Kingdom of Saudi Arabia
Ministry of Education
Najran University
Faculty of Computer Science and Information Systems
Department of Computer Science

المملكة العربية السعودية
وزارة التعليم
جامعة نجران
كلية علوم الحاسب ونظم المعلومات
قسم علوم الحاسب

First Semester (2025–2026)

Software Engineering (4513 CSS)

Group Project

Student Task Manager System (STMS)

Academic Task Management & Collaboration

Team/Student:

Student Name	Student ID
نوير خالد مسعود الوايلي	443304167
سامية حسين ال عباس	443404259
ايمان عبدالشكور عمر	444305249
ريضه محمد فلاح العجمي	443300751
لطيفه مبارك	443407106

Table of contents:

1. Introduction

1.1 Purpose

The purpose of the **Student Task Manager System (STMS)** is to enhance students' productivity by providing a centralized platform for managing academic tasks such as assignments, exams, and projects. The system aims to:

- Reduce missed deadlines
- Enable effective collaboration in study groups
- Provide automated reminders and progress tracking for tasks

1.2 Scope

Target Users:

- University and school students
- Study groups working on team projects

Included Features:

- Task creation, editing, and deletion
- Categorization of tasks (Assignments, Exams, Projects)
- Calendar and list views for task tracking
- Automated reminders and notifications
- Study group collaboration and resource sharing

Excluded Features (Future Work):

- Mobile app version
- Integration with Learning Management Systems (LMS) such as Blackboard

1.3 Definitions, Acronyms, Abbreviations

- **STMS:** Student Task Manager System
- **FR:** Functional Requirement
- **NFR:** Non-Functional Requirement
- **LMS:** Learning Management System
- **UML**: Unified Modeling Language
- **UC**: Use Case

1.4 References

1. IEEE Software Engineering Standards
   https://standards.ieee.org
2. UML 2.5 Specification — Object Management Group
   https://www.omg.org/spec/UML/2.5
3. PlantUML Documentation
   https://plantuml.com
4. diagrams.net (draw.io)
   https://app.diagrams.net

1.5 Overview

This document presents the system design and documentation of STMS, covering:

- Requirement analysis (functional and non-functional)
- System design through UML diagrams (class, use case, sequence, activity, and state diagrams)
- Specific requirements, acceptance criteria, and traceability
- Change management and appendices including glossary, meeting notes, and UI sketches

2. General Description

2.1 Product Perspective

The Student Task Manager System (STMS) is a standalone web-based application designed to help students manage their academic responsibilities. The system operates independently but follows a modular design that allows future integration with external platforms such as Learning Management Systems (LMS), school portals, or mobile applications.

It communicates through a secure backend API and stores all data in a centralized database. Future versions may include synchronization with cloud services, cross-device access, and mobile push notifications.

2.2 Product Functions

The system provides a clear set of core features to help students organize their tasks. These include:

- User registration, login, and password recovery
- Creating, editing, deleting, and categorizing tasks
- Calendar and list views for displaying tasks
- Automatic reminders and user-defined notifications
- Creating and joining study groups, sharing resources, and collaborating on tasks
- Viewing progress summaries and basic productivity analytics

2.3 User Characteristics

The primary users are students aged 16 to 25 with basic computer and internet skills.

They are comfortable using web browsers and email services.

These users typically manage assignments, exams, quizzes, and group projects, and may use the system frequently to organize academic responsibilities.

2.4 General Constraints

The system must follow several constraints:

- Works only on modern browsers such as Chrome, Firefox, and Edge
- Requires a stable internet connection
- Must ensure secure communication using HTTPS
- Must maintain acceptable performance and response times
- Depends on scheduled background processes for sending reminders

2.5 Assumptions and Dependencies

The system depends on several assumptions:

- Users have access to a valid email address for verification and notifications
- A reliable database server is available to store tasks and user data
- An active email service provider exists for sending automated reminders
- Future expansions may depend on APIs or LMS integrations

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interface

- The system shall provide a responsive web interface accessible through modern browsers (Chrome, Firefox, Edge).
- The UI shall include navigation menus for Tasks, Calendar, Groups, and Analytics.
- The system shall provide form-based interfaces for adding, editing, and viewing tasks.

3.1.2 Hardware Interfaces

- No special hardware required; only a device capable of running a modern browser.

3.1.3 Software Interfaces

- The system shall connect to a relational database (e.g., MySQL or PostgreSQL).
- The system shall use an email service (SMTP or API provider) for sending reminders.

3.1.4 Communication Interfaces

- All communications shall use HTTPS for secure data transmission.
- The system shall support RESTful APIs for potential future integrations.

