



# **INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT, AKURDI, PUNE**

**“ProjexFlow”**

**PG-DAC August 2025**

Submitted By:

Group No: 66

<b>Roll No.</b>	<b>Name of Student</b>
258205	Tanmay Mahajan
258112	Abhishek Khedkar

**Mr. Vaibhav Verulkar**  
Project Guide

**Mr. Anil Sharma**  
Centre Coordinator

## **ABSTRACT**

The rapid growth of academic institutions and the increasing complexity of project-based learning have created a need for efficient project management systems. Traditional methods of managing academic projects often involve manual processes, leading to communication gaps, delayed feedback, and difficulty in tracking progress.

ProjexFlow is a comprehensive Smart Academic Project Management System designed to streamline the entire lifecycle of academic project management. Built using a modern microservices architecture with Spring Boot backend and React frontend, ProjexFlow provides a centralized platform for administrators, mentors, and students to collaborate effectively.

The system features role-based access control with three distinct user roles: Admin, Mentor, and Student. Administrators can manage users, configure batches, and control the grouping phase. Mentors can create and assign tasks, monitor group progress, and evaluate submissions. Students can form groups, view assigned tasks, submit their work, and track their project activities.

ProjexFlow implements a microservices architecture consisting of nine independent services: User Management Service (UMS), Group Management Service (GMS), Task Management Service (TMS), Project Management Service (PMS), Mentor Assignment Management Service (MAMS), Notification Service (NMS), Activity Log Service (ALS), along with Eureka Service Registry and API Gateway. Each service operates independently with its own database, ensuring scalability, maintainability, and fault tolerance. This project demonstrates the effective application of modern software engineering principles, including microservices architecture, RESTful API design, responsive web development, and secure authentication mechanisms. ProjexFlow aims to enhance the academic project management experience by providing a robust, scalable, and user-friendly platform.

## **ACKNOWLEDGEMENT**

We would like to express our sincere gratitude to all those who have contributed to the successful completion of this project.

First and foremost, we extend our heartfelt thanks to our project guide, Mr. Vaibhav Velurkar, for their invaluable guidance, continuous support, and encouragement throughout the development of this project. Their expertise and insights have been instrumental in shaping this work.

I sincerely thank our respected Centre Coordinator, Mr. Anil Sharma, for allowing us to use the available facilities. I would also like to thank the other faculty members at this occasion. Last but not least, I would like to thank my friends and family for the support and encouragement they have given me during our work.

Tanmay Mahajan (250841220187)

Abhishek Khedkar (250841220003)

**Table of Contents**

<b>Sr. No.</b>	<b>Description</b>	<b>Page No.</b>
1	Introduction	1
2	SRS	7
3	Diagrams	16
3.1	ER Diagram	16
3.2	Use Case Diagram	17
3.3	Data Flow Diagram	18
3.4	Activity Diagram	19
3.5	Class Diagram	22
3.6	Sequence Diagram	24
4	Database Design	27
5	Snapshots	40
6	Conclusion	46
7	References	47

## 1. INTRODUCTION

**ProjexFlow** is a comprehensive Smart Academic Project Management System designed to address these challenges by providing a unified, digital platform for managing the entire lifecycle of academic projects. The system brings together administrators, mentors, and students on a single platform, enabling seamless collaboration, efficient task management, and real-time progress tracking.

Built using cutting-edge technologies including Spring Boot microservices architecture for the backend and React for the frontend, ProjexFlow demonstrates the practical application of modern software engineering principles. The system implements role-based access control, ensuring that each user type (Admin, Mentor, Student) has access to appropriate functionalities tailored to their responsibilities.

The platform facilitates group formation through a structured request-and-approval workflow, allows mentors to create and assign tasks with specific deadlines, enables students to submit their work digitally, and provides comprehensive activity logging for audit and tracking purposes. By centralizing these functionalities, ProjexFlow eliminates the need for multiple disparate tools and reduces the administrative overhead associated with academic project management

## 1.1 Problem Statement :

"To design and develop a comprehensive, scalable, and secure Smart Academic Project Management System that facilitates efficient collaboration between administrators, mentors, and students, streamlines group formation and task management processes, provides real-time progress tracking, and eliminates the inefficiencies associated with traditional manual project management methods in academic institutions."

The system must address the following specific problems:

1. **Fragmented Communication:** Lack of a centralized platform for project-related communication
2. **Inefficient Group Formation:** Time-consuming manual processes for forming student groups
3. **Task Management Complexity:** Difficulty in creating, assigning, and tracking tasks across multiple groups
4. **Submission Tracking:** Absence of a systematic approach to manage and evaluate student submissions
5. **Limited Visibility:** Insufficient transparency in project progress for all stakeholders
6. **Scalability Issues:** Inability of existing solutions to handle growing numbers of users and projects
7. **Security Concerns:** Need for secure authentication and authorization mechanisms
8. **Data Persistence:** Requirement for reliable data storage and retrieval across distributed services

## 1.2 Objectives :

The primary objectives of the ProjexFlow project are:

### Primary Objectives :

#### 1. Develop a Microservices-Based Architecture

- Design and implement a scalable microservices architecture using Spring Boot
- Ensure loose coupling and high cohesion between services
- Implement service discovery using Netflix Eureka
- Create an API Gateway for centralized request routing

#### 2. Implement Role-Based Access Control

- Define three distinct user roles: Admin, Mentor, and Student
- Implement JWT-based authentication and authorization
- Ensure secure access to role-specific functionalities

#### 3. Create User Management System

- Develop user registration and authentication mechanisms
- Implement profile management with photo upload capabilities
- Provide password management and account security features

#### 4. Build Group Management Functionality

- Enable students to send and receive group formation requests
- Implement approval/rejection workflow for group requests
- Provide batch-wise group organization
- Allow administrators to control grouping phase activation

#### 5. Develop Task Management System

- Enable mentors to create tasks with detailed descriptions and deadlines
- Implement task assignment to specific groups or entire batches
- Provide task submission capabilities for students
- Track task status and completion

#### 6. Implement Project Tracking

- Create project management capabilities for student groups
- Track project milestones and deliverables

- Provide project status visibility to all stakeholders

## 7. **Build Responsive Frontend**

- Develop a modern, responsive React-based user interface
- Ensure cross-device compatibility
- Implement intuitive navigation and user experience

### **Secondary Objectives :**

1. **Activity Logging:** Implement comprehensive activity logging for audit trails
2. **Notification System:** Develop notification mechanisms for important events
3. **Data Analytics:** Provide basic analytics and reporting capabilities
4. **Performance Optimization:** Ensure fast response times and efficient resource utilization
5. **Documentation:** Create comprehensive technical and user documentation

## 1.3 Roles and Access Control :

### For Administrators:

- User management (add mentors and students only)
- Batch configuration and management
- Grouping phase control (enable/disable group formation)
- System-wide monitoring and oversight
- Access to all groups and projects

### For Mentors:

- View assigned groups and their members
- Create tasks with titles, descriptions, instructions, and due dates
- Assign tasks to specific groups or all groups in a batch
- Review and evaluate student submissions
- Track group progress and activity
- Provide feedback on submissions

### For Students:

- Form groups through request/approval workflow
- View team members and group details
- Access assigned tasks
- Submit task solutions
- View project information (one project per group)
- Track personal activity logs
- View profile information

## 1.4 Technical Features:

- Microservices architecture with 9 independent services
- JWT-based authentication and authorization
- RESTful API design
- MySQL database for most microservices (UMS, GMS, TMS, MAMS, NMS, ALS)
- MongoDB database for Project Management Service (PMS)
- Service discovery and load balancing
- API Gateway for request routing
- Responsive web interface

## **2. SOFTWARE REQUIREMENT SPECIFICATION**

### **2.1 Functional Requirements**

#### **User Management**

##### **FR 1.1 : User Registration**

- The system shall allow administrators to register new users (mentors and students)
- Each user shall have a unique email address
- Users shall be assigned to specific batches
- User information shall include name, email, role, and batch ID

##### **FR 1.2 : User Authentication**

- The system shall provide secure login functionality using email and password
- The system shall generate JWT tokens upon successful authentication
- Tokens shall expire after a configured time period (default: 1 hour)
- The system shall validate tokens for all protected API endpoints

##### **FR 1.3 : Profile Management**

- Users shall be able to view their profile information
- Profile information shall include name, email, role, and batch ID

#### **Group Management**

##### **FR 2.1 : Group Formation**

- Students shall be able to send group requests to other students
- Students shall be able to view incoming group requests
- Students shall be able to accept or reject group requests
- The system shall automatically create groups when requests are accepted

##### **FR 2.2 : Grouping Phase Control**

- Administrators shall be able to enable/disable the grouping phase for specific batches
- Students shall only be able to send group requests when grouping is enabled
- The system shall prevent group modifications when grouping is disabled

**FR 2.3 : Group Viewing**

- Students shall be able to view their group members
- Mentors shall be able to view all groups assigned to them
- Administrators shall be able to view all groups in the system
- Group details shall include member names, emails, and roles

**Task Management****FR 3.1 : Task Creation**

- Mentors shall be able to create tasks with title, description, instructions, and due date
- Tasks shall be associated with specific batches
- Mentors shall specify task details including objectives and deliverables

**FR 3.2 : Task Assignment**

- Mentors shall be able to assign tasks to all groups in a batch
- Mentors shall be able to assign tasks to specific selected groups
- The system shall notify groups when tasks are assigned

**FR 3.3 : Task Submission**

- Students shall be able to view tasks assigned to their group
- Students shall be able to submit solutions for assigned tasks
- Submissions shall include submission text and optional file attachments
- The system shall record submission timestamps

**FR 3.4 : Submission Evaluation**

- Mentors shall be able to view all submissions for tasks they created
- Mentors shall be able to provide feedback on submissions
- Mentors shall be able to grade submissions
- Students shall be able to view feedback on their submissions

**Project Management****FR 4.1 : Project Tracking**

- Each group shall have one associated project
- The system shall track project milestones and progress

- Students shall be able to view their group's project details
- Mentors shall be able to monitor project progress
- The system shall enforce one-to-one relationship between groups and projects

### **Activity Logging**

#### **FR 5.1 : Activity Tracking**

- The system shall log all significant user activities
- Activity logs shall include timestamps and user information
- Students shall be able to view their own activity history
- Administrators shall have access to system-wide activity logs

### **Notification System**

#### **FR 6.1 : Notifications**

- The system shall generate notifications for important events
- Users shall receive notifications for group requests
- Users shall receive notifications for task assignments
- Users shall receive notifications for submission feedback

### **Administrative Functions**

#### **FR 7.1 : User Management**

- Administrators shall be able to add new mentors and students
- Administrators shall be able to view all users in the system
- Administrators shall be able to assign users to batches

#### **FR 7.2 : Batch Management**

- Administrators shall be able to create and manage batches
- Administrators shall be able to view all batches in the system
- Each batch shall have a unique identifier

#### **FR 7.3 : System Monitoring**

- Administrators shall have access to system-wide statistics
- Administrators shall be able to view all groups and projects

- Administrators shall be able to monitor system health

## **2.2 Non-Functional Requirements**

### **Performance Requirements**

#### **NFR 1.1 : Response Time**

- API endpoints shall respond within 2 seconds under normal load
- Database queries shall be optimized for fast retrieval
- The system shall handle concurrent requests efficiently

#### **NFR 1.2 : Throughput**

- The system shall support at least 100 concurrent users
- The system shall handle at least 1000 requests per minute
- Database operations shall be optimized for high throughput

#### **NFR 1.3 : Scalability**

- The microservices architecture shall allow horizontal scaling
- Each service shall be independently scalable
- The system shall support growing numbers of users and data

### **Security Requirements**

#### **NFR 2.1 : Authentication**

- All API endpoints (except login/register) shall require authentication
- JWT tokens shall be used for stateless authentication
- Passwords shall be encrypted using industry-standard algorithms

#### **NFR 2.2 : Authorization**

- Role-based access control shall be enforced
- Users shall only access resources appropriate to their role
- API endpoints shall validate user permissions

#### **NFR 2.3 : Data Protection**

- Sensitive data shall be encrypted in transit (HTTPS)
- Database credentials shall be securely stored

- SQL injection and XSS attacks shall be prevented

## **Reliability Requirements**

### **NFR 3.1 : Availability**

- The system shall have 99% uptime
- Services shall implement health checks
- Failed services shall be automatically restarted

