Design Document

Smart Course Registration System (SCRS)

Course: Software Engineering Project – Phase 2
Instructor: Mohameds Sami Rakha

Prepared by:

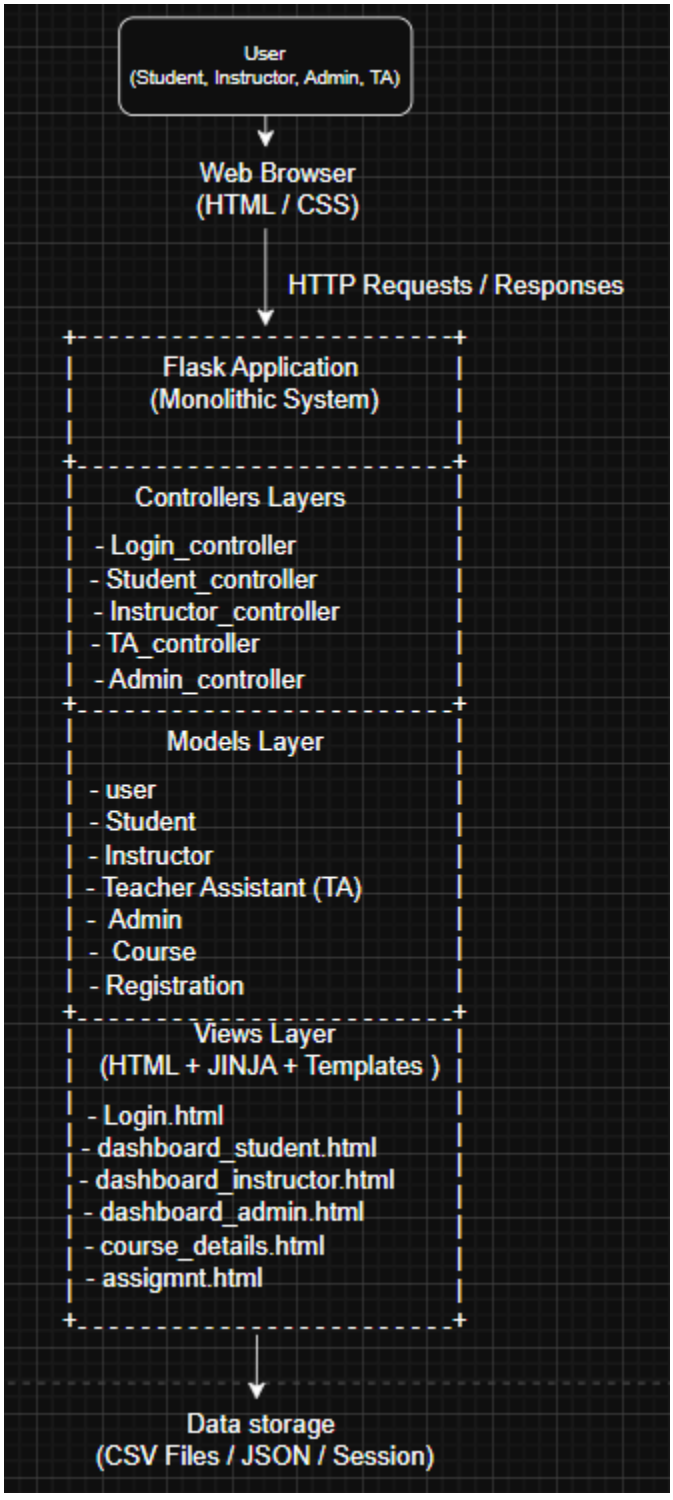
Team Member Name	Student ID	Student Email
Mariam Ahmed Maher	202400733	s-mariam.maher@zewailcity.edu.eg
Farida Ehab	202201509	s-farida.ashour@zewailcity.edu.eg
Elaliaa Elmoez	202402473	s-elaliaa.kotb@zewailcity.edu.eg
Rahma Ali	202400365	s-rahma.anwar@zewailcity.edu.eg

1. Introduction
   1.1 Purpose of the Document
   This document defines how the SCRS will be built using the MVC model.
   1.2 Scope of the Design Phase
   It will cover the system architecture, database design, UI wireframes and component designs.
   1.3 Intended Audience
   The Student body of the university (including students,instructors,TAs and admins)
   1.4 Overview of the Contents
   This will give an overall roadmap of the design decisions and implementations strategy.

2. System Overview
   2.1 Brief Description of the System
   The SCRS is a web based registration system with role-based access that allows for different functionalities depending on role in the university.
   2.2 Key Design Goals and Constraints
   The main design goals are maintaining the scalability, security and easy usage for users. The constraints include using python flask and sql database as a backend and using HTML for the front end and utilising MVC architecture.

3. Architectural Design

3.1 System Architecture Diagram



3.2 Discussion of Architecture Style and Components

The Smart Course Registration System follows a Monolithic Architecture, where all the components of the system—controllers, models, views, and the data layer—run in one united Flask application. In this architecture, the entire system is deployed as an application, with all modules running under the same runtime environment and sharing the same memory space, which eases the burden of developing and maintaining communications between them.
The architecture is organized into three main layers according to the MVC pattern:
Controller Layer: This layer is responsible for system routing, role-based navigation-student, instructor, TA, admin-and processing of all HTTP requests.

Model Layer: Represents the system entities, including User, Student, Instructor, Teaching Assistant, Admin, Course, and Registration. It encapsulates data and business rules such as enrollment validation, schedule conflict detection, and permissions.

View Layer: Implementation of user interfaces is carried out in HTML and Jinja templates for dashboards, course pages, assignments, and admin pages.

This architecture fits SCRS because the system is of medium size, role-based, and does not require distributed microservices. A monolithic structure ensures fast internal communication and easier maintenance during the academic development timeline.

3.3 Technology Stack and Tools :

The project will use
BACKEND : python flask.
FRONTEND: html,css/bootstrap,
DATABASE: MySQL
Development Enviromint : VS Code
Design Tools: Drow.io
Version Control : Git / GitHub
Testing Tools : Manual testing

4. Detailed Design

4.1 Model-View-Controlled(MVC) Design Pattern

4.1.1 Description of MVC pattern

The Model–View–Controller pattern separates the application into three components to improve modularity:

Model:

This package contains system data and business logic, including classes like Student, Instructor, TA, Admin, Course, and Registration.

Models manage validation (e.g., course conflicts) and course capacity, and also represent real university entities.

View:

Responsible for the user interface.

Built using HTML and Jinja templates including but not limited to: login.html, student dashboard, course pages, assignments, admin screens.

Views display data passed to them by controllers without carrying out any logic.

Controller:

Handles routing and orchestrates system flow.

Controllers manage user authentication, course registration, assignment actions, and permission-based navigation based on the user's role: student, TA, instructor, admin. They communicate with models and decide which view to render. SCRS uses MVC for the separation of logic, supporting multiple roles more effectively while ensuring the system remains maintainable and scalable.

