile application (React Native)
- Integration with third-party tools (Slack, GitHub)

9 Appendices

9.1 Appendix A: Glossary

Additional terms and definitions used throughout the system.

9.2 Appendix B: Use Case Diagrams

Visual representations of system interactions (to be added).

9.3 Appendix C: Database ERD

Entity-Relationship Diagram showing database structure (to be added).