### **NFR 3.2 : Fault Tolerance**

- Service failures shall not cascade to other services
- The system shall gracefully handle service unavailability
- Data consistency shall be maintained across services

### **NFR 3.3 : Data Integrity**

- Database transactions shall be ACID-compliant
- Data validation shall be performed at multiple layers
- Backup mechanisms shall be implemented

## **Usability Requirements**

### **NFR 4.1 : User Interface**

- The interface shall be intuitive and easy to navigate
- The system shall provide clear error messages
- Help documentation shall be accessible

### **NFR 4.2 : Responsiveness**

- The frontend shall be responsive across devices
- The UI shall adapt to different screen sizes
- Mobile browsers shall be supported

### **NFR 4.3 : Accessibility**

- The system shall follow web accessibility guidelines
- Keyboard navigation shall be supported
- Screen readers shall be compatible

## **Maintainability Requirements**

### **NFR 5.1 : Code Quality**

- Code shall follow industry best practices
- Services shall be loosely coupled
- Code shall be well-documented

### **NFR 5.2 : Modularity**

- Each microservice shall have a single responsibility
- Services shall communicate via well-defined APIs
- Changes to one service shall not require changes to others

### **NFR 5.3 : Testability**

- Unit tests shall cover critical functionality
- Integration tests shall verify service interactions
- Test coverage shall be maintained above 70%

## **Portability Requirements**

### **NFR 6.1 : Platform Independence**

- The backend shall run on any platform supporting Java 17
- The frontend shall run on modern web browsers
- The system shall not depend on platform-specific features

### **NFR 6.2 : Database Portability**

- Database schema shall be portable across MySQL versions
- JPA shall be used for database abstraction
- Database migrations shall be version-controlled

## **2.3 Hardware Requirements**

### **Development Environment**

#### **Minimum Requirements:**

- Processor: Intel Core i5 or equivalent
- RAM: 8 GB

- Storage: 20 GB free space
- Network: Broadband internet connection

**Recommended Requirements:**

- Processor: Intel Core i7 or equivalent
- RAM: 16 GB
- Storage: 50 GB SSD
- Network: High-speed internet connection

**Production Environment****Server Requirements:**

- Processor: Multi-core server processor (4+ cores)
- RAM: 16 GB minimum (32 GB recommended)
- Storage: 100 GB SSD
- Network: High-bandwidth network interface
- Backup: Redundant storage for backups

**Database Server:**

- Processor: Multi-core processor
- RAM: 8 GB minimum
- Storage: 50 GB SSD with RAID configuration
- Network: Dedicated network interface

**Client Requirements****Desktop/Laptop:**

- Processor: Any modern processor
- RAM: 4 GB minimum
- Display: 1366x768 minimum resolution
- Network: Internet connection

**Mobile Devices:**

- Modern smartphone or tablet
- Internet connectivity

- Modern web browser

## **2.4 Software Requirements**

### **Backend Development**

- Java Development Kit (JDK): Version 17 or higher
- Spring Boot: Version 3.5.x
- Spring Cloud: Version 2025.0.1
- Maven: Version 3.6 or higher
- MySQL: Version 8.0 or higher (for UMS, GMS, TMS, MAMS, NMS, ALS)
- MongoDB: Version 7.0 or higher (for PMS)
- Netflix Eureka: For service discovery
- Spring Cloud Gateway: For API routing

### **Frontend Development**

- Node.js: Version 16 or higher
- npm: Version 8 or higher
- React: Version 18.3.1
- Vite: Version 5.4.11
- React Router: Version 6.28.0
- Axios: Version 1.7.9

### **Development Tools**

- IDE: IntelliJ IDEA / Eclipse / VS Code
- API Testing: Postman / Insomnia
- Version Control: Git
- Database Management: MySQL Workbench / MongoDB Compass

### **Third-Party Services**

- JWT: For authentication tokens

### **Operating System**

- Development: Windows 10/11, macOS, or Linux

**Web Browsers (Client-Side)**

- Google Chrome (latest version)
- Mozilla Firefox (latest version)
- Microsoft Edge (latest version)
- Safari (latest version)

### 3. DIAGRAMS

#### 3.1 Entity Relationship Diagram:

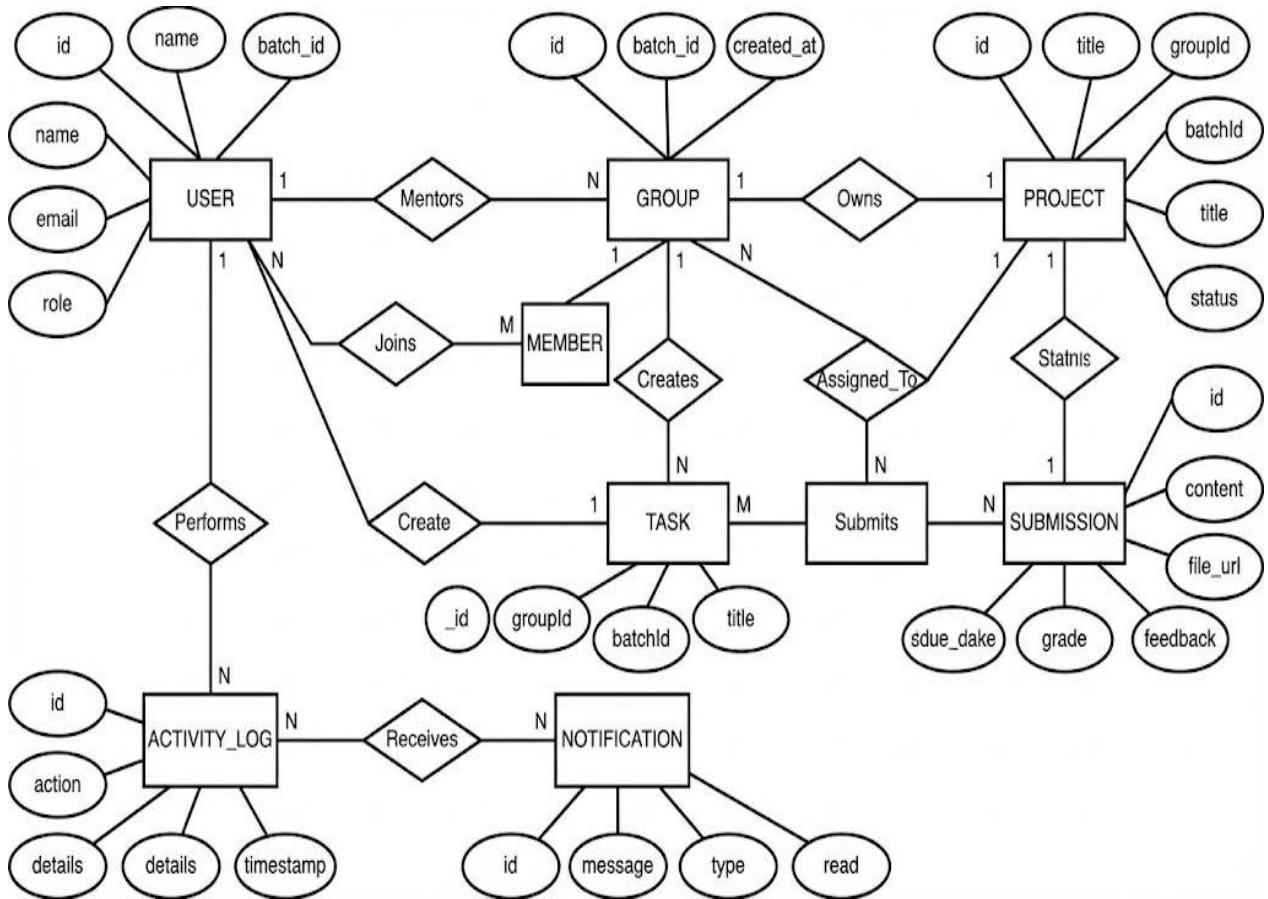


Fig. ER Diagram for ProjexFlow

### 3.2 Use Case Diagram:

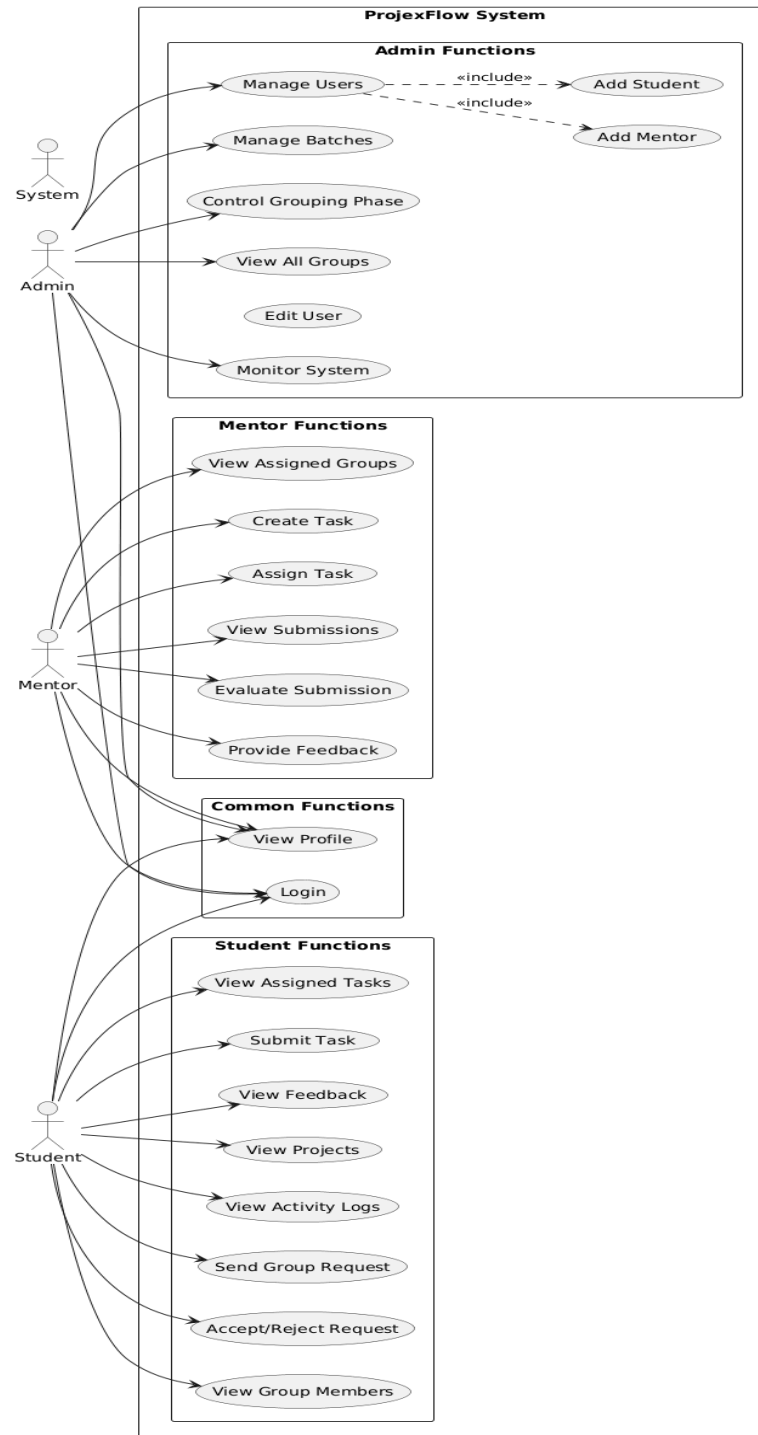
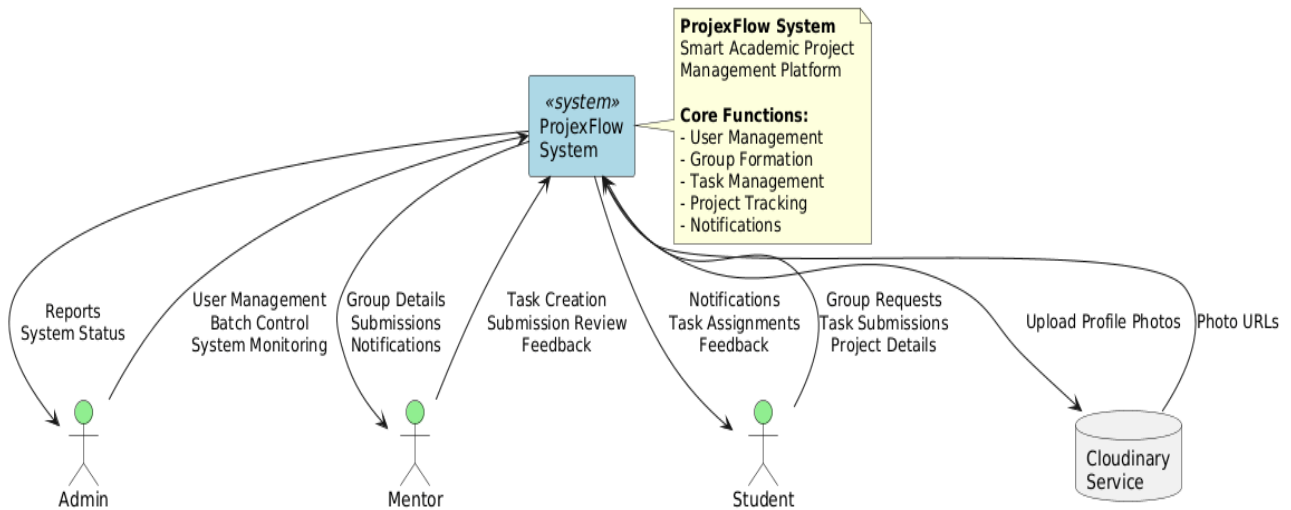