4.1.2 Mapping of Project Components to MVC

Model Layer

Model class	*Description*
User	*Base class for all system users.*
Student	*Registers/Drop courses, View schedule & grades.*
Instructor	*Posts announcements, assignment, grades, manages TA.*
TeacherAssistant	*Grades assignments, post*

	announcements, assists instructor.
Admin	*Adds/Removes courses, users, roles.*
Course	*Contains course information (seats, schedule, timings).*
Registration	*Links students to courses, enforces conflicts, majors.*

View Layer

View File	*Function*
Login.html	*Login form for all roles.*
dashboard_student.html	*Shows registered course, schedule, grade.*
dashboard_instructor.html	*Manage course, assignments, announcements.*
dashboard_admin.html	*Manage users & courses.*
course_admin.html	*Course information, announcements, assignments.*
assignment.html	*Submission or Grading screen.*

Controller	*purpose*
login_controller	*Authentication, redirects based on role (Checking who the user is, then sending them to the correct page depending on their role).*
student_controller	*Course registration, dropping, viewing schedule.*
instructor_controller	*Posting assignments, announcements, grading .*
TA_controller	*Grading and posting under instructor*

	permission.
admin_controller	*User & course management.*

4.1.3 Responsibilities of Model,View and Controller

Model Responsibilities

- Represent entities: users, courses, registration records.
- Enforce rules: schedule conflicts, course major restrictions, seat availability.
- Store and retrieve data (from session / CSV / JSON).
- Provide methods like:
get_courses(), register(), drop(), creat_assignment().

View Responsibilities

***-** Display data returned by controllers.*
- Provide input forms for login, course registration, assignment submission.
- Render user-specific dashboards based on role.
- Use Jinja2 templates and basic HTML only.

Controller Responsibilities

***-** Process HTTP requests and determine user flow.*
- Perform authentication and role-based direction.
- Call models to modify or retrieve data.
- Pass model results to appropriate views.
- Handle actions: register, drop, create assignment, grade, add user, etc.

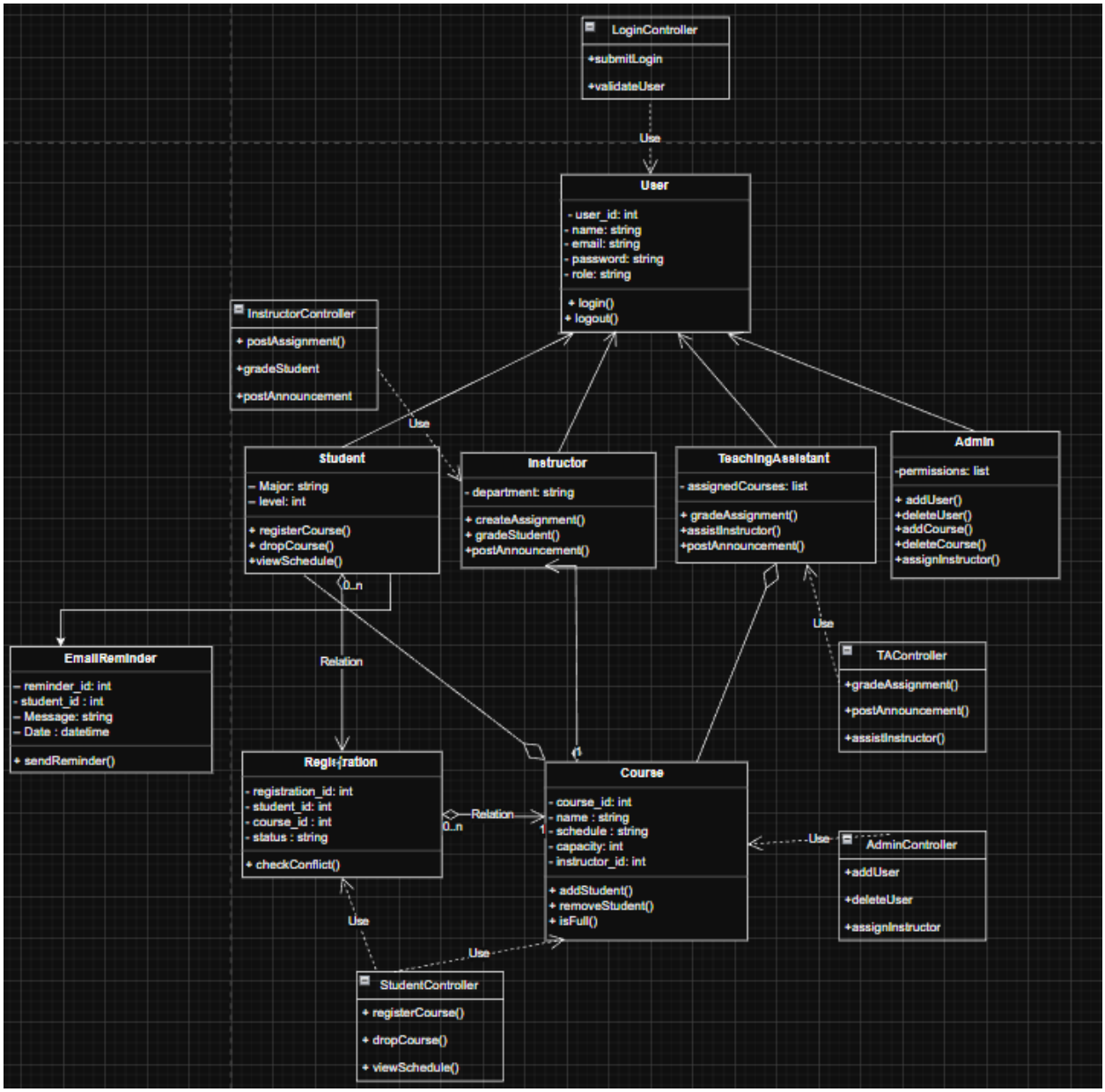
4.1.4 Interaction Between Components

When a user interacts with SCRS, the flow is as follows:

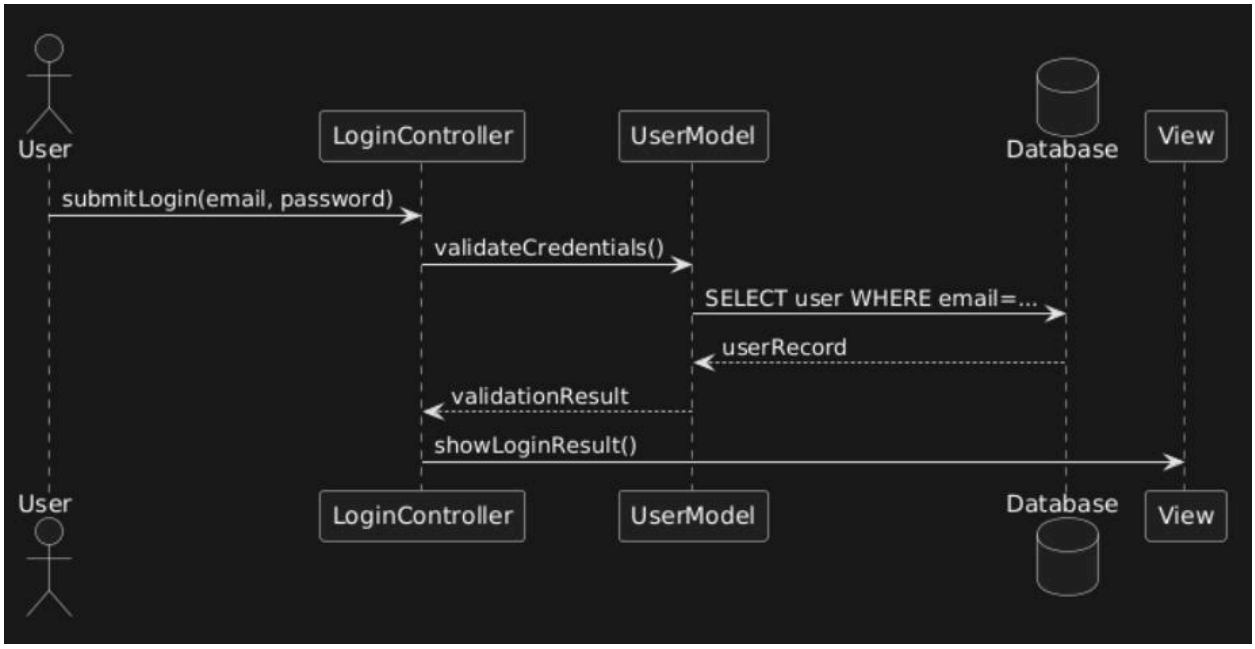
1. The user makes a request, such as registering for a course.
2. The controller receives the request and validates the user's role.
3. The controller calls the Model (for example, Course and Registration) to carry out actions such as checking conflicts or updating enrollment.
4. The model returns the result to the controller.
5. The controller decides which View (HTML template) to render.
6. A view generates the final webpage and sends it to the browser.