3.2 Functional Requirements (FR)

FR-ID	Requirement (shall)	Notes / Source
FR-01	The system shall allow users to register using email and password.	UC-UserRegistration
FR-02	The system shall allow users to log in and log out securely.	UC-Login
FR-03	The system shall allow users to reset their password via email verification.	UC-PasswordReset
FR-04	The system shall allow users to create a new task.	UC-01
FR-05	The system shall allow users to edit an existing task.	UC-01

FR-06	The system shall allow users to delete a task.	UC-01
FR-07	The system shall categorize tasks by type (assignment, exam, project, etc.).	UC-01
FR-08	The system shall display tasks in list and calendar views.	UC-01, UI-Design
FR-09	The system shall send automatic reminders before task deadlines.	UC-02
FR-10	The system shall allow users to set custom reminder times.	UC-02
FR-11	The system shall allow users to create study groups.	UC-03
FR-12	The system shall allow users to join existing study groups.	UC-03
FR-13	The system shall allow group members to share resources.	UC-03
FR-14	The system shall provide progress analytics such as completed tasks.	UC-Analytics
FR-15	The system shall display task completion charts and statistics.	UC-Analytics

3.3 Non-Functional Requirements (measurable)

NFR-ID	Requirement	Metric / Target (measurable)
NFR-P1	System performance must support fast responses.	95% of requests ≤ 2 seconds; worst case ≤ 5 seconds.
NFR-P2	The system shall support concurrent users without performance loss.	Minimum 1,000 simultaneous users.
NFR-S1	User credentials shall be stored securely.	Password hashing using SHA-256.
NFR-S2	All communication with the server shall be encrypted.	100% of requests served over HTTPS.

NFR-U1	The system user interface shall be easy to use.	New users complete main task within ≤ 5 minutes.
NFR-U2	The system shall support major web browsers.	Chrome, Firefox, Edge (latest 2 versions).
NFR-R1	System availability must remain high.	99% uptime excluding maintenance.
NFR-R2	Task changes shall be preserved automatically.	Auto-save triggered every ≤ 10 seconds.
NFR-SC1	The system shall support future scalability.	Architecture supports API expansion and mobile integration.

3.4 Data / Logical Model — Key Entities and Fields

The Student Task Manager System (STMS) contains several key entities to manage users, tasks, reminders, study groups, and shared resources.

- **User:** This entity stores student account information. Its main fields are user_id (primary key), name, email, and password_hash. A user can create multiple tasks and join multiple study groups.
- **Task:** This entity stores all task-related information. Each task is associated with a single user (user_id) and has fields including task_id (primary key), title, description, due_date, category, priority, and status. Each task can also have a reminder linked to it.
- **Reminder:** This entity manages scheduled notifications for tasks. It contains reminder_id (primary key), task_id (foreign key), and reminder_time. Each task has at most one reminder.
- **StudyGroup:** This entity represents collaborative study groups. It has fields such as group_id (primary key), group_name, and created_by (foreign key referring to the user who created the group).
- **GroupMember:** This entity manages user membership in study groups. It includes group_id and user_id as foreign keys, and a role field to define permissions or roles within the group.

- **SharedResource:** This entity stores files or resources shared within study groups. Its main fields are resource_id (primary key), group_id (foreign key), file_link, and description. Each study group can have multiple shared resources.

Relationships Overview:

- A single user can create multiple tasks (one-to-many).
- Each task has one reminder (one-to-one).
- Users and study groups have a many-to-many relationship through the GroupMember entity.
- Each study group can have multiple shared resources (one-to-many).

3.5 Use Cases

UC-01: Manage Tasks

- **Actor:** Student
- **Goal:** Create, edit, and delete academic tasks.
- **Precondition:** User is logged in.
- **Main Flow:**
    1. Student selects “Add Task.”
    2. Enters task details (title, description, due date, category, priority).
    3. System saves the task and displays it in the task list or calendar view.
- **Linked FRs:** FR-04, FR-05, FR-06, FR-07, FR-08

UC-02: Receive Task Reminders

- **Actor:** Student
- **Goal:** Receive automatic and custom reminders for tasks.
- **Precondition:** Tasks exist with upcoming deadlines.
- **Main Flow:**
    1. System checks tasks nearing their due dates.
    2. Sends automated email or in-app notifications.

3. Student receives notification and can view task details.

- **Linked FRs:** FR-09, FR-10

UC-03: Join Study Group

- **Actor:** Student
- **Goal:** Participate in study groups and collaborate on shared tasks.
- **Precondition:** Student is registered and logged in.
- **Main Flow:**
    1. Student searches for or selects a study group.
    2. System verifies membership and adds the student to the group.
    3. Student can view and share resources with other group members.
- **Linked FRs:** FR-11, FR-12, FR-13

UC-04: Track Progress

- **Actor:** Student
- **Goal:** Monitor task completion and progress over time.
- **Precondition:** Student has tasks recorded in the system.
- **Main Flow:**
    1. Student requests a progress summary or report.
    2. System calculates completed vs pending tasks.
    3. Generates charts or tables showing productivity trends.
- **Linked FRs:** FR-14, FR-15