Fig. Use Case Diagram for ProjexFlow

### 3.3 Data Flow Diagram:

#### DFD Level 0:



#### DFD level 1

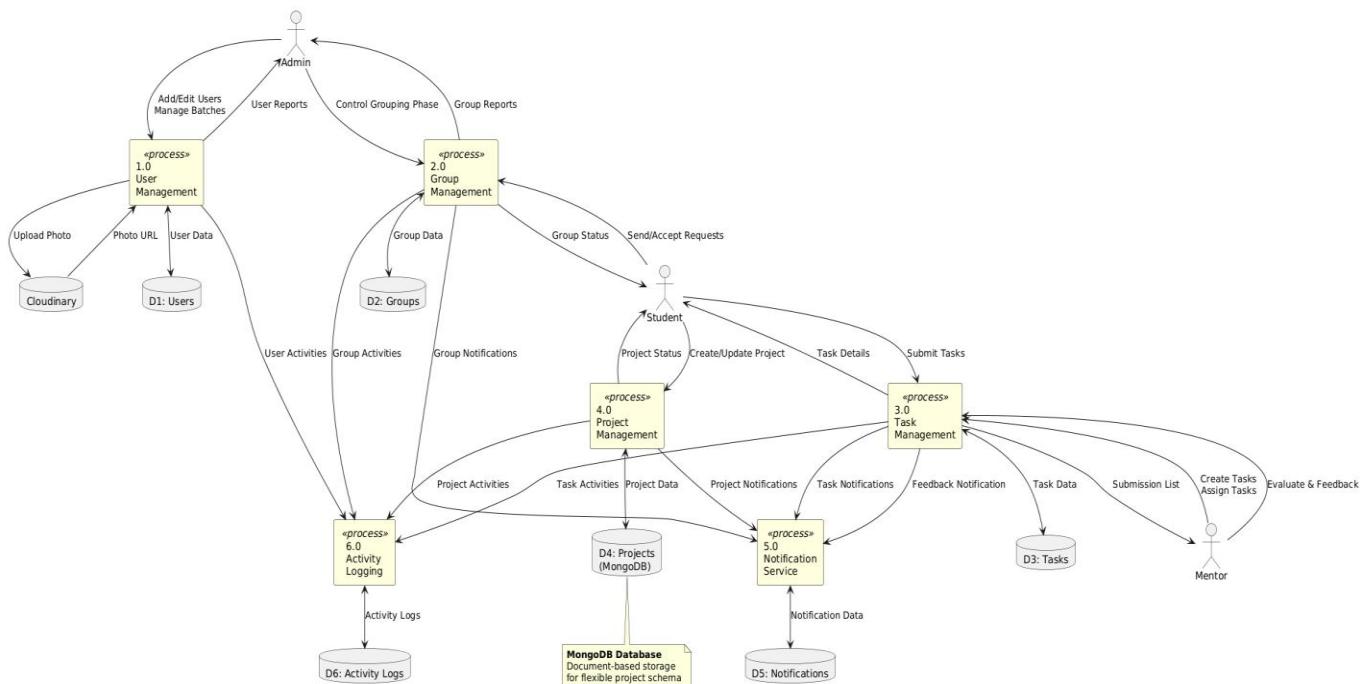
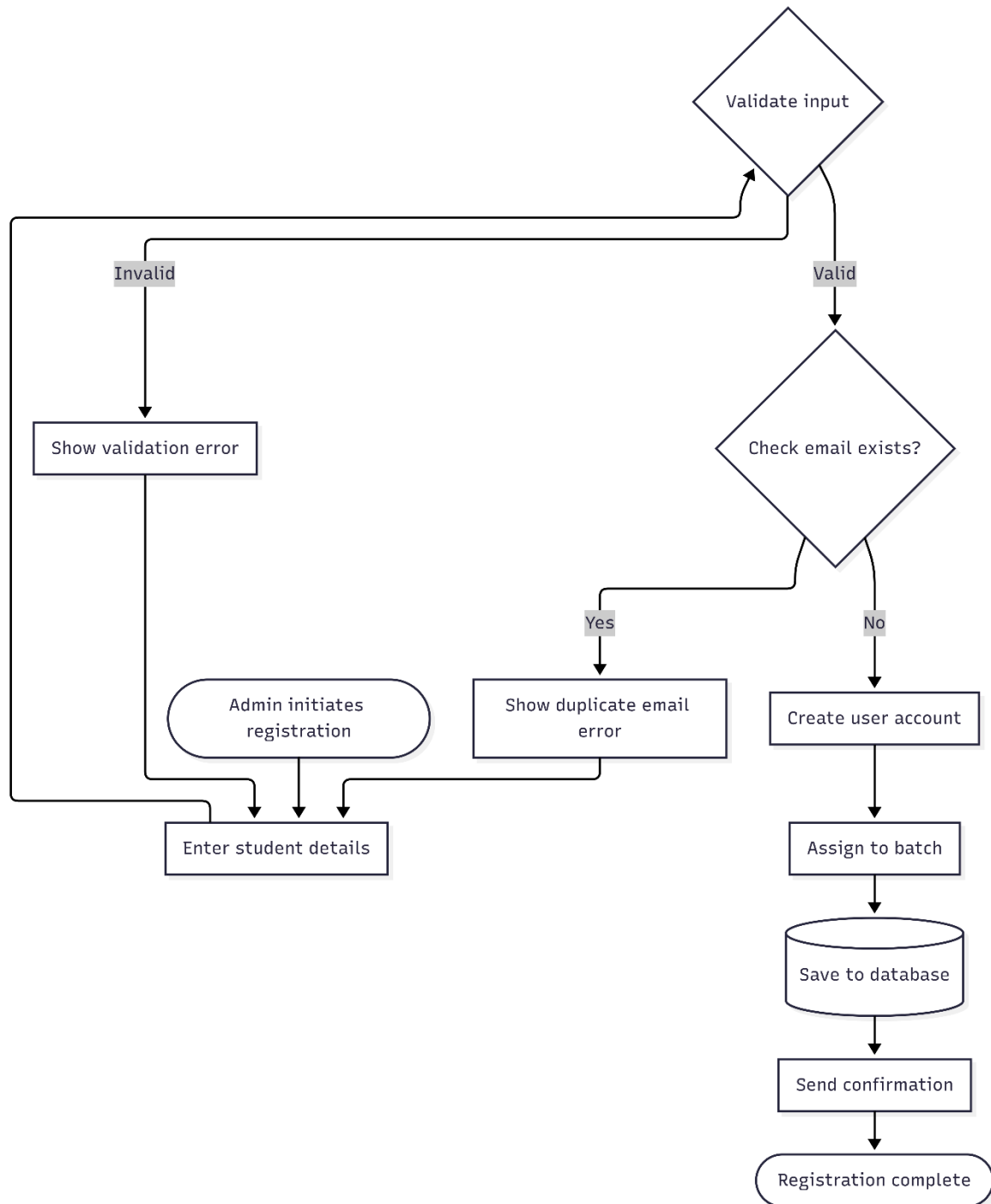