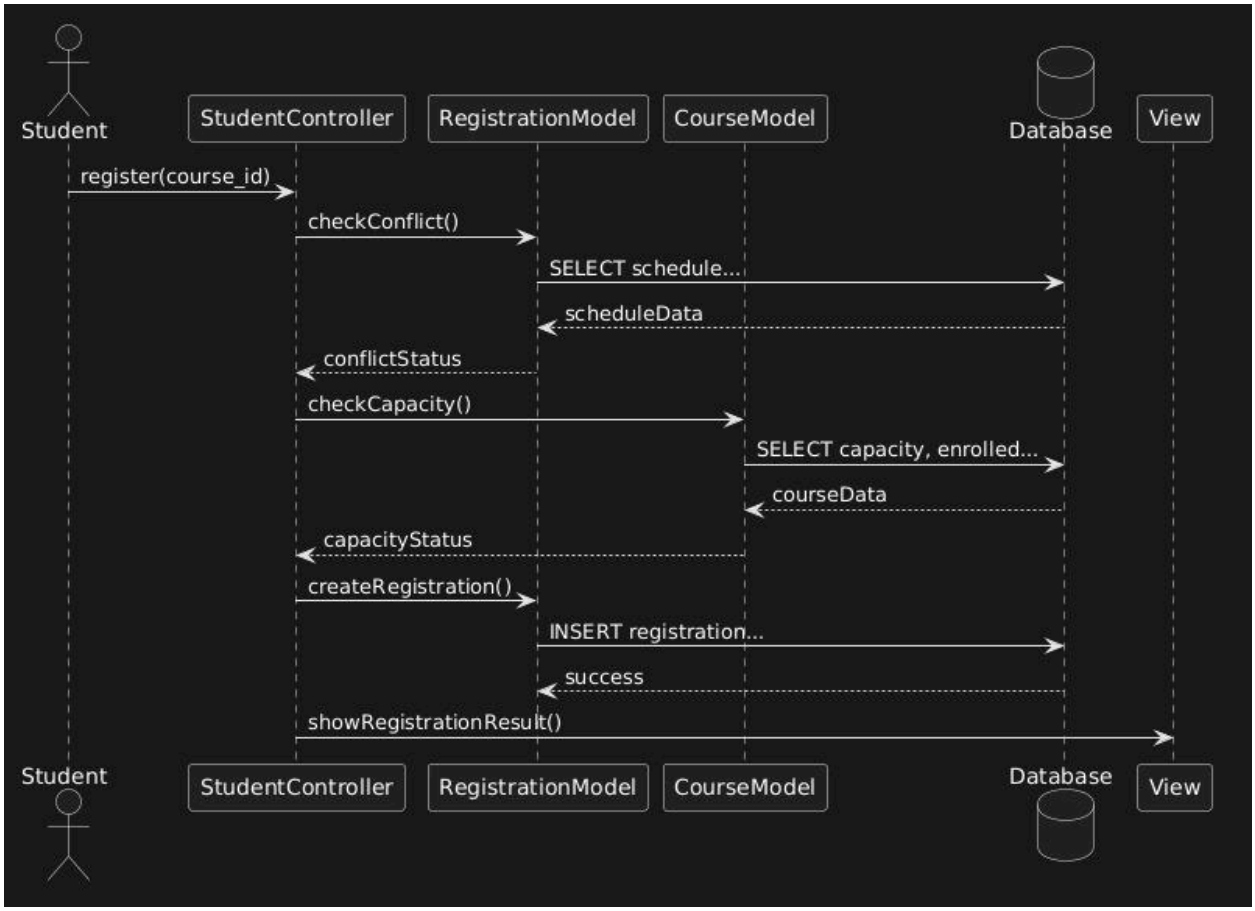
4.2 UML diagrams

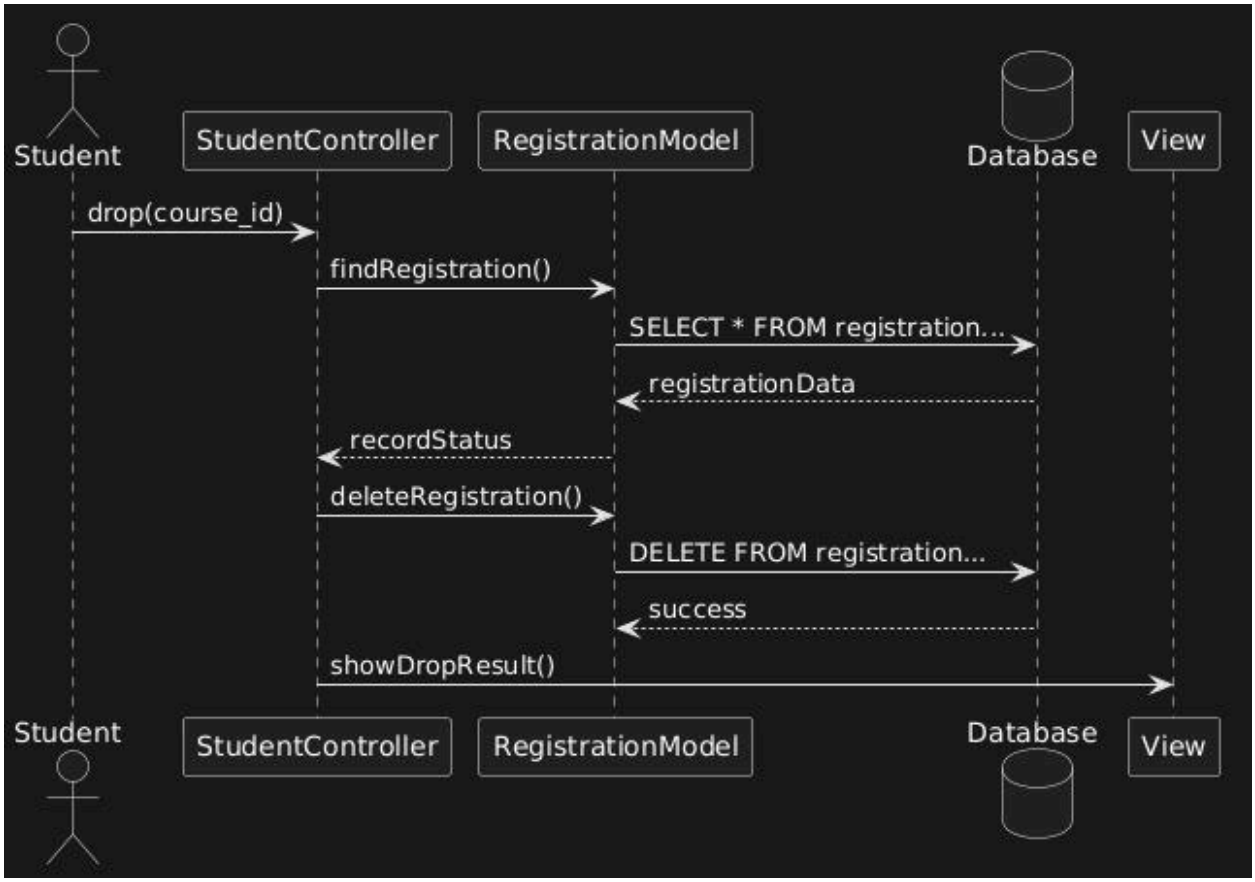
4.2.1 Detailed Class Diagram

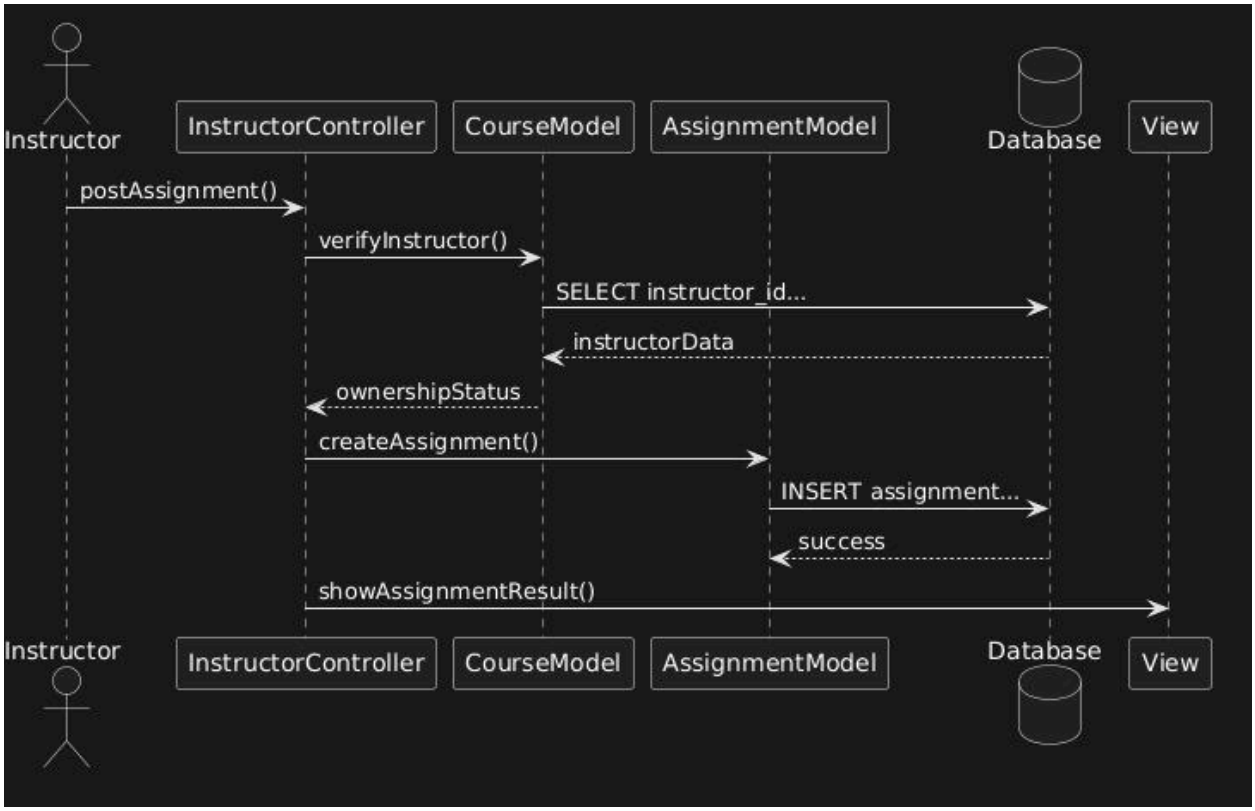


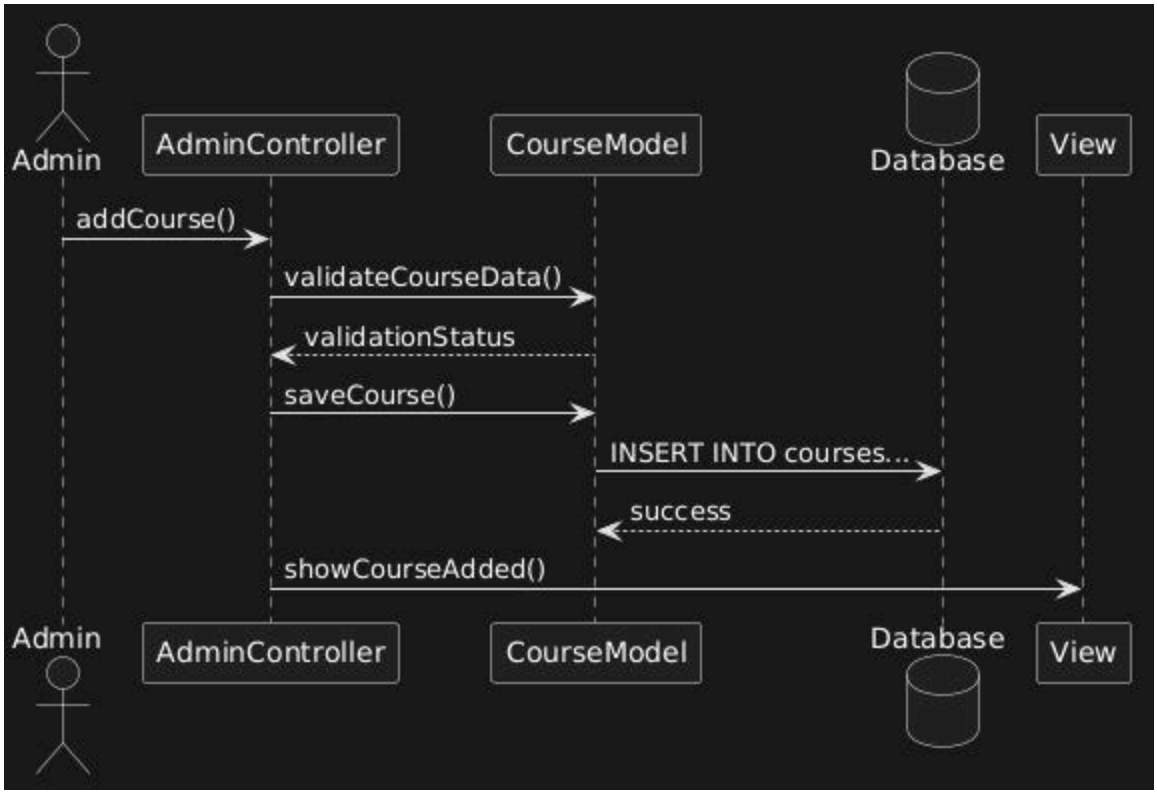
4.2.2Sequence Diagrams


User
LoginController
UserModel
Database
View
submitLogin(email, password)
validateCredentials()
SELECT user WHERE email=...
userRecord
validationResult
showLoginResult()
User
LoginController
UserModel
Database
View


Student
StudentController
RegistrationModel
CourseModel
Database
View
register(course_id)
checkConflict()
SELECT schedule...