3.6 Acceptance Criteria

Scenario 1: Task Creation

- **Input:** Student adds a new task with a title, due date, category, and priority.
- **Expected Result:** The system saves the task and displays it in the task list or calendar within **2 seconds**.
- **FR Covered:** FR-04, FR-07, FR-08

Scenario 2: Task Editing

- **Input:** Student edits an existing task's title, due date, or category.
- **Expected Result:** The system updates the task correctly and reflects the changes immediately in the list/calendar.
- **FR Covered:** FR-05

Scenario 3: Task Deletion

- **Input:** Student deletes a task.
- **Expected Result:** The system removes the task from all views within **1 second** and confirms deletion to the user.
- **FR Covered:** FR-06

Scenario 4: Reminder Notification

- **Input:** A task's deadline is approaching within the defined reminder period.
- **Expected Result:** The system sends an email or in-app notification to the student **at least 1 hour before the deadline**.
- **FR Covered:** FR-09, FR-10

Scenario 5: Study Group Joining

- **Input:** Student requests to join a study group.
- **Expected Result:** The system verifies membership, adds the student to the group, and displays shared resources **within 3 seconds**.
- **FR Covered:** FR-12, FR-13

Scenario 6: Progress Tracking

- **Input:** Student requests a progress report.
- **Expected Result:** The system generates task completion charts and statistics **accurately and within 2 seconds**.
- **FR Covered:** FR-14, FR-15

3.7 Traceability Matrix

Functional Requirements to Design Artifacts Traceability

FR-ID	Use Case (UC)	Sequence Diagram	Class Diagram	Activity Diagram	State Diagram
FR-01	UC-User Registration	Seq-UserReg	User	Act-UserReg	State-User
FR-02	UC-Login	Seq-Login	User	Act-Login	State-User
FR-03	UC-Password Reset	Seq-PwdReset	User	Act-PwdReset	State-User
FR-04	UC-Manage Tasks	Seq-TaskCreate	Task	Act-TaskFlow	State-Task
FR-05	UC-Manage Tasks	Seq-TaskEdit	Task	Act-TaskFlow	State-Task
FR-06	UC-Manage Tasks	Seq-TaskDelete	Task	Act-TaskFlow	State-Task
FR-07	UC-Manage Tasks	Seq-TaskCreate	Task	Act-TaskFlow	State-Task
FR-08	UC-Manage Tasks	Seq-TaskView	Task	Act-TaskFlow	State-Task
FR-09	UC-Receive Reminders	Seq-Reminder	Reminder	Act-Reminder	State-Reminder
FR-10	UC-Receive Reminders	Seq-Reminder	Reminder	Act-Reminder	State-Reminder
FR-11	UC-Join Study Group	Seq-GroupCreate	StudyGroup	Act-Group	State-Group
FR-12	UC-Join Study Group	Seq-GroupJoin	StudyGroup	Act-Group	State-Group
FR-13	UC-Join Study Group	Seq-ShareResource	SharedResource	Act-Group	State-Group
FR-14	UC-Track Progress	Seq-ProgressReport	Task, AnalyticsService	Act-Analytics	State-Task
FR-15	UC-Track Progress	Seq-ProgressReport	Task, AnalyticsService	Act-Analytics	State-Task

4. Analysis Models (UML)

4.1 Use Case Diagram

The Use Case Diagram illustrates the interactions between students and the system. It shows the main actors and their goals, as well as relationships between use cases.

Actors:

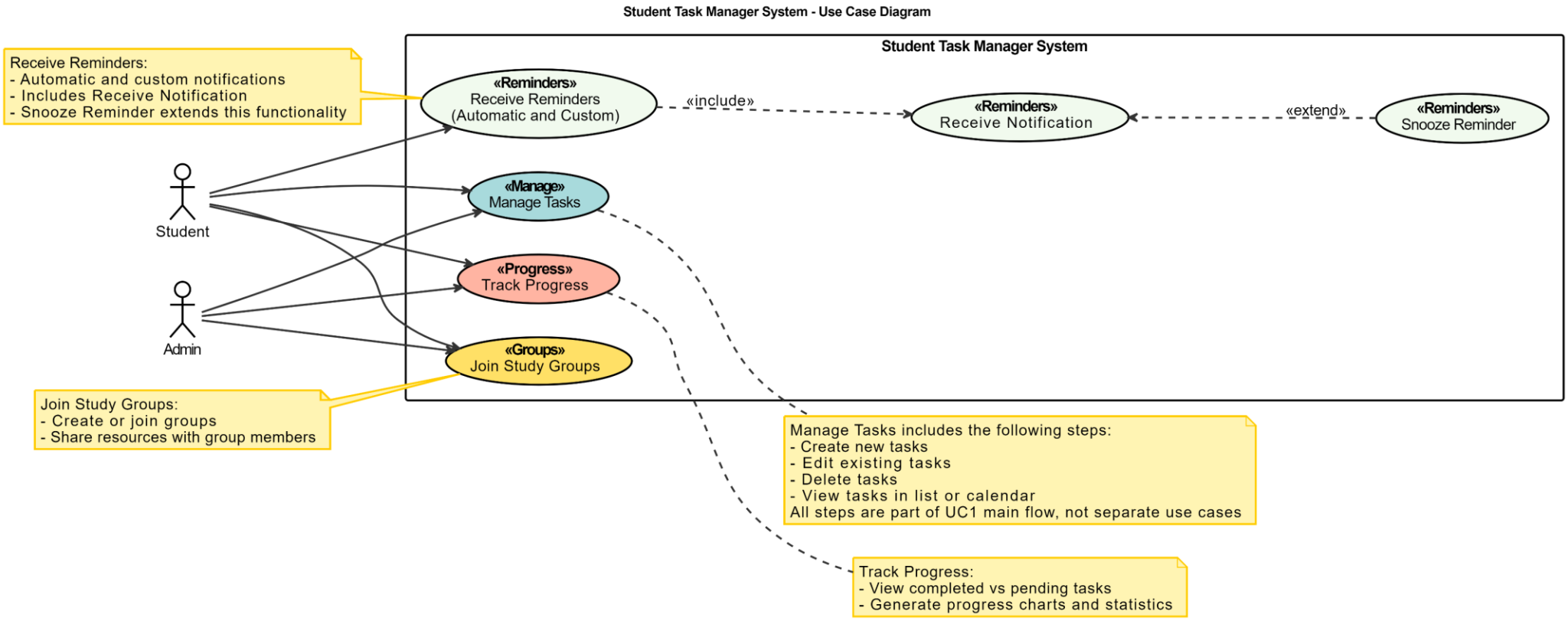
- Student
- Admin (for future scalability)