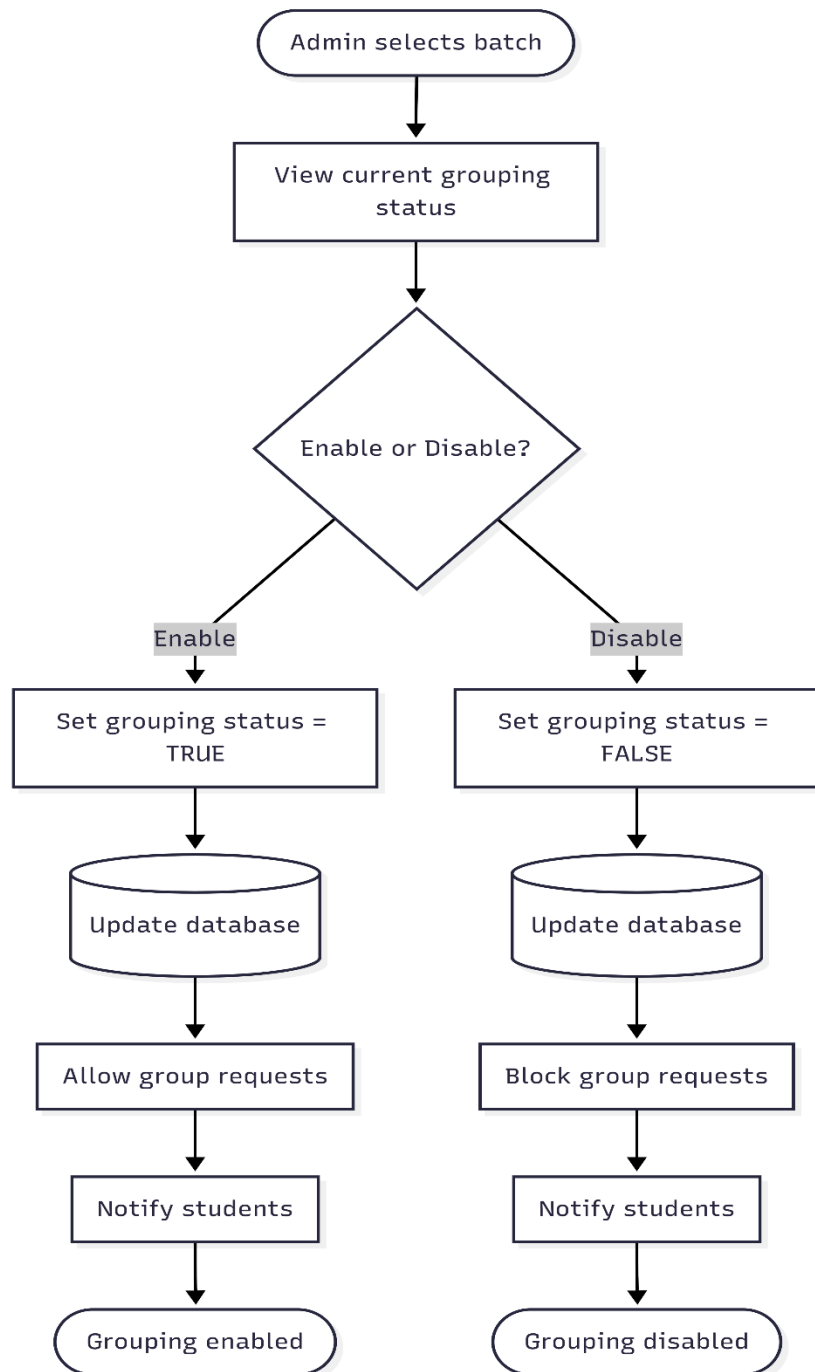
Fig. Data Flow Diagram for ProjexFlow

### 3.4 Activity Diagram :

#### 1. Registration Activity Diagram



## 2. Grouping Phase Activity Diagram:



### 3. Task Management Activity Diagram

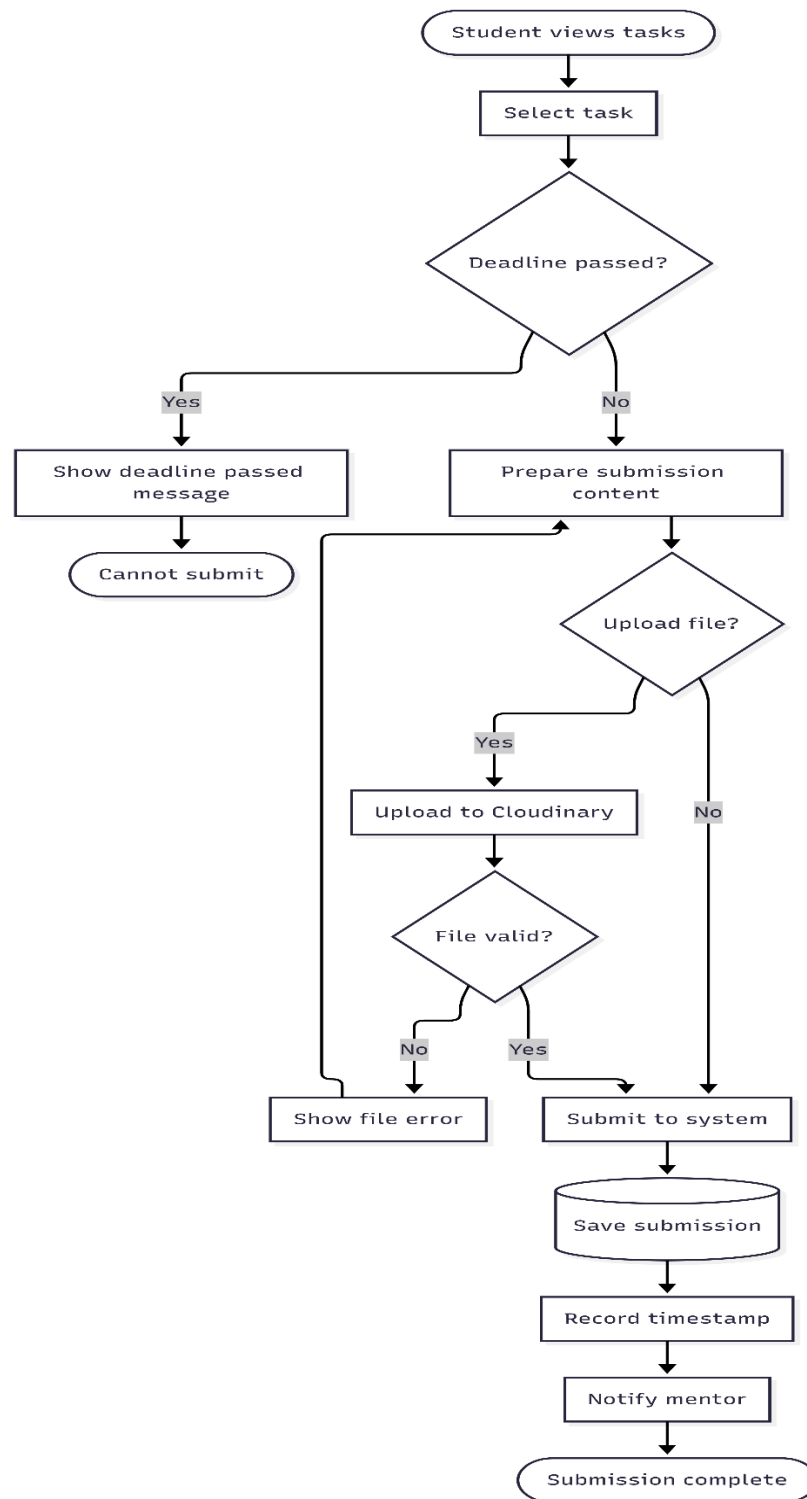
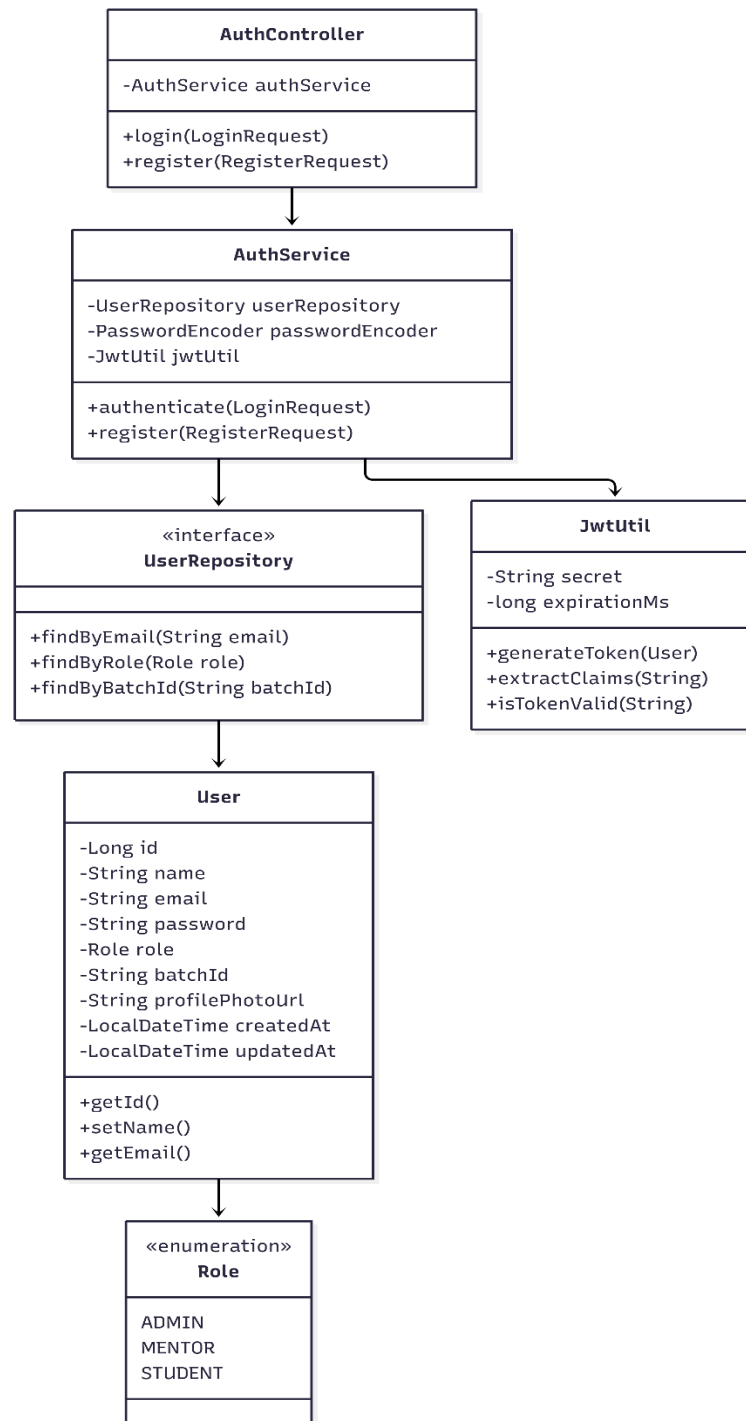


Fig. Activity Diagram for ProjexFlow

### 3.5 Class Diagram:

#### 1. User management class Diagram



## 2.Group Management Class Diagram

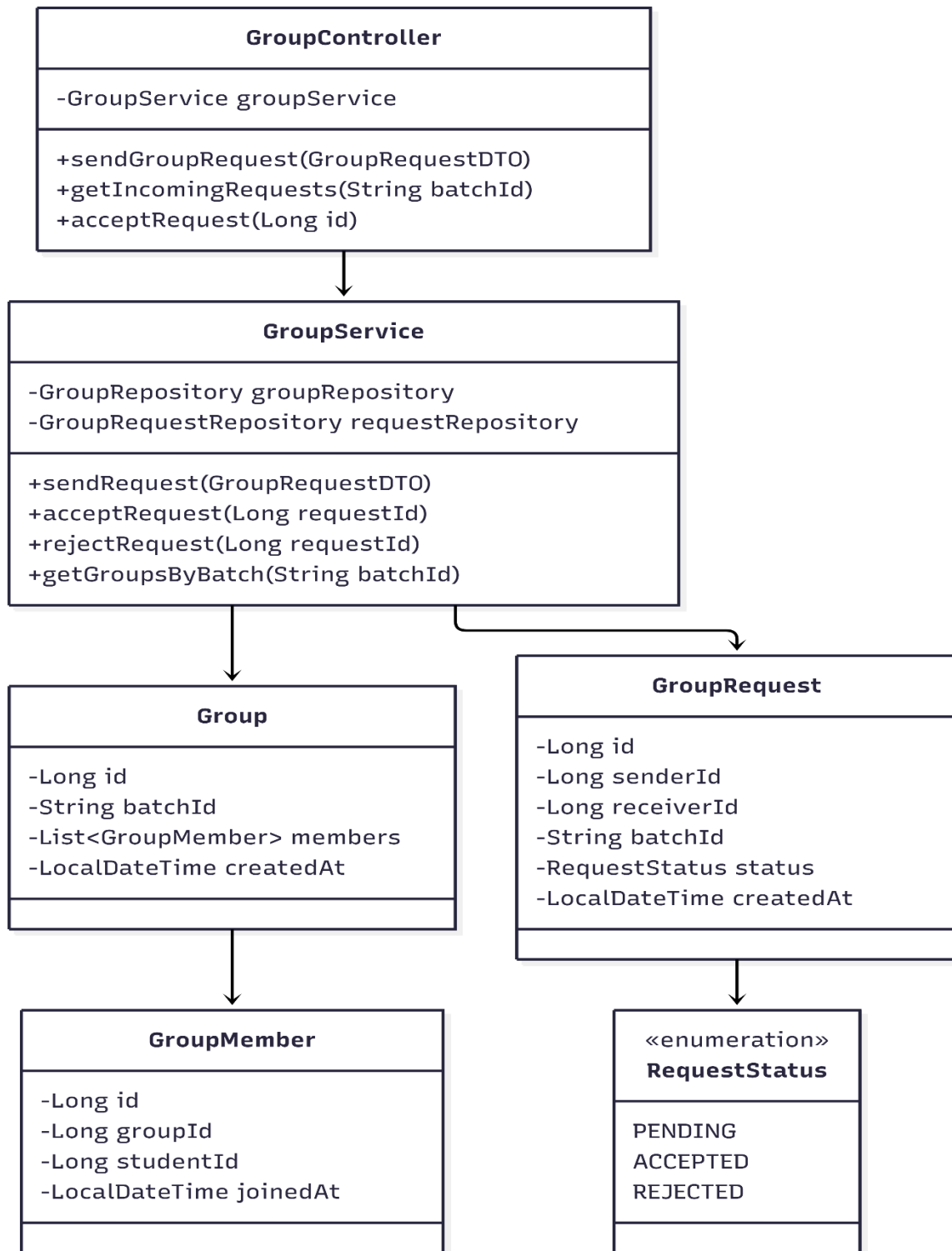
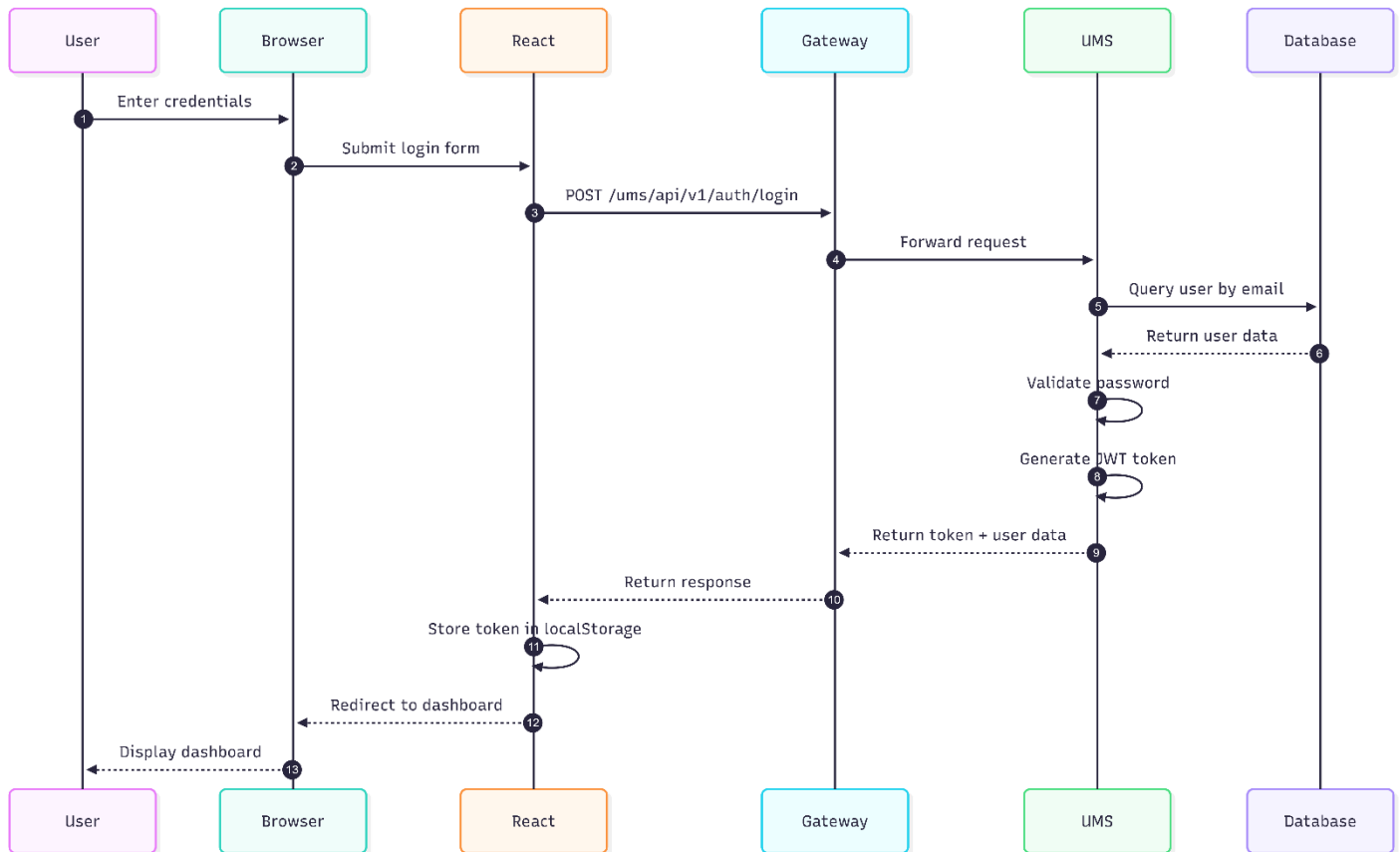