scheduleData
conflictStatus
checkCapacity()
SELECT capacity, enrolled...
courseData
capacityStatus
createRegistration()
INSERT registration...
success
showRegistrationResult()
Student
StudentController
RegistrationModel
CourseModel
Database
View

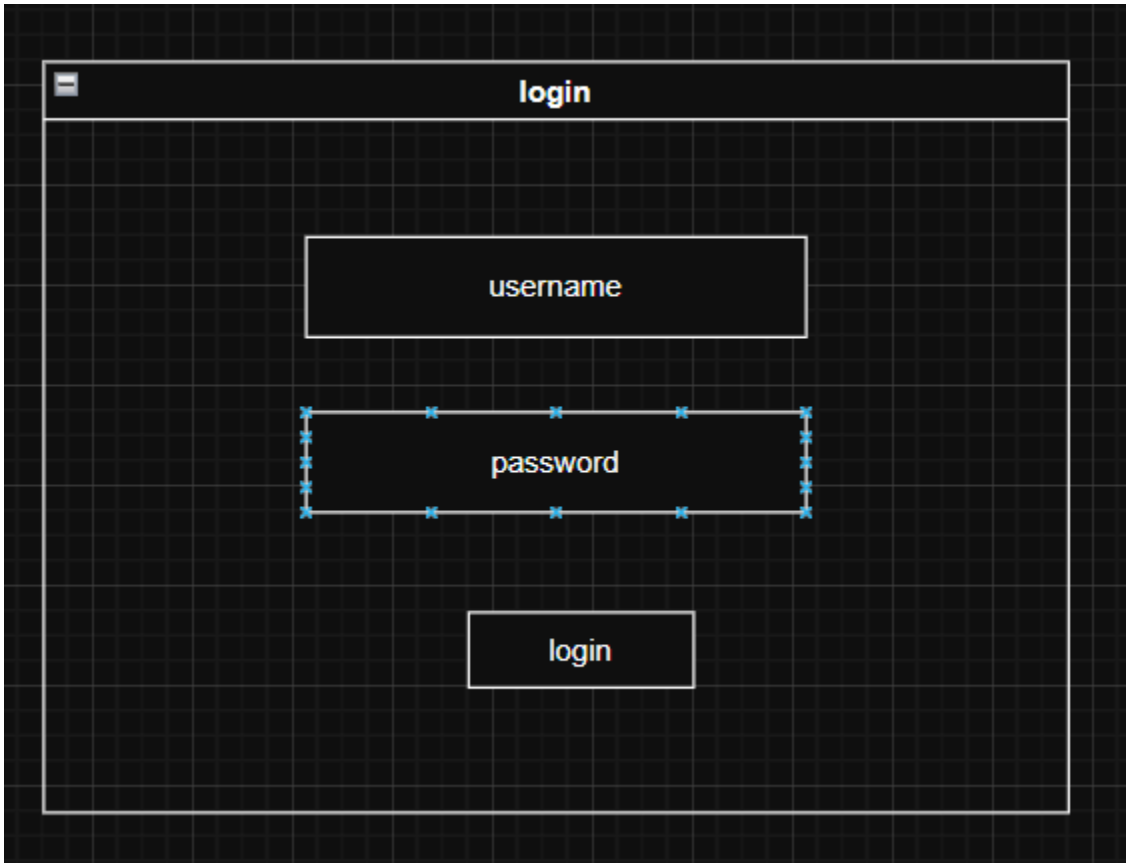

Student
StudentController
RegistrationModel
Database
View
drop(course_id)
findRegistration()
SELECT * FROM registration...
registrationData
recordStatus
deleteRegistration()
DELETE FROM registration...
success
showDropResult()
Student
StudentController
RegistrationModel
Database
View