Main Use Cases:

- Manage Tasks (Create, Edit, Delete, View)
- Receive Reminders (Automatic and Custom)
- Join Study Groups
- Track Progress

Relationships:

- <<include>>: Receive Notification is included in Set Task Reminder
- <<extend>>: Snooze Reminder extends Receive Notification


Student Task Manager System - Use Case Diagram
Student Task Manager System
Receive Reminders:
- Automatic and custom notifications
- Includes Receive Notification
- Snooze Reminder extends this functionality
«Reminders»
Receive Reminders
(Automatic and Custom)
«include»
«Reminders»
Receive Notification
«extend»
«Reminders»
Snooze Reminder
Student
Admin
«Manage»
Manage Tasks
«Progress»
Track Progress
«Groups»
Join Study Groups
Join Study Groups:
- Create or join groups
- Share resources with group members
Manage Tasks includes the following steps:
- Create new tasks
- Edit existing tasks
- Delete tasks
- View tasks in list or calendar
All steps are part of UC1 main flow, not separate use cases
Track Progress:
- View completed vs pending tasks
- Generate progress charts and statistics

4.2 Class Diagram

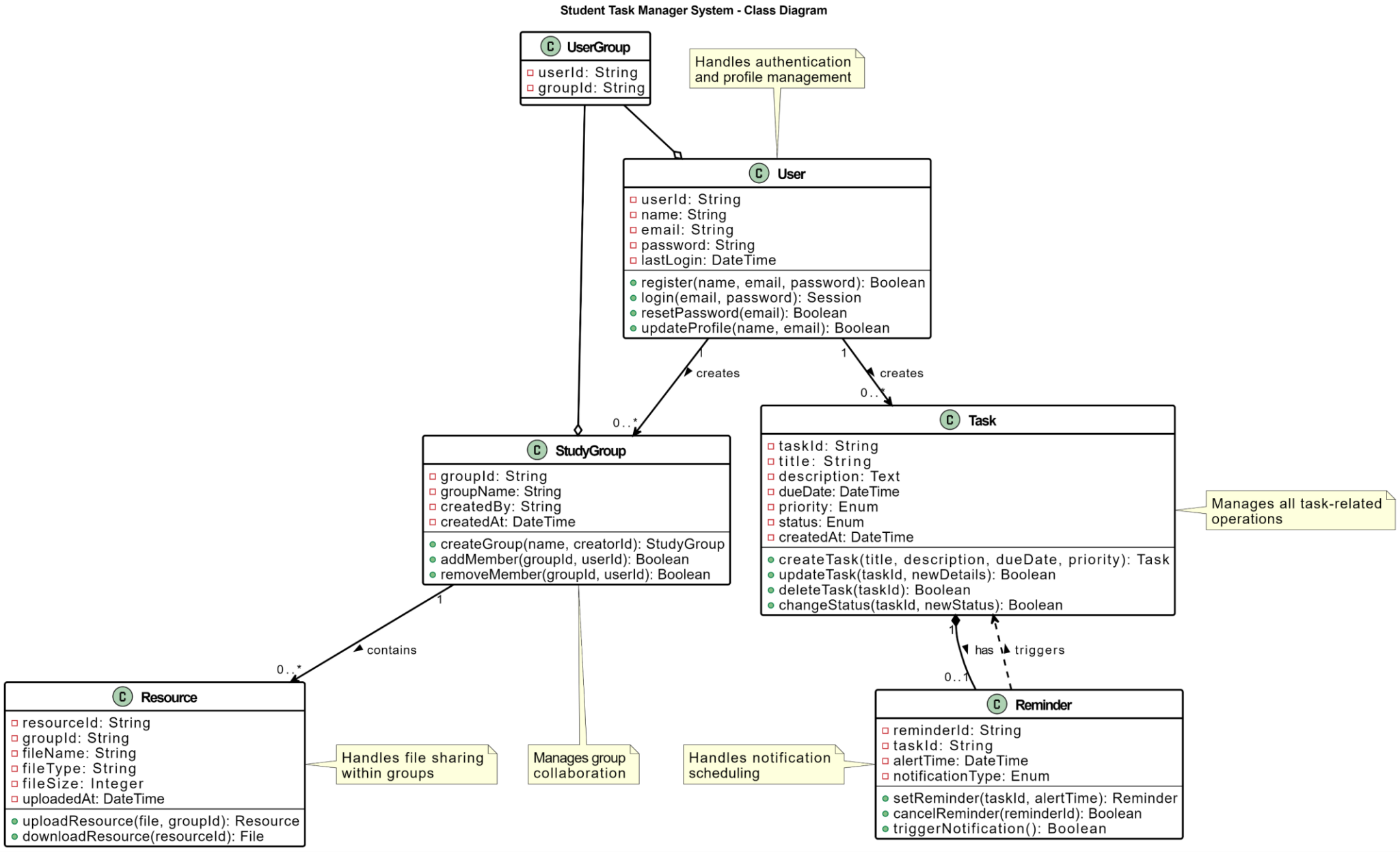
The Class Diagram shows the static structure of STMS, including key classes, attributes, methods, and relationships:

Core Classes:

- **User**: Handles registration, login, and profile management
- **Task**: Manages creation, editing, deletion, and categorization
- **Reminder**: Handles scheduling and sending notifications
- **StudyGroup**: Manages collaborative groups
- **SharedResource**: Handles file sharing within groups

Key Relationships:

- One User → Many Tasks (1-to-Many)
- Task ← → Reminder (1-to-1, composition)
- User ← → StudyGroup (Many-to-Many via GroupMember)
- StudyGroup → SharedResource (1-to-Many, aggregation)


Student Task Manager System - Class Diagram
UserGroup
userId: String
groupId: String
Handles authentication and profile management
User
userId: String
name: String
email: String
password: String
lastLogin: DateTime
register(name, email, password): Boolean
login(email, password): Session
resetPassword(email): Boolean
updateProfile(name, email): Boolean
1
creates
0..*
1
creates
0..*
StudyGroup
groupId: String
groupName: String
createdBy: String
createdAt: DateTime
createGroup(name, creatorId): StudyGroup
addMember(groupId, userId): Boolean
removeMember(groupId, userId): Boolean
Task
taskId: String
title: String
description: Text
dueDate: DateTime
priority: Enum
status: Enum
createdAt: DateTime
createTask(title, description, dueDate, priority): Task
updateTask(taskId, newDetails): Boolean
deleteTask(taskId): Boolean
changeStatus(taskId, newStatus): Boolean
Manages all task-related operations
1
contains
0..*
1
has
triggers
0..1
Resource
resourceId: String
groupId: String
fileName: String
fileType: String
fileSize: Integer
uploadedAt: DateTime
uploadResource(file, groupId): Resource
downloadResource(resourceId): File
Handles file sharing within groups
Manages group collaboration
Handles notification scheduling
Reminder
reminderId: String
taskId: String
alertTime: DateTime
notificationType: Enum
setReminder(taskId, alertTime): Reminder
cancelReminder(reminderId): Boolean
triggerNotification(): Boolean

