****

**Task Management System Documentation**

**Name: Laxit Khanpara**

**Email:** [**laxitkhanpara36464@gmail.com**](mailto:laxitkhanpara36464@gmail.com)

**Linkedin:** [**www.linkedin.com/in/laxit-khanpara**](http://www.linkedin.com/in/laxit-khanpara)

**Github:** [**https://github.com/laxitkhanpara/Stamurai-Task-Management-System.git**](https://github.com/laxitkhanpara/Stamurai-Task-Management-System.git)

**Live Project Link:** [**https://stamurai-tms.onrender.com**](https://stamurai-tms.onrender.com)

**Api Link:** [**https://stamurai-task-management-system-w5a1.onrender.com**](https://stamurai-task-management-system-w5a1.onrender.com)

## **Login Credentials:**

**Admin**

* **Email**: [anshulagarwal@stamurai.com](mailto:anshulagarwal@stamurai.com)
* **Password**: 123456

**Manager**

* **Email**: [harshtyagi@stamurai.com](mailto:harshtyagi@stamurai.com)
* **Password**: 123456

**User**

* **Email**: [anoopdutta@stamurai.com](mailto:anoopdutta@stamurai.com)
* **Password**: 123456

**Table of Contents**

1. [Introduction](https://claude.ai/chat/6adfa0b4-8408-461f-a4eb-4b8511d6da66#introduction)
2. [System Architecture](https://claude.ai/chat/6adfa0b4-8408-461f-a4eb-4b8511d6da66#system-architecture)
3. [Features and Functionalities](https://claude.ai/chat/6adfa0b4-8408-461f-a4eb-4b8511d6da66#features-and-functionalities)
4. [Technical Implementation](https://claude.ai/chat/6adfa0b4-8408-461f-a4eb-4b8511d6da66#technical-implementation)
5. [Database Schema](https://claude.ai/chat/6adfa0b4-8408-461f-a4eb-4b8511d6da66#database-schema)
6. [API Documentation](https://claude.ai/chat/6adfa0b4-8408-461f-a4eb-4b8511d6da66#api-documentation)
7. [Frontend Components](https://claude.ai/chat/6adfa0b4-8408-461f-a4eb-4b8511d6da66#frontend-components)
8. [Authentication and Authorization](https://claude.ai/chat/6adfa0b4-8408-461f-a4eb-4b8511d6da66#authentication-and-authorization)
9. [Real-time Notifications](https://claude.ai/chat/6adfa0b4-8408-461f-a4eb-4b8511d6da66#real-time-notifications)
10. [State Management](https://claude.ai/chat/6adfa0b4-8408-461f-a4eb-4b8511d6da66#state-management)
11. [Deployment](https://claude.ai/chat/6adfa0b4-8408-461f-a4eb-4b8511d6da66#deployment)
12. [Future Enhancements](https://claude.ai/chat/6adfa0b4-8408-461f-a4eb-4b8511d6da66#future-enhancements)

**Introduction**

The Task Management System is a comprehensive web application designed to help small teams efficiently organize, assign, and track tasks. This document provides a detailed overview of the system's architecture, features, and implementation.

**Project Overview**

The Task Management System enables team members to create tasks, assign them to colleagues, track progress, and receive notifications about task updates. The application features a user-friendly interface with a dashboard that provides a quick overview of tasks based on various parameters such as due date, priority, and status.

**Target Users**

* Small to medium-sized teams
* Project managers
* Team leaders
* Individual contributors

**System Architecture**

The Task Management System follows a modern web application architecture with separate frontend and backend components, connected via RESTful APIs.

**Technology Stack**

* **Frontend**: Next.js (React framework)
* **State Management**: Redux Toolkit
* **Backend**: Node.js with Express.js
* **Database**: MongoDB
* **Real-time Communication**: Socket.io
* **Authentication**: JWT (JSON Web Tokens)
* **Version Control**: Git

**Features and Functionalities**

**User Authentication**

* Secure user registration with email verification
* Login with JWT authentication
* Password reset functionality
* Role-based access control (Admin, Manager, Team Member)

**Task Management**

* Create new tasks with detailed information:
  + Title
  + Description
  + Due date
  + Priority (Low, Medium, High, Urgent)
  + Status (To Do, In Progress, Review, Completed)
* Edit existing tasks
* Delete tasks (with appropriate permissions)
* View task details
* Track task history and changes