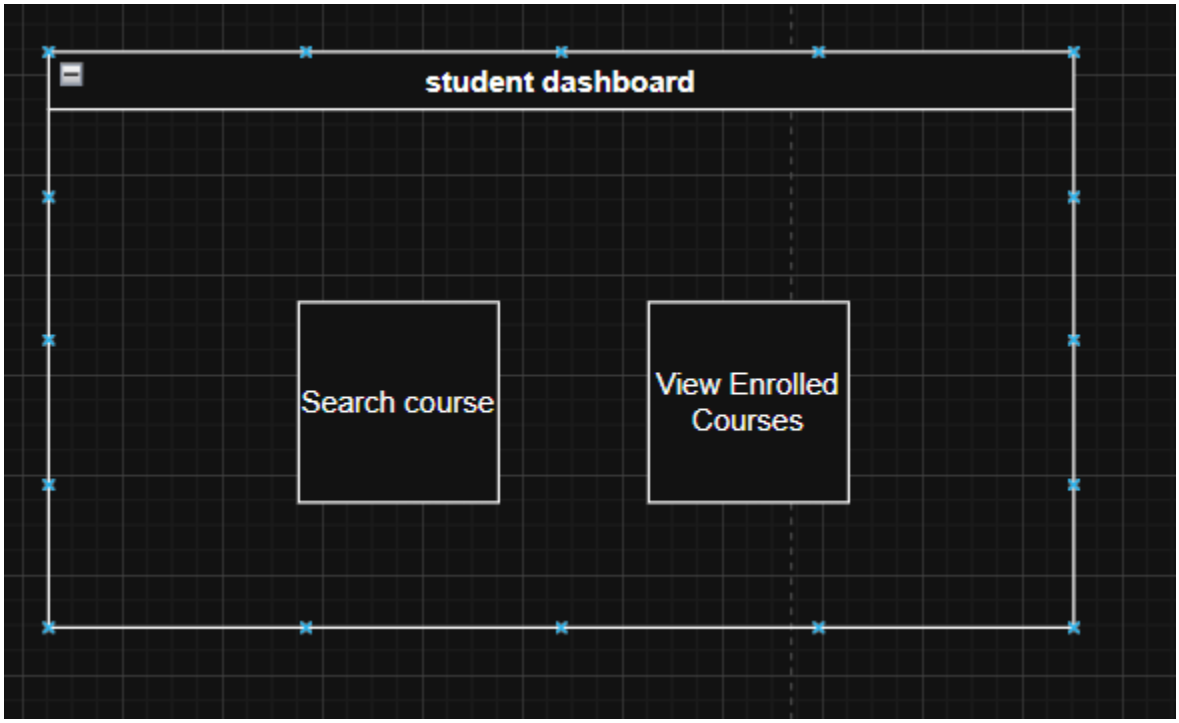

Instructor
InstructorController
CourseModel
AssignmentModel
Database
View
postAssignment()
verifyInstructor()
SELECT instructor_id...
instructorData
ownershipStatus
createAssignment()
INSERT assignment...
success
showAssignmentResult()
Instructor
InstructorController
CourseModel
AssignmentModel
Database
View