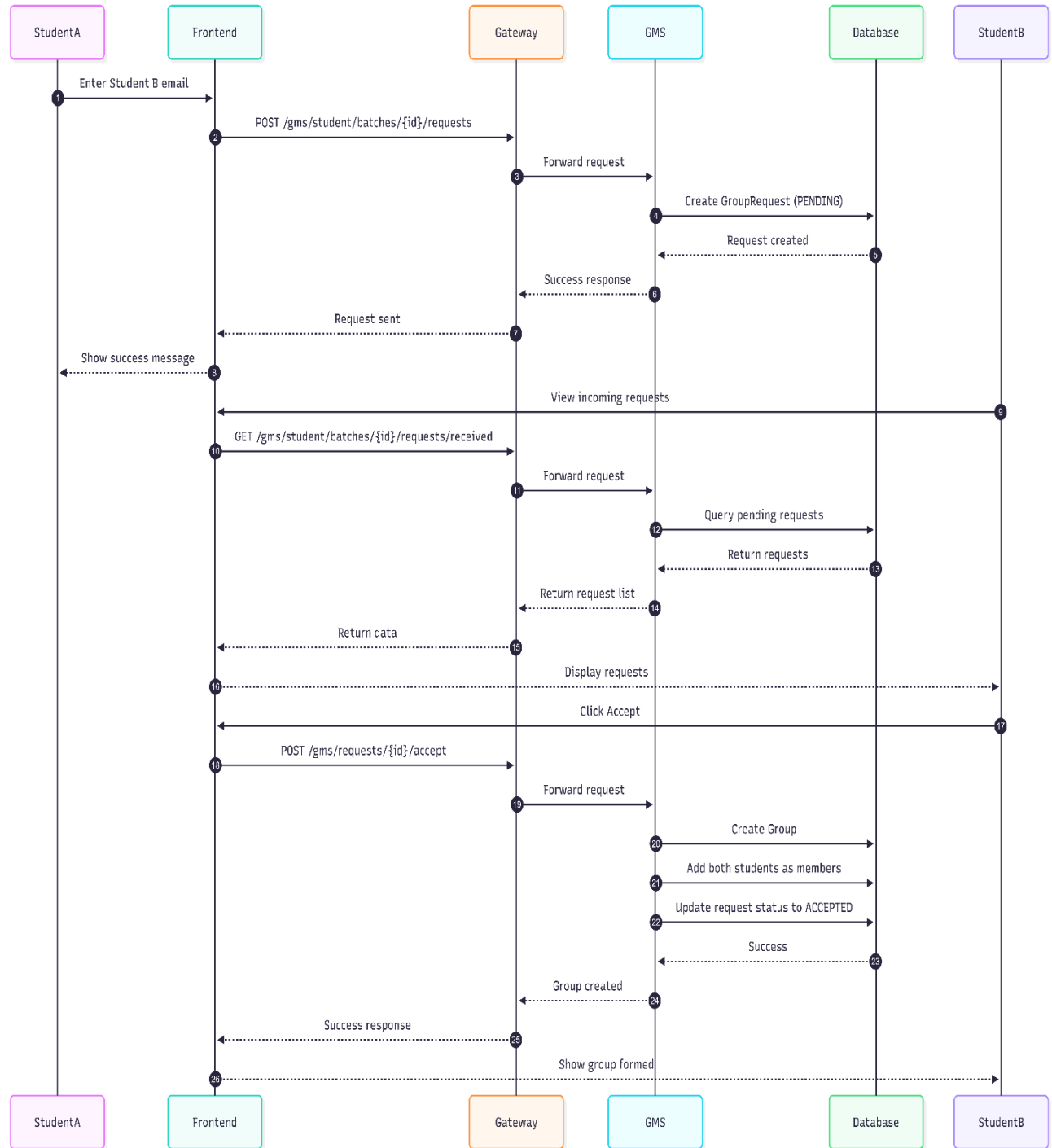
Fig. Class Diagram for ProjexFlow

## 3.6 Sequence Diagram

### 1.Login Sequence Diagram



## 2.Grouping Sequence Diagram



### 3.Task Sequence Diagram

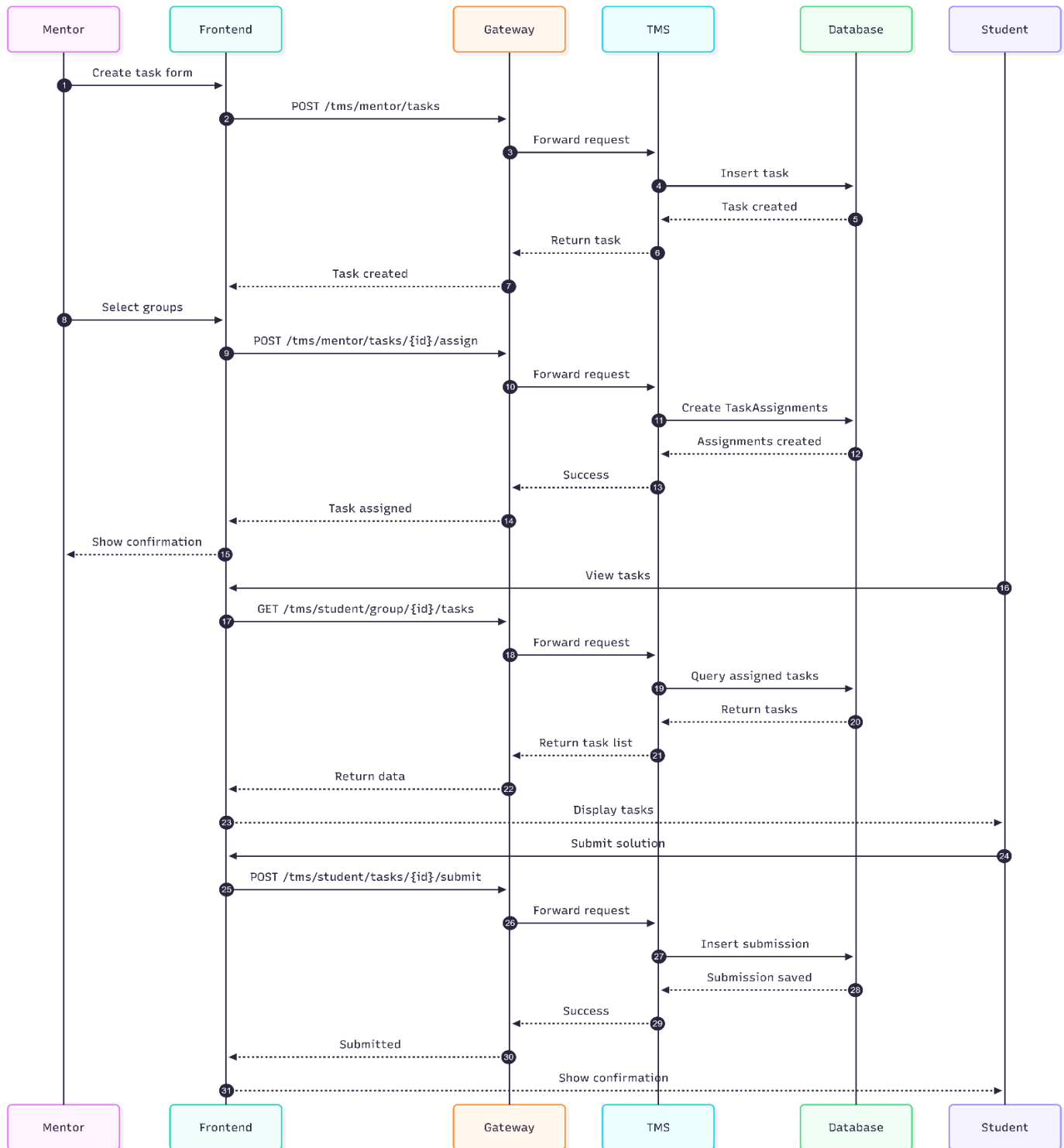
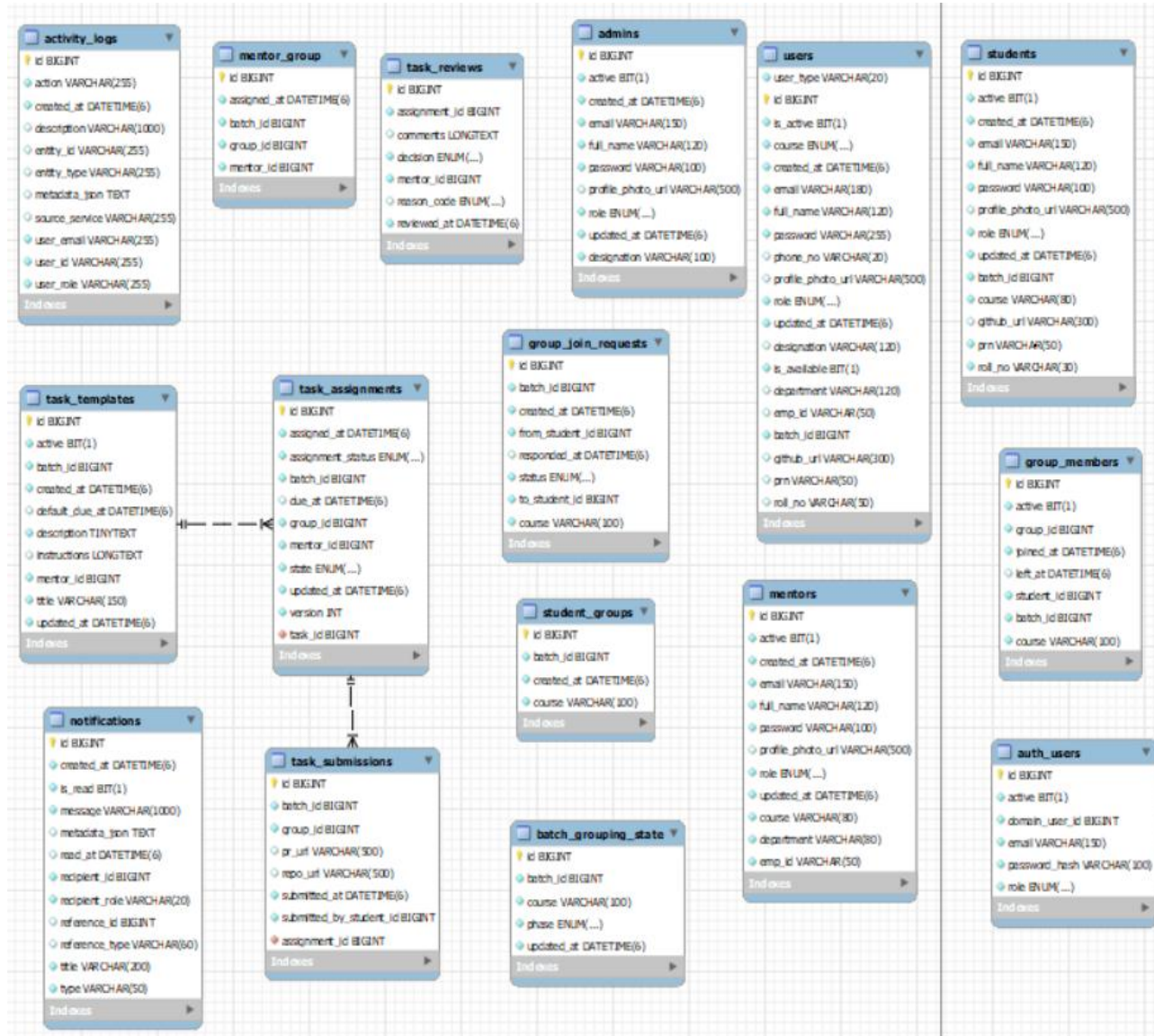