4.3 Sequence Diagrams

Scenario 1: Task Creation with Reminder

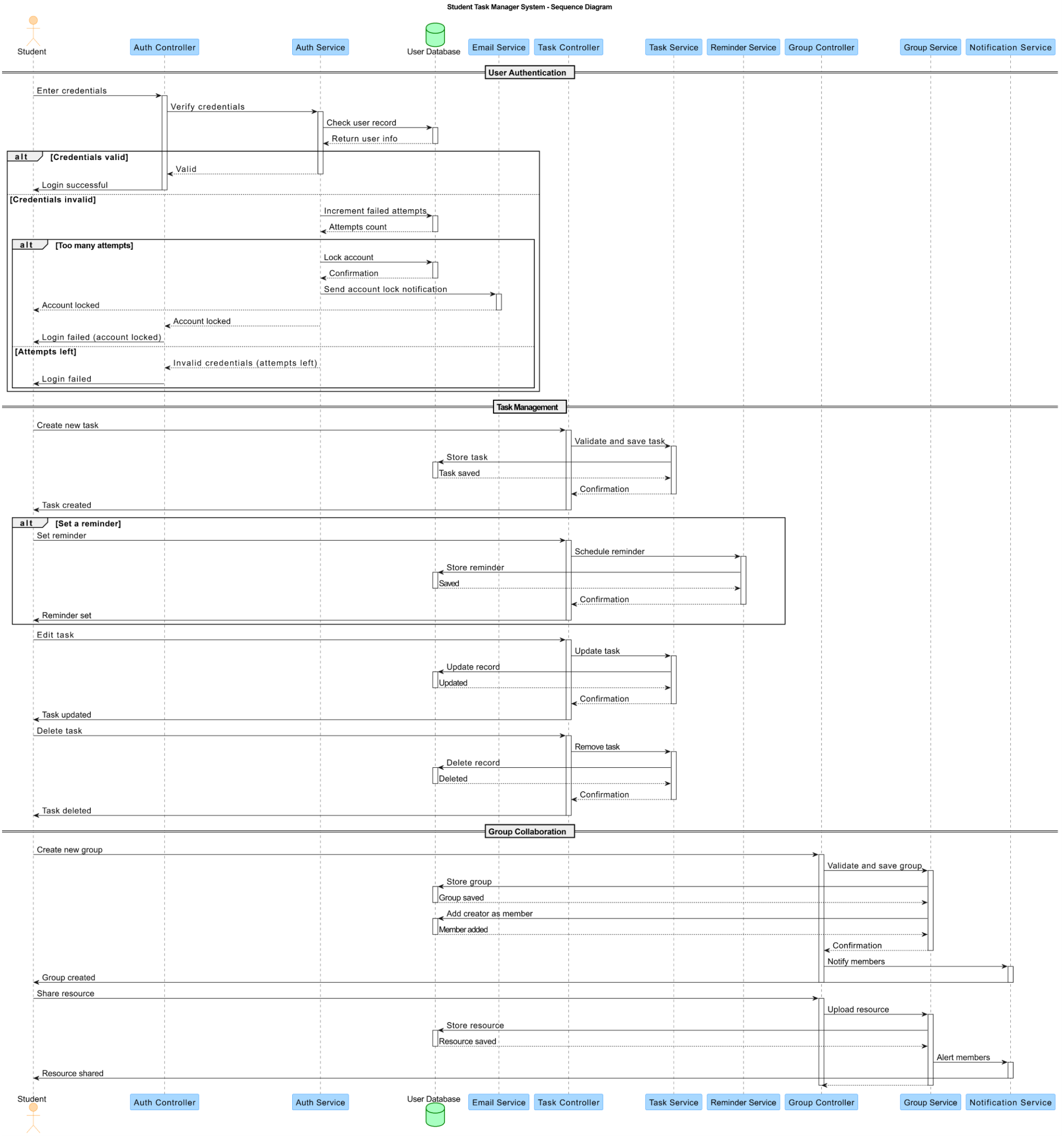
- **Actors/Objects:** Student, TaskController, TaskService, ReminderService, Database
- **Flow:**
    1. Student initiates task creation.
    2. System validates input and saves the task.
    3. If a reminder is set, the system schedules it in ReminderService.

Scenario 2: Study Group Collaboration

- **Actors/Objects:** Student, GroupController, GroupService, NotificationService, Database
- **Flow:**
    1. Student creates or joins a study group.
    2. System notifies group members when resources are shared.