**Team Collaboration**

* Assign tasks to team members
* Transfer task ownership
* Add comments to tasks
* @mention functionality to notify specific team members
* Task discussion threads

**Dashboard**

The dashboard provides a centralized view of:

* Tasks assigned to the logged-in user
* Tasks created by the logged-in user
* Overdue tasks
* Tasks grouped by status
* Tasks sorted by priority
* Recent activities

**Search and Filtering**

* Search tasks by title or description
* Filter tasks by:
  + Status
  + Priority
  + Due date
  + Assignee
  + Creator
* Sort tasks by various parameters

**Notifications**

* Real-time notifications using Socket.io
* Email notifications for critical updates
* Notification preferences management
* In-app notification center

**Technical Implementation**

**Frontend Architecture**

The frontend is built using Next.js, which provides server-side rendering capabilities and an optimized React framework. The application follows a component-based architecture with reusable UI elements.

**Key Frontend Libraries**

* **Redux Toolkit**: For state management
* **React Hook Form**: For form validation
* **Axios**: For API requests
* **Socket.io-client**: For real-time communications
* **React Icons**: For UI icons
* **React Datepicker**: For date selection

**Backend Architecture**

The backend follows an MVC (Model-View-Controller) architecture pattern, with clear separation of concerns:

* **Models**: Define the data structure and database interactions
* **Controllers**: Handle business logic and request processing
* **Routes**: Define API endpoints and route requests to appropriate controllers
* **Middlewares** : Handle authentication, logging, error handling, etc.

**Database Schema**

**User Collection**

{

"\_id": "ObjectId",

"name": "String",

"email": "String",

"password": "String (hashed)",

"role": "String (admin, manager, member)",

"profilePicture": "String (URL)",

"createdAt": "Date",

"updatedAt": "Date"

}

**Task Collection**

{

"\_id": "ObjectId",

"title": "String",

"description": "String",

"createdBy": "ObjectId (ref: User)",

"assignedTo": "ObjectId (ref: User)",

"dueDate": "Date",

"priority": "String (low, medium, high, urgent)",

"status": "String (todo, in-progress, review, completed)",

"attachments": ["String (URLs)"],

"comments": [{

"user": "ObjectId (ref: User)",

"text": "String",

"createdAt": "Date"

}],

"createdAt": "Date",

"updatedAt": "Date"

}

**Notification Collection**

{

"\_id": "ObjectId",

"recipient": "ObjectId (ref: User)",

"sender": "ObjectId (ref: User)",

"task": "ObjectId (ref: Task)",

"type": "String (task-assigned, comment-added, etc.)",

"message": "String",

"read": "Boolean",

"createdAt": "Date"

}

**API Documentation**

**Authentication Endpoints**

**User Registration**

* **Route**: POST /api/auth/register
* **Body**:
* { "name": "String", "email": "String", "password": "String", "role": "String (optional)"}
* **Response**: User object with JWT token

**User Login**

* **Route**: POST /api/auth/login
* **Body**:
* { "email": "String", "password": "String"}
* **Response**: User object with JWT token

**Task Endpoints**

**Create Task**

* **Route**: POST /api/tasks
* **Authentication**: Required
* **Body**:
* { "title": "String", "description": "String", "assignedTo": "ObjectId (optional)", "dueDate": "Date", "priority": "String", "status": "String"}
* **Response**: Created task object

**Get All Tasks**

* **Route**: GET /api/tasks
* **Authentication**: Required
* **Query Parameters**:
  + status: Filter by status
  + priority: Filter by priority
  + assignedTo: Filter by assignee
  + search: Search in title or description
* **Response**: Array of task objects

**Get Task by ID**

* **Route**: GET /api/tasks/:id
* **Authentication**: Required
* **Response**: Task object

**Update Task**

* **Route**: PUT /api/tasks/:id
* **Authentication**: Required
* **Body**: Fields to update
* **Response**: Updated task object

**Delete Task**

* **Route**: DELETE /api/tasks/:id
* **Authentication**: Required
* **Response**: Success message

**User Endpoints**

**Get All Users**

* **Route**: GET /api/users
* **Authentication**: Required (Admin only)
* **Response**: Array of user objects

**Get User Profile**

* **Route**: GET /api/users/profile
* **Authentication**: Required
* **Response**: User object

