Smart Task Monitoring System

Mason Hawkins

03-18-2025

System Implementation

Table of Contents

Introduction:

The **Smart Task Monitoring System** is a task management solution designed to improve team efficiency by providing an intuitive task assignment and tracking system. The system allows **Admins, Team Members, and Assignees** to collaborate seamlessly, ensuring real-time task tracking and notifications.

Problem Statement & System Requirements

Problem Statement:

Before the development of this system, users faced challenges such as:

- Lack of a centralized platform for managing team tasks.
- Inefficient manual task assignment and tracking.
- No real-time notifications for assigned or completed tasks.
- No structured access control for different roles.

System Requirements:

Hardware Requirements:

- Processor: Intel Core i5 or higher
- RAM: 8GB or higher
- Storage: Minimum 500MB of available space
- Operating System: Windows 10, macOS, or Linux

Software Requirements:

- **Java 17+** (For back-end development)
- **JavaFX** (For the GUI)

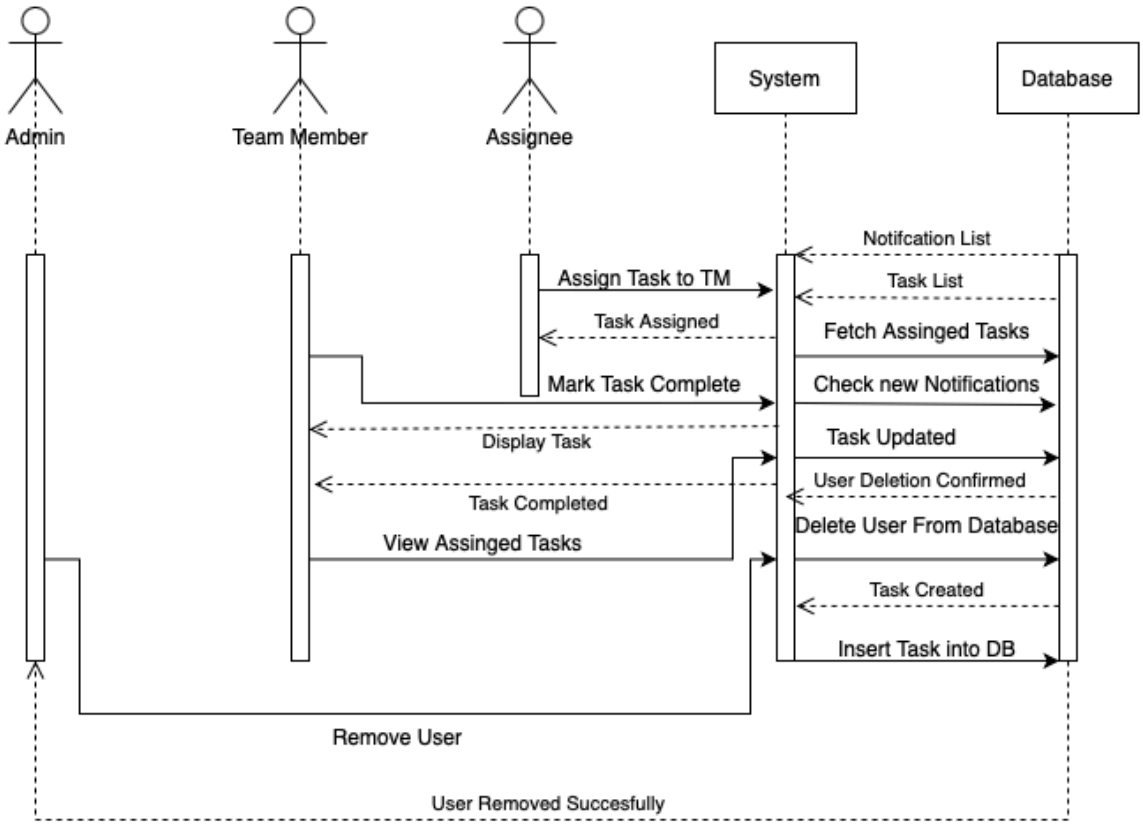
- **MySQL Database** (For data storage)
- **IntelliJ IDE or Eclipse** (For development)