Scenario 3: User Login with Failed Attempt Handling

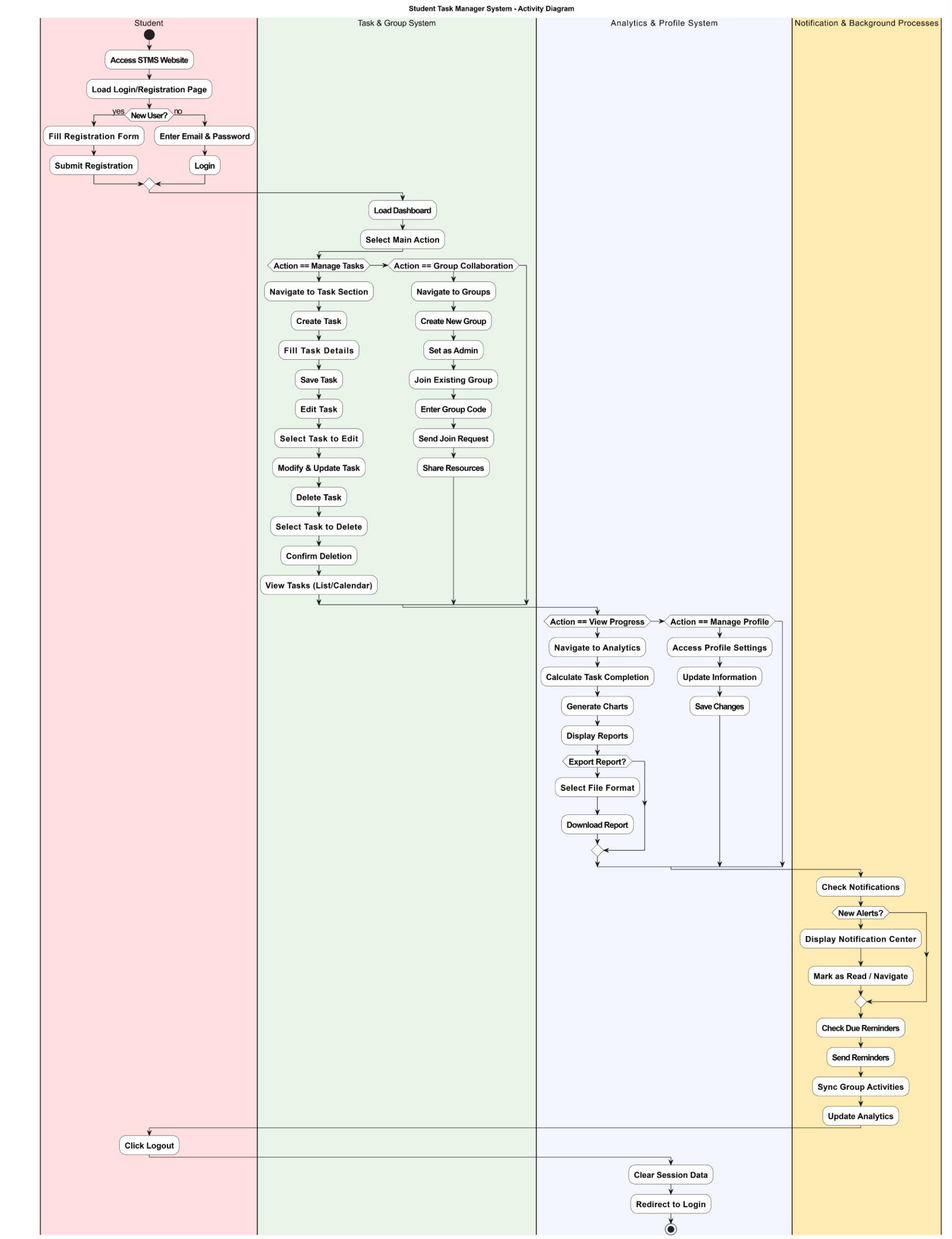
- **Actors/Objects:** Student, AuthController, AuthService, Database, EmailService
- **Flow:**
    1. Student attempts login.
    2. System validates credentials.
    3. On 3 failed attempts, account is locked and notification is sent.


Student Task Manager System - Sequence Diagram
Student
Auth Controller
Auth Service
User Database
Email Service
Task Controller
Task Service
Reminder Service
Group Controller
Group Service
Notification Service
User Authentication
Enter credentials
Verify credentials
Check user record
Return user info
alt [Credentials valid]
Valid
Login successful
[Credentials invalid]
Increment failed attempts
Attempts count
alt [Too many attempts]
Lock account
Confirmation
Send account lock notification
Account locked
Account locked
Login failed (account locked)
[Attempts left]
Invalid credentials (attempts left)
Login failed
Task Management
Create new task
Validate and save task
Store task
Task saved
Confirmation
Task created
alt [Set a reminder]
Set reminder
Schedule reminder
Store reminder
Saved
Confirmation
Reminder set
Edit task
Update task
Update record
Updated
Confirmation
Task updated
Delete task
Remove task
Delete record
Deleted
Confirmation
Task deleted
Group Collaboration
Create new group
Validate and save group
Store group
Group saved
Add creator as member
Member added
Confirmation
Notify members
Group created
Share resource
Upload resource
Store resource
Resource saved
Alert members
Resource shared

4.4 Activity Diagram

Activity diagrams describe workflows for core operations:

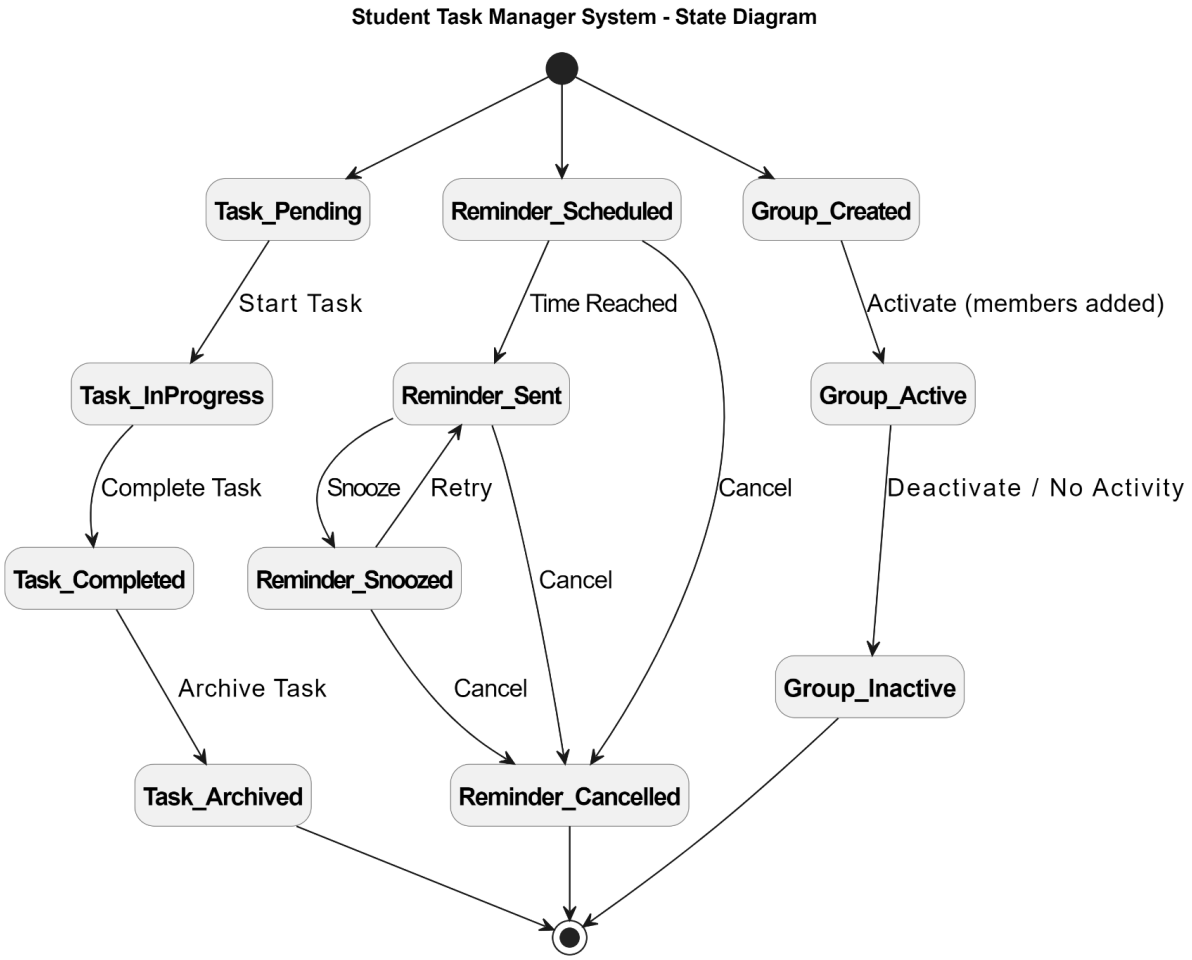
- **Task Management Workflow:** Add, edit, delete tasks; update status; trigger reminders
- **Study Group Workflow:** Create/join groups, share resources, receive notifications
- **Progress Tracking Workflow:** Calculate task completion, generate charts, export reports



4.5 State Diagram

State diagrams show the life cycle of main entities:

- **Task States:** Pending → In Progress → Completed → Archived
- **Reminder States:** Scheduled → Sent → Snoozed → Cancelled
- **Study Group States:** Created → Active → Inactive



5. Change Management Process

To ensure all changes to the Student Task Manager System (STMS) are properly evaluated, approved, and documented to maintain system integrity and quality.

1. Change Request Initiation:

- **Who can request changes:**
    - Students (end users)
    - Project team members
    - Instructors or supervisors (for academic purposes)

2. Logging Changes:

- All change requests must be submitted via a standardized **Change Request Form**.
- Each request includes:
    - Requester name and role
    - Date of submission
    - Description of the change
    - Reason for the change
    - Priority (Low, Medium, High)

3. Review and Approval:

- The project team reviews all requests in weekly meetings.
- Evaluation criteria:
    - Impact on system functionality
    - Alignment with project scope
    - Required resources and time
- Approval hierarchy:
    - Minor changes: Approved by Project Lead
    - Major changes: Requires Team and Supervisor approval

4. Implementation and Versioning:

- Approved changes are implemented following standard development procedures.
- Each update is logged with a version number (e.g., v1.0, v1.1).
- Updated documentation, UML diagrams, and user guides are maintained in the project repository.

5. Tracking:

- All change requests, approvals, and implementations are stored in a Change Log table for traceability.

Appendices

1. Glossary (Terms and Definitions)

Term	Definition
Task	A task created by the student in the system, which can be new, in progress, completed, or archived.
Reminder	A notification associated with a task, set to alert the student at a specific time.
Group	A study group consisting of multiple students for collaborative work or resource sharing.
Analytics	Statistics and reports on the student's progress, including completed tasks, performance, and achievement rates.
Student	The primary user of the system, responsible for managing tasks, groups, and tracking progress.
Auth Controller	System component responsible for handling user login and authentication.
Task Controller	System component responsible for task management (create, edit, delete).
Group Controller	System component responsible for managing study groups.
Notification Service	System service that sends notifications and alerts to students.
Database (DB)	The system database storing all information about students, tasks, groups, and reminders.