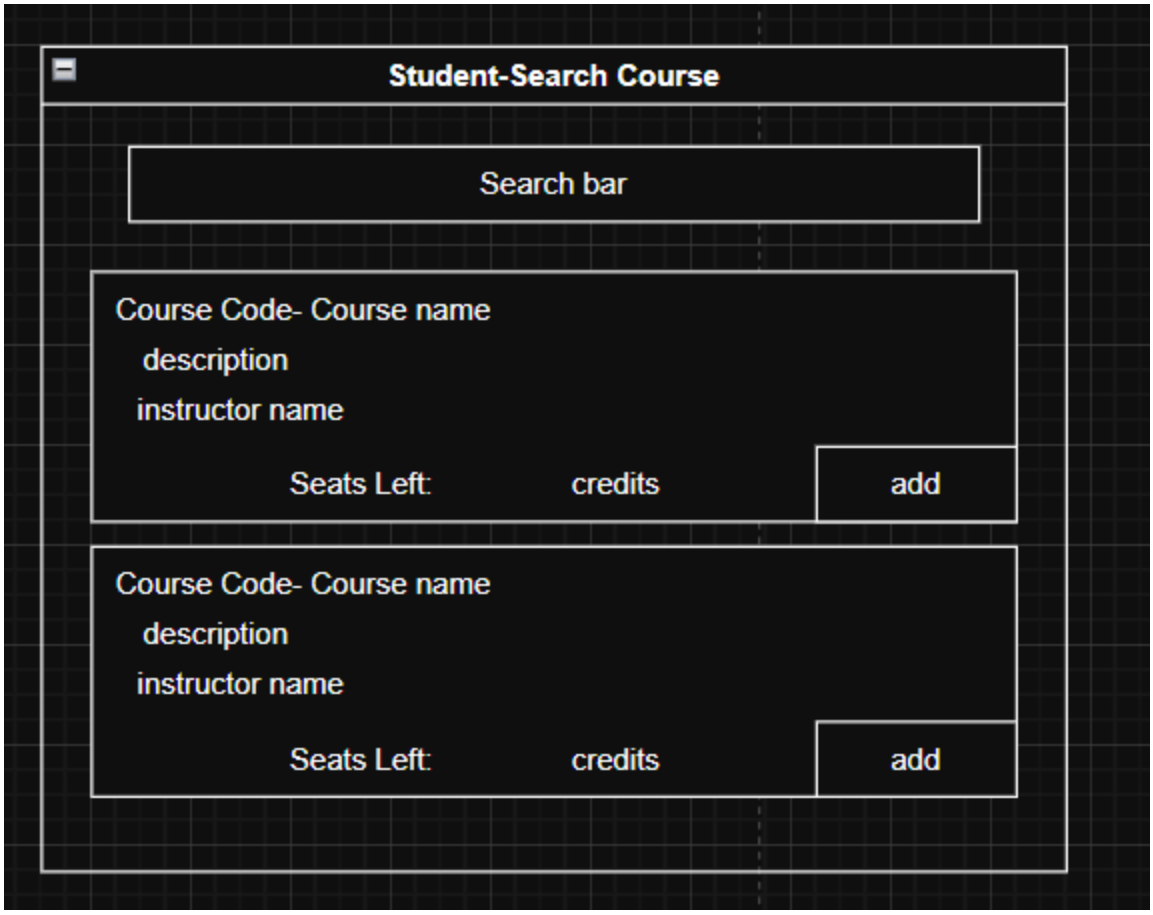


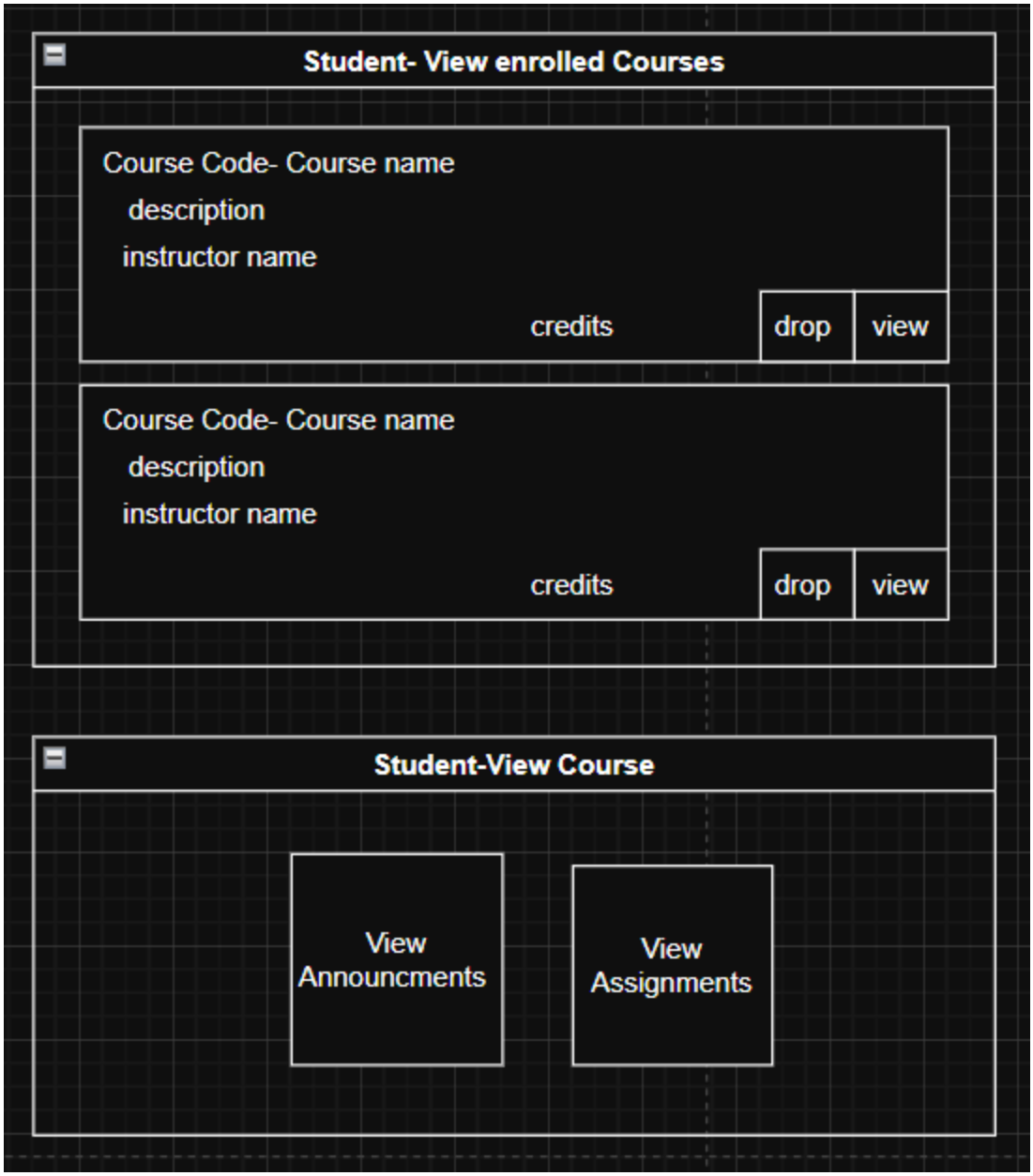
4.3 UI/UX Design

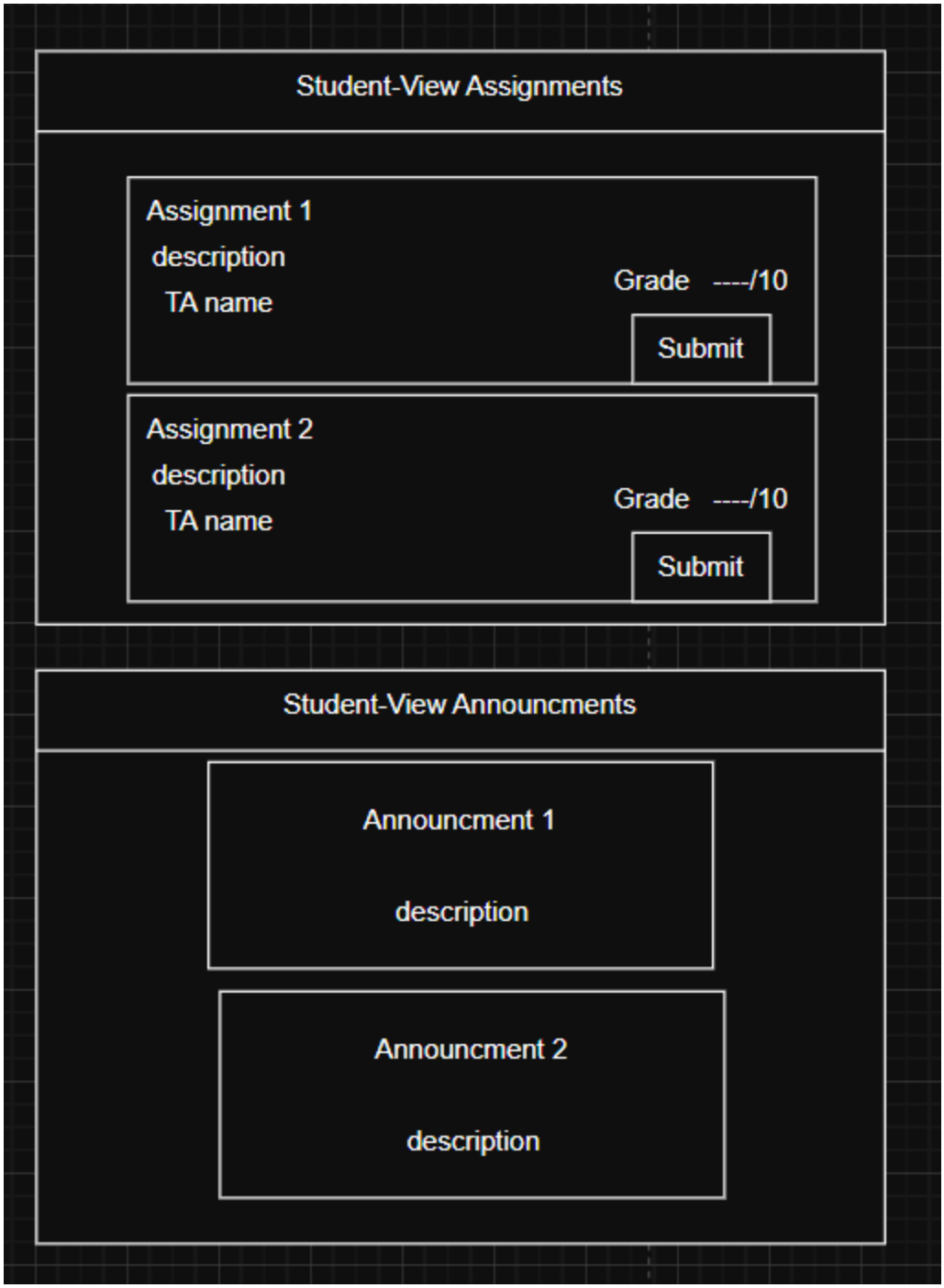
4.3.1 WireFrames/Mockups

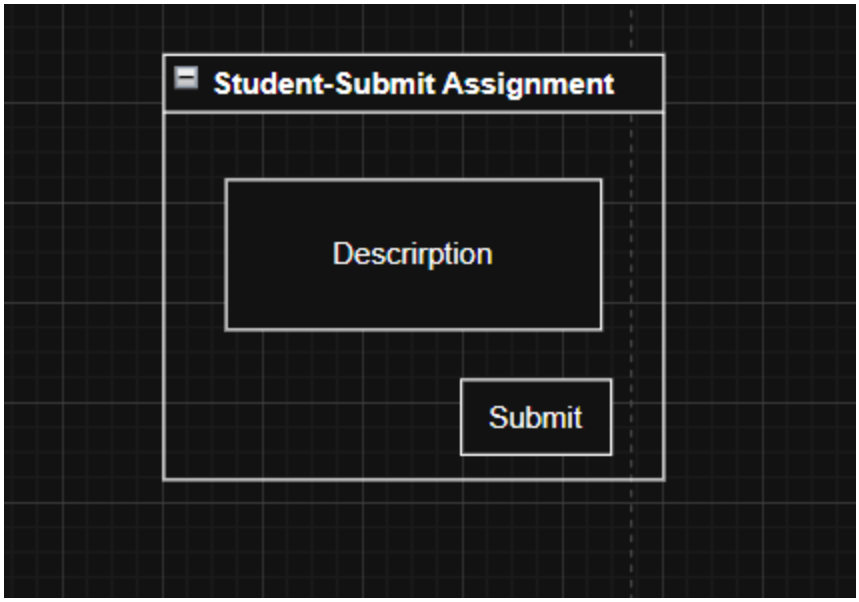

login
username
password
login

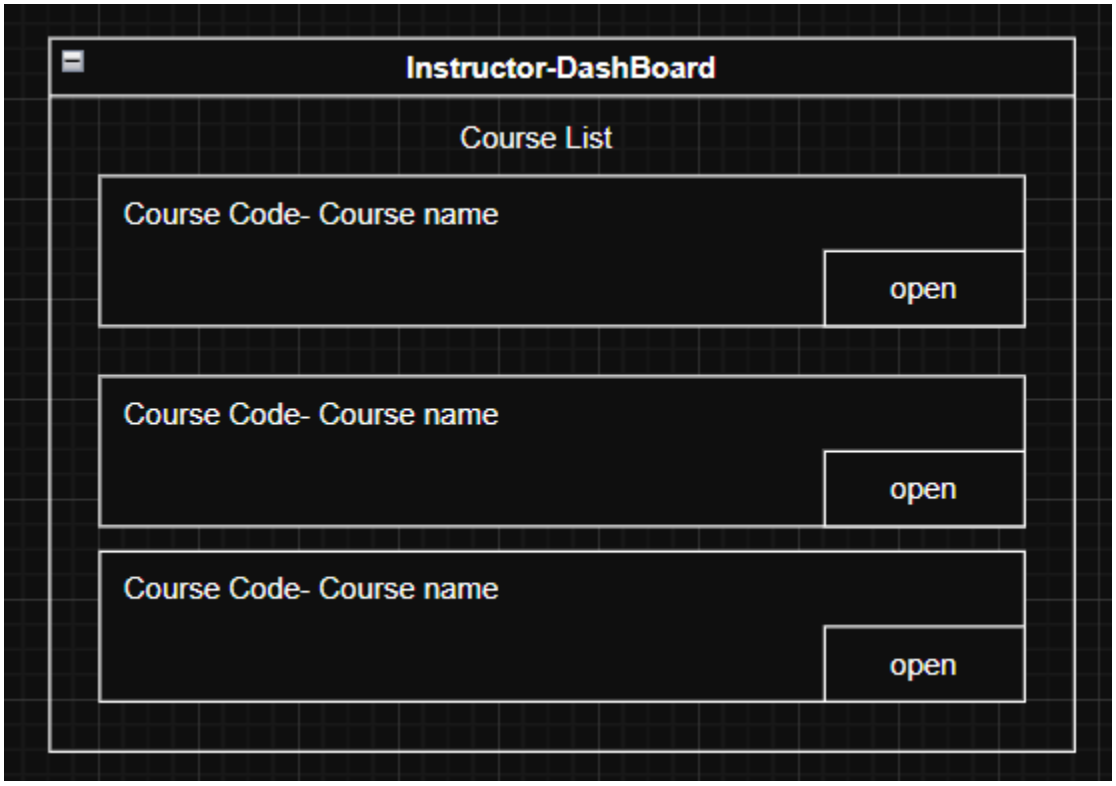