Fig. Sequence Diagram for ProjexFlow

## 4. DATABASE DESIGN

### 4.1 Design:



Ip

projects +

projexflow\_pms > projexflow\_pms > projects >\_ Open MongoDB shell

Documents 1 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#) Reset Analyze Options

[EXPORT SCHEMA](#) This report is based on a sample of 1 document. [Learn more](#)

**endDate**  
date  
Inserted: 2026-04-09 18:30:00

**groupId**  
long  
101

**lockedAfterCreate**  
boolean  
true

**repoUrl**  
string  
<https://github.com/dummy/smart-attendance>

**startDate**  
date  
Inserted: 2026-01-09 18:30:00

## 4.2 Tables:

### User Management Database

#### Table students

```
mysql> desc students;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
active	bit(1)	NO		NULL	
created_at	datetime(6)	NO		NULL	
email	varchar(150)	NO	UNI	NULL	
full_name	varchar(120)	NO		NULL	
password	varchar(100)	NO		NULL	
profile_photo_url	varchar(500)	YES		NULL	
role	enum('ADMIN','MENTOR','STUDENT')	NO		NULL	
updated_at	datetime(6)	NO		NULL	
batch_id	bigint	NO		NULL	
course	varchar(80)	NO		NULL	
github_url	varchar(300)	YES		NULL	
prn	varchar(50)	NO	UNI	NULL	
roll_no	varchar(30)	NO		NULL	

14 rows in set (0.01 sec)

#### Table mentors

```
mysql> desc mentors;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
active	bit(1)	NO		NULL	
created_at	datetime(6)	NO		NULL	
email	varchar(150)	NO	UNI	NULL	
full_name	varchar(120)	NO		NULL	
password	varchar(100)	NO		NULL	
profile_photo_url	varchar(500)	YES		NULL	
role	enum('ADMIN','MENTOR','STUDENT')	NO		NULL	
updated_at	datetime(6)	NO		NULL	
course	varchar(80)	NO		NULL	
department	varchar(80)	NO		NULL	
emp_id	varchar(50)	NO	UNI	NULL	

12 rows in set (0.00 sec)

**Table admin**

```
mysql> desc admin;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
active	bit(1)	NO		NULL	
created_at	datetime(6)	NO		NULL	
email	varchar(150)	NO	UNI	NULL	
full_name	varchar(120)	NO		NULL	
password	varchar(100)	NO		NULL	
profile_photo_url	varchar(500)	YES		NULL	
role	enum('ADMIN','MENTOR','STUDENT')	NO		NULL	
updated_at	datetime(6)	NO		NULL	
designation	varchar(100)	NO		NULL	

10 rows in set (0.00 sec)

**Table auth\_users**

```
mysql> desc auth_users;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
active	bit(1)	NO		NULL	
domain_user_id	bigint	NO		NULL	
email	varchar(150)	NO	UNI	NULL	
password_hash	varchar(100)	NO		NULL	
role	enum('ADMIN','MENTOR','STUDENT')	NO	MUL	NULL	

6 rows in set (0.00 sec)

## Task Management Tables

**Table task\_submissions**

```
mysql> desc task_submissions;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
batch_id	bigint	NO		NULL	
group_id	bigint	NO	MUL	NULL	
pr_url	varchar(500)	YES		NULL	
repo_url	varchar(500)	YES		NULL	
submitted_at	datetime(6)	NO		NULL	
submitted_by_student_id	bigint	NO	MUL	NULL	
assignment_id	bigint	NO	MUL	NULL	

8 rows in set (0.03 sec)

**Table task\_assignments**

```
mysql> desc task_assignments;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
assigned_at	datetime(6)	NO		NULL	
assignment_status	enum('ASSIGNED', 'CANCELLED')	NO		NULL	
batch_id	bigint	NO		NULL	
due_at	datetime(6)	YES		NULL	
group_id	bigint	NO	MUL	NULL	
mentor_id	bigint	NO	MUL	NULL	
state	enum('CHANGES_REQUESTED', 'NOT_SUBMITTED', 'PENDING_REVIEW', 'REJECTED', 'VERIFIED')	NO		NULL	
updated_at	datetime(6)	NO		NULL	
version	int	NO		NULL	
task_id	bigint	NO	MUL	NULL	

11 rows in set (0.00 sec)

**Table task\_reviews**

```
mysql> desc task_reviews;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
assignment_id	bigint	NO	MUL	NULL	
comments	longtext	YES		NULL	
decision	enum('CHANGES_REQUESTED', 'REJECTED', 'VERIFIED')	NO		NULL	
mentor_id	bigint	NO	MUL	NULL	
reason_code	enum('INCOMPLETE', 'INVALID_LINK', 'NEEDS_RECHECK', 'NEEDS_REUPLOAD', 'NOT_AS_DIRECTED')	YES		NULL	
reviewed_at	datetime(6)	NO		NULL	

7 rows in set (0.00 sec)

**Table task\_templates**

```
mysql> desc task_templates;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
active	bit(1)	NO		NULL	
batch_id	bigint	NO	MUL	NULL	
created_at	datetime(6)	NO		NULL	
default_due_at	datetime(6)	YES		NULL	
description	tinytext	NO		NULL	
instructions	longtext	YES		NULL	
mentor_id	bigint	NO	MUL	NULL	
title	varchar(150)	NO		NULL	
updated_at	datetime(6)	NO		NULL	

10 rows in set (0.00 sec)

**Group Management Tables:-****Table student\_groups**

```
mysql> desc student_groups;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
batch_id	bigint	NO	MUL	NULL	
created_at	datetime(6)	NO		NULL	
course	varchar(100)	NO		NULL	

4 rows in set (0.01 sec)

**Table group\_members**

```
mysql> desc group_members;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
active	bit(1)	NO	MUL	NULL	
group_id	bigint	NO	MUL	NULL	
joined_at	datetime(6)	NO		NULL	
left_at	datetime(6)	YES		NULL	
student_id	bigint	NO	MUL	NULL	
batch_id	bigint	NO	MUL	NULL	
course	varchar(100)	NO		NULL	

8 rows in set (0.01 sec)

**Table group\_join\_requests**

```
mysql> desc group_join_requests;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
batch_id	bigint	NO	MUL	NULL	
created_at	datetime(6)	NO		NULL	
from_student_id	bigint	NO		NULL	
responded_at	datetime(6)	YES		NULL	
status	enum('ACCEPTED', 'CANCELLED', 'PENDING', 'REJECTED')	NO		NULL	
to_student_id	bigint	NO		NULL	
course	varchar(100)	NO		NULL	

8 rows in set (0.01 sec)

**Table student\_groups**

```
mysql> desc student_groups;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
action	varchar(255)	NO		NULL	
created_at	datetime(6)	NO	MUL	NULL	
description	varchar(1000)	YES		NULL	
entity_id	varchar(255)	YES		NULL	
entity_type	varchar(255)	YES	MUL	NULL	
metadata_json	text	YES		NULL	
source_service	varchar(255)	YES		NULL	
user_email	varchar(255)	NO		NULL	
user_id	varchar(255)	NO	MUL	NULL	
user_role	varchar(255)	NO		NULL	

11 rows in set (0.04 sec)

Activity Management Database

Table activity\_logs

```
mysql> desc activity_logs;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
action	varchar(255)	NO		NULL	
created_at	datetime(6)	NO	MUL	NULL	
description	varchar(1000)	YES		NULL	
entity_id	varchar(255)	YES		NULL	
entity_type	varchar(255)	YES	MUL	NULL	
metadata_json	text	YES		NULL	
source_service	varchar(255)	YES		NULL	
user_email	varchar(255)	NO		NULL	
user_id	varchar(255)	NO	MUL	NULL	
user_role	varchar(255)	NO		NULL	

11 rows in set (0.04 sec)

**Mentor Assignment Management Database****Table mentor\_group**

```
mysql> desc mentor_group;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
assigned_at	datetime(6)	NO		NULL	
batch_id	bigint	NO	MUL	NULL	
group_id	bigint	NO		NULL	
mentor_id	bigint	NO		NULL	

5 rows in set (0.03 sec)