**Update User Profile**

* **Route**: PUT /api/users/profile
* **Authentication**: Required
* **Body**: Fields to update
* **Response**: Updated user object

**Notification Endpoints**

**Get User Notifications**

* **Route**: GET /api/notifications
* **Authentication**: Required
* **Response**: Array of notification objects

**Mark Notification as Read**

* **Route**: PUT /api/notifications/:id/read
* **Authentication**: Required
* **Response**: Updated notification object

**Key UI Screens**

1. **Login/Registration**: User authentication screens
2. **Dashboard**: Overview of tasks and statistics
3. **Task List**: Filterable list of all accessible tasks
4. **Task Details**: Detailed view of a specific task
5. **Create/Edit Task**: Form for task creation and editing
6. **User Profile**: User information and settings
7. **Admin Panel**: User management for administrators

**Authentication and Authorization**

**Authentication Flow**

1. User submits login credentials
2. Server validates credentials and generates JWT token
3. Token is stored in local storage and included in API requests
4. Token expiration is handled with automatic refresh or logout

**Role-Based Access Control**

The system implements three primary roles:

1. **Admin**:
   * Full system access
   * User management
   * System settings
   * All task operations
2. **Manager**:
   * Create projects and tasks
   * Assign tasks to team members
   * View all team tasks
   * Generate reports
3. **Team Member**:
   * View assigned tasks
   * Update task status
   * Create tasks for self
   * Comment on tasks

**Permission Structure**

const permissions = {

'create:task': ['admin', 'manager', 'member'],

'read:task': ['admin', 'manager', 'member'],

'update:task': ['admin', 'manager', 'member'],

'delete:task': ['admin', 'manager'],

'assign:task': ['admin', 'manager'],

'create:user': ['admin'],

'read:user': ['admin'],

'update:user': ['admin'],

'delete:user': ['admin']

};

**Real-time Notifications**

The notification system uses Socket.io to provide real-time updates to users.

**Socket.io Implementation**

1. Socket connection is established upon user login
2. Users join a personal room based on their user ID
3. Server emits events to specific users or rooms when relevant actions occur
4. Client listens for events and updates the UI accordingly

**Notification Types**

* Task assignment
* Task status change
* Due date reminders
* Comment mentions
* Task completion

**State Management**

The application uses Redux Toolkit for state management, providing a centralized store for application data.

**Deployment**

**Deployment Architecture**

The application is deployed using a modern cloud infrastructure:

We have used <https://render.com/> for deploy stamurai application.

**Deployment Process**

1. **Frontend**: Deployed on render with continuous integration from GitHub
2. **Backend**: Deployed on render with automatic deployments
3. **Database**: Hosted on MongoDB Atlas with proper backup strategies
4. **Environment Variables**: Managed securely in deployment platforms

**Future Enhancements**

1. **Enhanced Reporting**: Generate detailed reports on task completion, user productivity, etc.
2. **Time Tracking**: Add time tracking functionality for tasks
3. **Project Management**: Group tasks into projects with project-level metrics
4. **Mobile Application**: Develop a companion mobile app
5. **Integration with Third-party Tools**: Calendar, email, Slack, etc.
6. **Advanced Analytics**: Task completion trends, bottleneck identification
7. **File Management**: Enhanced file attachment and organization
8. **Custom Workflows**: Allow teams to create custom task workflows

**Conclusion**

The Task Management System provides a robust solution for small teams to effectively manage their tasks and collaborate. With features like real-time notifications, role-based access control, and comprehensive task management capabilities, the system streamlines team productivity and project tracking.

**Developer's Note**

This project has been implemented with a strong focus on code quality and maintainability:

* **Optimized Code**: The codebase has been structured for optimal performance and scalability, with careful attention to efficient rendering and data fetching patterns.
* **High Readability**: Clean, well-commented code with consistent formatting and naming conventions ensures easy understanding and maintenance.
* **Redux Toolkit Implementation**: State management is handled through a well-structured Redux Toolkit setup, with proper organization of slices, actions, and selectors.
* **Comprehensive Documentation**: For more detailed information about installation, setup, and usage, please refer to the README.md file in the GitHub repository: <https://github.com/laxitkhanpara/Stamurai-Task-Management-System.git>

While the current implementation meets all requirements, there is always room for further optimization and enhancement as the project evolves.