Functional Requirements Specifications

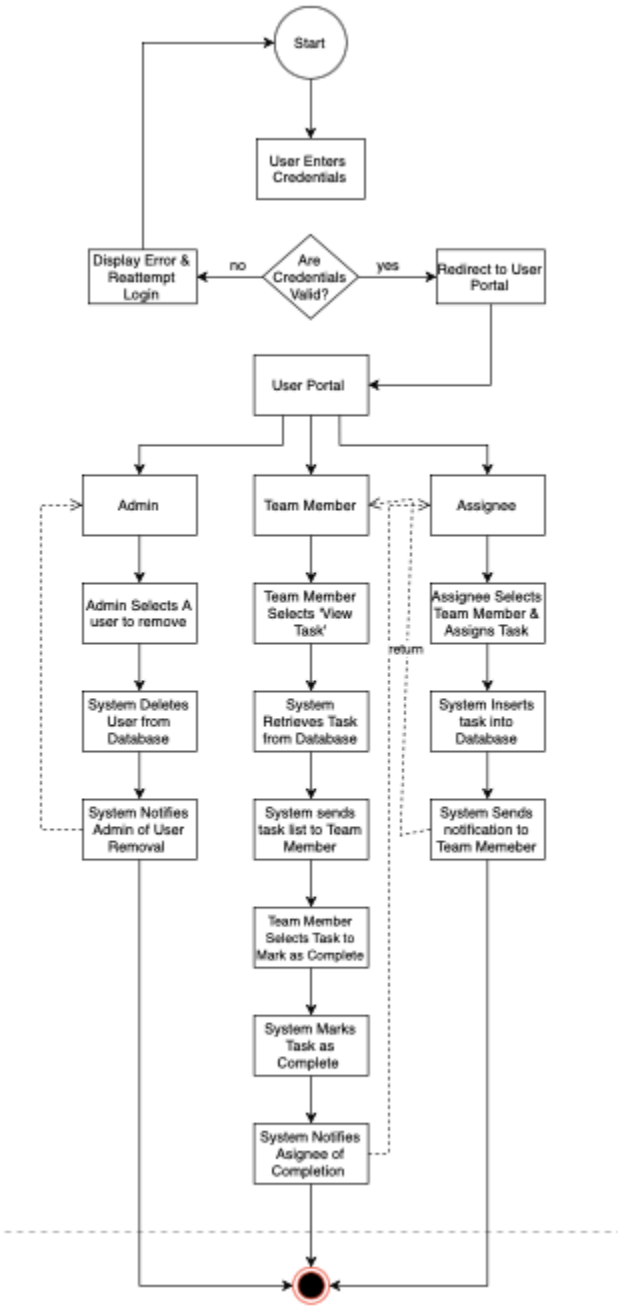
Functional Features:

1. **User Registration & Authentication**
    - Users can create accounts and log in based on their role.
2. **Task Management**
    - Team Members can create personal tasks.
    - Assignees can assign tasks to Team Members.
    - Users can mark tasks as complete.
3. **Notification System**
    - Users receive notifications for assigned and completed tasks.
    - Users can clear notifications.
4. **Admin Features**
    - Remove users from the system.
    - Clear individual or all tasks.
5. **Return to Main Menu Option**
    - Users can return to the main menu without restarting the application.

System Sequence Diagram



Activity Diagram



User Interface Specification

UI Component	Description
Login Page	User enters credentials to access the system.
Task Dashboard	Displays all pending tasks with statuses
Task Creation Page	Allows Team Members to create personal Tasks
Task Assignment Page	Assignees can assign tasks to team members
Notification Tab	Users see updates on task assignments and completions.

Project Plan

Milestone	Status
System Design	Completed
User Registration/Login	Completed
Task Creation and Assignment	Completed
Notification System	Completed
Return to Main Menu	Completed
UI Enhacments	In Progress
Final Testing	In Progress

Implementation Planning

The implementation followed an iterative approach, starting with database schema design, followed by core DAO and backend logic, and ending with GUI integration. Key milestones included user role logic, task assignment features, notification handling, and a Swing-based user interface.

System Architecture and System Design

The system uses a modular architecture with:

- A MySQL database for persistent storage
- Java with a DAO (Data Access Object) pattern to separate data handling logic
- A Swing based GUI for user interaction: Roles (Admin, Assignee, Team Member) are enforced through conditional logic in both the backend and UI layers. Each feature is encapsulated in individual classes promoting reusability and scalability.

Algorithms and Data Model Implmentation

Core algorithms include:

- Task creation and completion with notification triggers
- User authentication with role-based access
- Notification delivery based on task status changes: Data models are implemented as plain Java objects (User, Task) backed by DAO classes (UserDAO, TaskDAO, NotificationsDAO, AdminDAO).

User Interface Design and Implmentation

The system uses Java Swing GUI implementation. Key interfaces include:

- Login and registration screens
- Role-specific dashboards (Admin, Assignee, Team Member)

- Pop-up dialogs for input and confirmation

Design of Tests

Manual testing was conducted for:

- Task assignment, creation, and completion
- Login validation and role based access control
- Notification creation, delivery, and clearance: Unit tests were informally performed on DAO methods to confirm successful database transactions and error handling.

Error Handling and Debugging

Each DAO method includes try-catch blocks with exception printing for runtime debugging. Input validation is performed before executing database actions (e.g., checking user/task existence). User feedback is provided via pop-up dialogs for errors and confirmations.

Collaboration and Code Integration

Code was developed in modules (User, Task, Notifications, GUI). Integration was done by calling respective DAO methods within GUI event listeners. Portals were tested individually before integration into the main TaskManagerGUI. A consistent coding style and structure facilitated seamless merging of components.

Performance Optimization Plan

While performance is sufficient for small to medium datasets, future improvements could include:

- Pagination for task/ notification views
- Indexing critical database columns (user_id, task_id)
- Lazy loading of notifications

Maintenance and Support Strategy

The modular design (DAO, portal classes, styling class) supports easy maintenance. To support future changes:

- Styling can be updated in StyleManager without touching functional code
- New roles or features can be added by extending the existing menu and DAO patterns
- Database changes require minimal code updates due to the abstraction layer

References:

- JavaFX
- MySQL
- Draw.io
- Java

Highlighted Changes

- Added ‘Return to Main Menu’ feature for better usability.
- Implemented a Notification Tab for Team Members and Assignees.
- Ensured tasks are displayed before marking them as complete.