Notification Management Database

Table notifications













```
mysql> desc notifications;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
created_at	datetime(6)	NO	MUL	NULL	
is_read	bit(1)	NO		NULL	
message	varchar(1000)	NO		NULL	
metadata_json	text	YES		NULL	
read_at	datetime(6)	YES		NULL	
recipient_id	bigint	NO	MUL	NULL	
recipient_role	varchar(20)	NO		NULL	
reference_id	bigint	YES		NULL	
reference_type	varchar(60)	YES		NULL	
title	varchar(200)	NO		NULL	
type	varchar(50)	NO		NULL	

12 rows in set (0.03 sec)

## Project Management Database

### Collection projects Schema

<b>_id</b> objectId	  
<b>_class</b> string	com.projexflow.pms.ProjexFlow_PMS.entity.Project
<b>adminEdits</b> array document	Array of documents with 3 nested fields. Array lengths min: 1 average: 1.0 max: 1
<b>batchId</b> long	1
<b>createdAt</b> date	  
<b>createdByStudentId</b> long	1001
<b>description</b> string	Attendance system using face recognition and AI.
<b>description</b> string	Attendance system using face recognition and AI.
<b>docsUrl</b> string	https://docs.google.com/dummy
<b>endDate</b> date	  
<b>groupId</b> long	101
<b>lockedAfterCreate</b> boolean	True
<b>repoUrl</b> string	https://github.com/dummy/smart-attendance
<b>startDate</b> date	  

status

string

IN\_PROGRESS

technologyStack

array

string

Array lengths

min: 5

overage: 5.0

max: 5

title

string

Smart Attendance System

updatedAt

date

S

M

T

W

T

F

S

0000

6000

12000

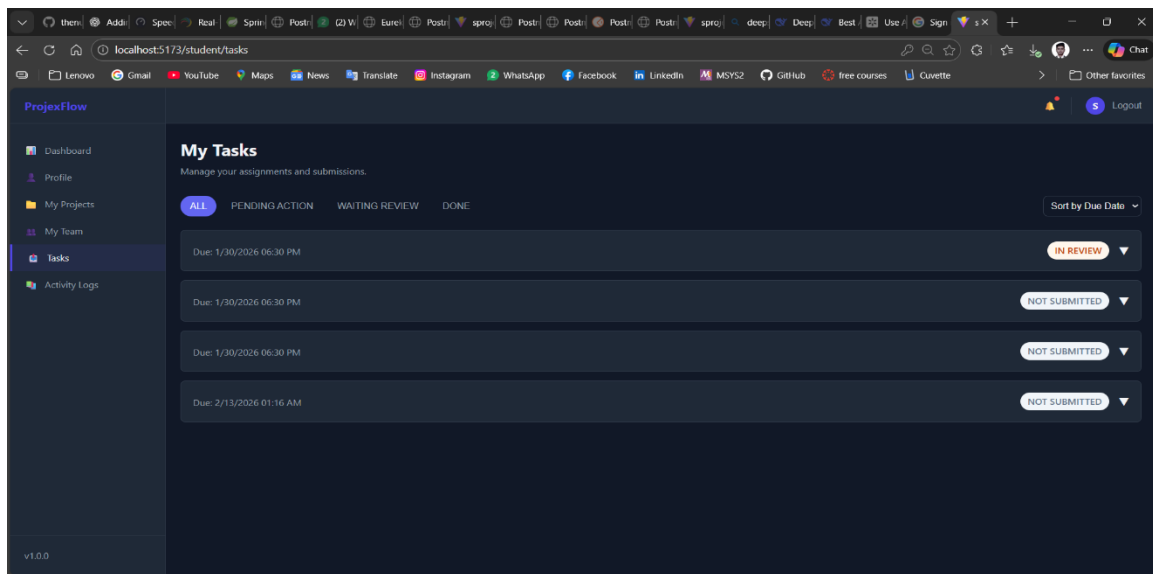
18000

24000

2024-03-25 18:54:57

## 5. SNAPSHOTS

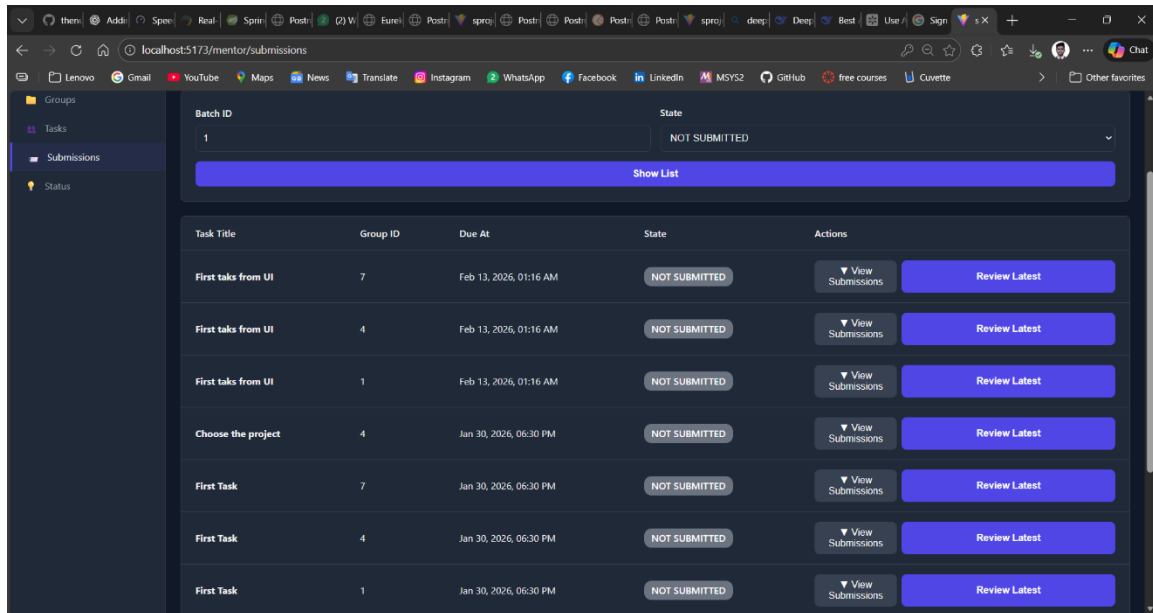
### Assigned task page



### Task status of groups under mentor

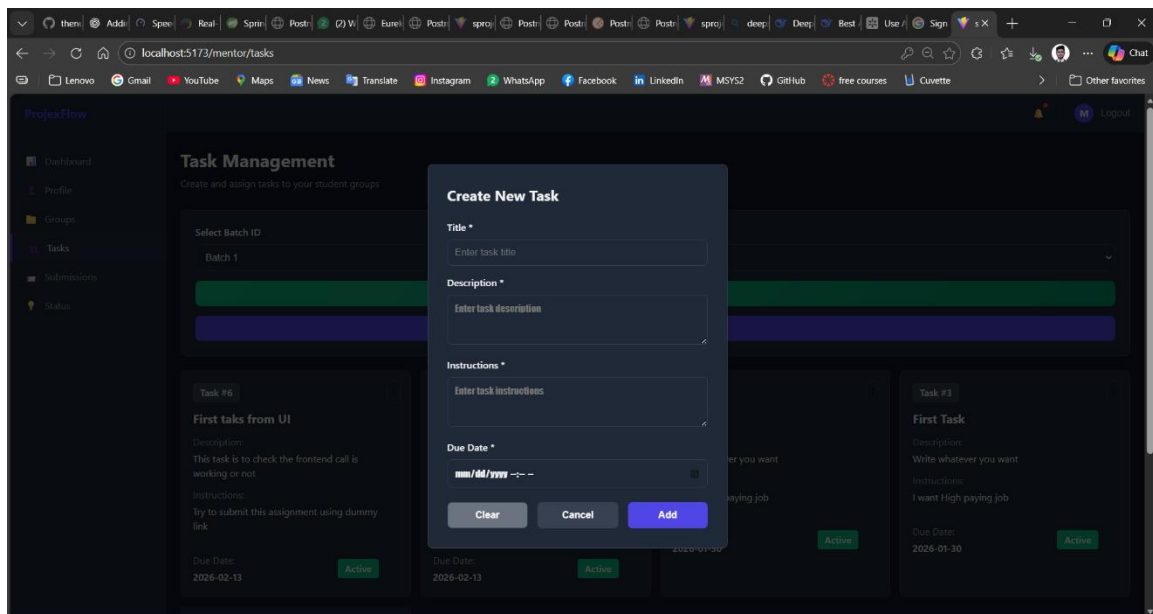
Task Title	Group ID	Due At	State	Actions
First taks from UI	7	Feb 13, 2026, 01:16 AM	NOT SUBMITTED	<a href="#">▼ View Submissions</a> <a href="#">Review Latest</a>
First taks from UI	4	Feb 13, 2026, 01:16 AM	NOT SUBMITTED	<a href="#">▼ View Submissions</a> <a href="#">Review Latest</a>
First taks from UI	1	Feb 13, 2026, 01:16 AM	NOT SUBMITTED	<a href="#">▼ View Submissions</a> <a href="#">Review Latest</a>
Choose the project	4	Jan 30, 2026, 06:30 PM	NOT SUBMITTED	<a href="#">▼ View Submissions</a> <a href="#">Review Latest</a>
First Task	7	Jan 30, 2026, 06:30 PM	NOT SUBMITTED	<a href="#">▼ View Submissions</a> <a href="#">Review Latest</a>
First Task	4	Jan 30, 2026, 06:30 PM	NOT SUBMITTED	<a href="#">▼ View Submissions</a> <a href="#">Review Latest</a>
First Task	1	Jan 30, 2026, 06:30 PM	NOT SUBMITTED	<a href="#">▼ View Submissions</a> <a href="#">Review Latest</a>
Choose the project	7	Jan 30, 2026, 06:30 PM	NOT SUBMITTED	<a href="#">▼ View Submissions</a> <a href="#">Review Latest</a>
First Task	7	Jan 30, 2026, 06:30 PM	NOT SUBMITTED	<a href="#">▼ View Submissions</a> <a href="#">Review Latest</a>

## List of submission with their status



Task Title	Group ID	Due At	State	Actions
First taks from UI	7	Feb 13, 2026, 01:16 AM	NOT SUBMITTED	<a href="#">View Submissions</a> <a href="#">Review Latest</a>
First taks from UI	4	Feb 13, 2026, 01:16 AM	NOT SUBMITTED	<a href="#">View Submissions</a> <a href="#">Review Latest</a>
First taks from UI	1	Feb 13, 2026, 01:16 AM	NOT SUBMITTED	<a href="#">View Submissions</a> <a href="#">Review Latest</a>
Choose the project	4	Jan 30, 2026, 06:30 PM	NOT SUBMITTED	<a href="#">View Submissions</a> <a href="#">Review Latest</a>
First Task	7	Jan 30, 2026, 06:30 PM	NOT SUBMITTED	<a href="#">View Submissions</a> <a href="#">Review Latest</a>
First Task	4	Jan 30, 2026, 06:30 PM	NOT SUBMITTED	<a href="#">View Submissions</a> <a href="#">Review Latest</a>
First Task	1	Jan 30, 2026, 06:30 PM	NOT SUBMITTED	<a href="#">View Submissions</a> <a href="#">Review Latest</a>

## Creating a new task



### Task Management

Create and assign tasks to your student groups

Select Batch ID: Batch 1

#### Create New Task

Title \*  
Enter task title

Description \*  
Enter task description

Instructions \*  
Enter task instructions

Due Date \*  
mm/dd/yyyy --

[Clear](#) [Cancel](#) [Add](#)

#### Task #6

**First taks from UI**

Description  
This task is to check the frontend call is working or not

Instructions  
Try to submit this assignment using dummy link

Due Date: 2026-02-13 [Action](#)

#### Task #3

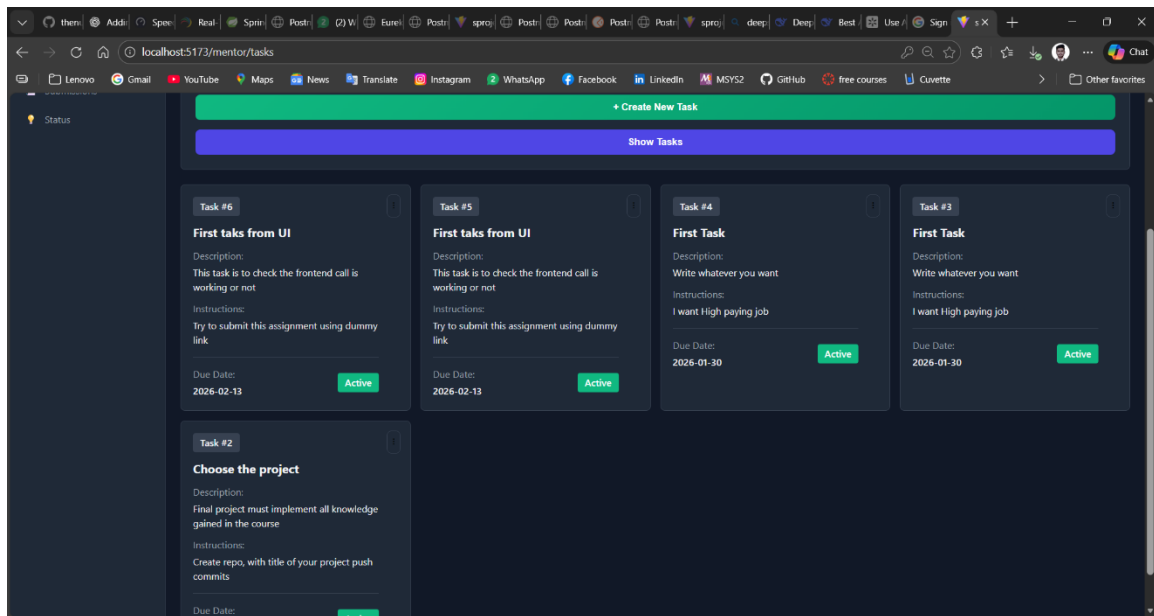
**First Task**

Description  
Write whatever you want

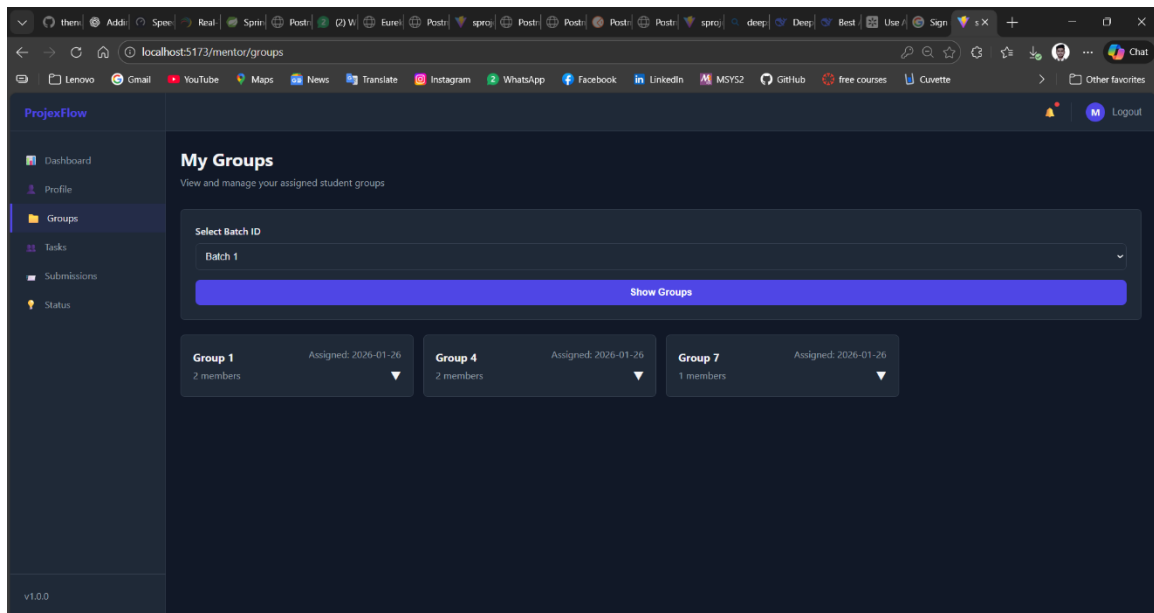
Instructions  
I want High paying job

Due Date: 2026-01-30 [Action](#)

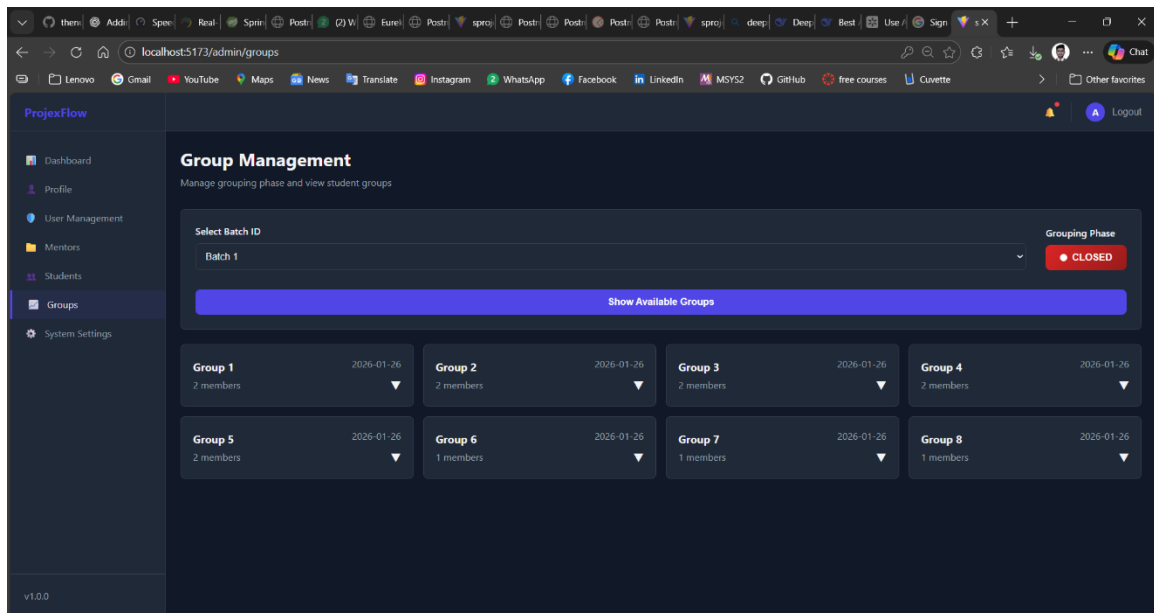
## Listing the created tasks



## Listing the groups under mentor



## Listing all groups in a batch



## Listing all students in a batch

The screenshot shows the 'Students' interface in ProjexFlow. The left sidebar contains navigation links: Dashboard, Profile, User Management, Mentors, Students (selected), and System Settings. The main content area displays a table of students with columns: ROLL NO, NAME, EMAIL, COURSE, and STATUS.

ROLL NO	NAME	EMAIL	COURSE	STATUS
IT-24-003	Aditya Joshi	aditya.joshi@student.projexflow.edu	B.Tech IT	Active
01	Student 01	student01@projexflow.com	CS	Active
02	Student 02	student02@projexflow.com	CS	Active
03	Student 03	student03@projexflow.com	CS	Active
04	Student 04	student04@projexflow.com	CS	Active
05	Student 05	student05@projexflow.com	CS	Active
06	Student 06	student06@projexflow.com	CS	Active
07	Student 07	student07@projexflow.com	CS	Active
08	Student 08	student08@projexflow.com	CS	Active
09	Student 09	student09@projexflow.com	CS	Active
11	Student 11	student11@projexflow.com	CS	Active
12	Student 12	student12@projexflow.com	CS	Active

## Adding new student record

The screenshot shows the ProjexFlow admin dashboard with the 'Add New Student' modal open. The modal contains the following fields:

- Full Name \***: Enter full name
- Email \***: admin1@projexflow.edu
- Password \***: [Masked]
- Profile Photo URL \***: https://example.com/photo.jpg
- Roll No \***: IT-24-003
- PRN \***: PRN2024IT0003
- GitHub URL \***: https://github.com/username
- Course \***: [Dropdown menu]

The background shows the 'Student Management' section with a sidebar containing: Dashboard, Profile, User Management, Mentors, Students, Groups, and System Settings. The main content area has a filter by Course ID and a table with columns: ROLL NO, COURSE, and STATUS.

## Listing the mentors in admin dashboard

The screenshot shows the ProjexFlow admin dashboard with the 'Mentors' list displayed. The list is filtered by Course ID and shows the following data:

PROFILE	NAME	EMAIL	COURSE	STATUS
	Rahul Patil	mentor.rahul@projexflow.edu	B.Tech IT	Active
	Dr. Suresh Kulkarni	mentor01.suresh@projexflow.com	CS	Active
	Prof. Anita Desai	mentor02.anita@projexflow.com	IT	Active
	Mr. Rahul Mehta	mentor03.rahul@projexflow.com	CS	Active
	Ms. Priya Nair	mentor04.priya@projexflow.com	IT	Active
	Dr. Vikram Joshi	mentor05.vikram@projexflow.com	CS	Active
	Prof. Sneha Iyer	mentor06.sneha@projexflow.com	IT	Active
	Rahul Subhramanyam	rahul1@gmail.com	IT	Active

The sidebar contains: Students, Groups, and System Settings. The main content area has a filter by Course ID and a 'Show List' button.

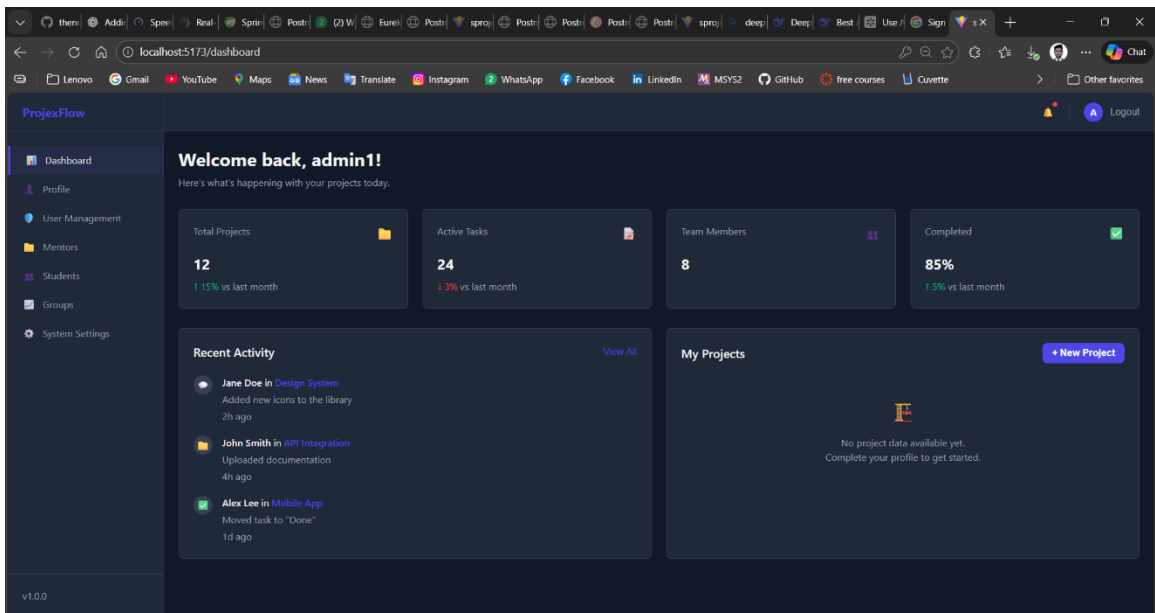
## Adding mentor record

The screenshot shows the 'Add New Mentor' form in the ProjexFlow Admin Dashboard. The form is a modal overlay on top of the 'Mentor Management' page. The form fields are as follows:

- Full Name \***: Text input field with placeholder 'Enter full name'.
- Email \***: Text input field with value 'admin1@projexflow.edu'.
- Password \***: Password input field with placeholder '\*\*\*\*\*'.
- Profile Photo URL \***: Text input field with placeholder 'https://example.com/photo.jpg'.
- Course \***: Text input field with value 'B.Tech IT'.
- Employee ID \***: Text input field with value 'EMP-IT-1021'.
- Department \***: Text input field with value 'Information Technology'.
- Active**: A checked checkbox.

The background page shows the 'Mentor Management' section with a sidebar menu containing: Dashboard, Profile, User Management, Mentors, Students, Groups, and System Settings. The main content area has a 'Filter by Course' dropdown and a table of mentors.

## Admin dashboard



## 6. CONCLUSION

The development of ProjexFlow - Smart Academic Project Management System successfully demonstrates the application of modern software engineering principles and technologies to solve real-world problems in academic project management. This project has achieved its primary objective of creating a comprehensive, scalable, and user-friendly platform that streamlines the entire lifecycle of academic project management.

### Achievement of Objectives

All primary objectives outlined at the beginning of this project have been successfully accomplished:

1. **Microservices Architecture:** Successfully implemented a robust microservices architecture with nine independent services, ensuring scalability, maintainability, and fault tolerance.
2. **Role-Based Access Control:** Implemented comprehensive role-based access control with three distinct user roles (Admin, Mentor, Student), each with appropriate functionalities and permissions.
3. **User Management:** Developed a complete user management system with secure authentication, profile management, and photo upload capabilities.
4. **Group Management:** Created an efficient group formation system with request/approval workflow and batch-wise organization.
5. **Task Management:** Built a comprehensive task management system enabling task creation, assignment, submission, and evaluation.
6. **Project Tracking:** Implemented project management capabilities for tracking group projects and milestones.
7. **Responsive Frontend:** Developed a modern, responsive React-based user interface that works across different devices.

## **7. REFERENCES**

1. Spring Framework Documentation. "Spring Boot Reference Documentation." Spring.io, 2025. <https://spring.io/projects/spring-boot>
2. Spring Cloud Documentation. "Spring Cloud Reference Documentation." Spring.io, 2025. <https://spring.io/projects/spring-cloud>
3. Netflix OSS. "Eureka - Service Discovery Server." GitHub, 2024. <https://github.com/Netflix/eureka>
4. React Documentation. "React - A JavaScript library for building user interfaces." React.dev, 2025. <https://react.dev>
5. Richardson, Chris. "Microservices Patterns: With examples in Java." Manning Publications, 2018.
6. Newman, Sam. "Building Microservices: Designing Fine-Grained Systems." O'Reilly Media, 2nd Edition, 2021.
7. Walls, Craig. "Spring Boot in Action." Manning Publications, 2016.
8. Carneiro, Piotr Mińkowski. "Mastering Spring Cloud." Packt Publishing, 2018.
9. JWT.io. "JSON Web Tokens - Introduction." Auth0, 2025. <https://jwt.io/introduction>
10. MySQL Documentation. "MySQL 8.0 Reference Manual." Oracle Corporation, 2025. <https://dev.mysql.com/doc/>
11. MongoDB Documentation. "MongoDB Manual." MongoDB Inc., 2025. <https://www.mongodb.com/docs/>
12. Hibernate ORM Documentation. "Hibernate ORM User Guide." Red Hat, 2025. <https://hibernate.org/orm/documentation/>
13. Axios Documentation. "Promise based HTTP client for the browser and node.js." GitHub, 2025. <https://axios-http.com/>
14. React Router Documentation. "Declarative routing for React." Remix Software, 2025. <https://reactrouter.com/>
15. Cloudinary Documentation. "Image and Video Upload, Storage, Optimization and CDN." Cloudinary, 2025. <https://cloudinary.com/documentation>
16. Fowler, Martin. "Patterns of Enterprise Application Architecture." Addison-Wesley, 2002.