student dashboard
Search course
View Enrolled Courses

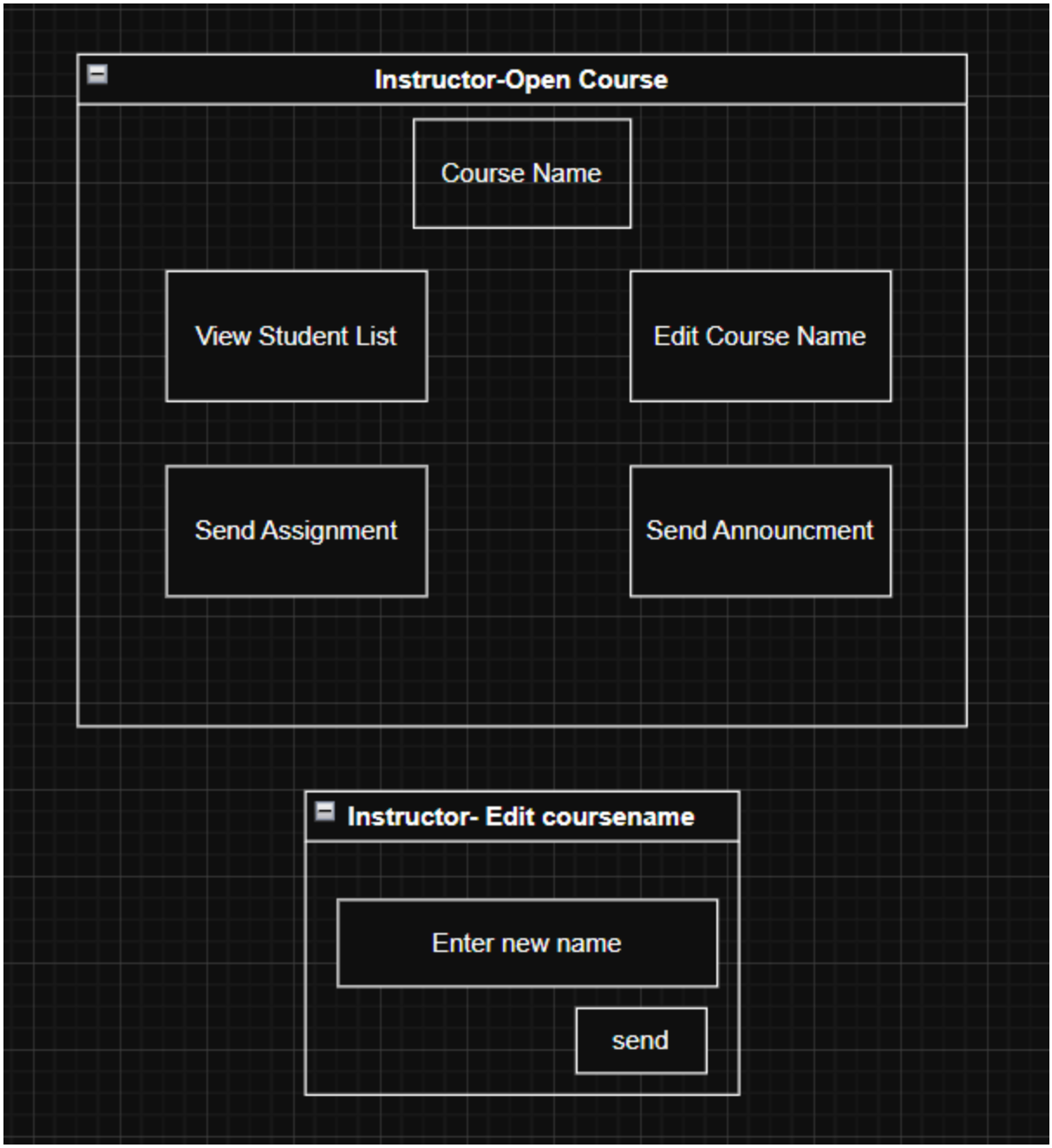

Student-Search Course
Search bar
Course Code- Course name
description
instructor name
Seats Left:
credits
add
Course Code- Course name
description
instructor name
Seats Left:
credits
add

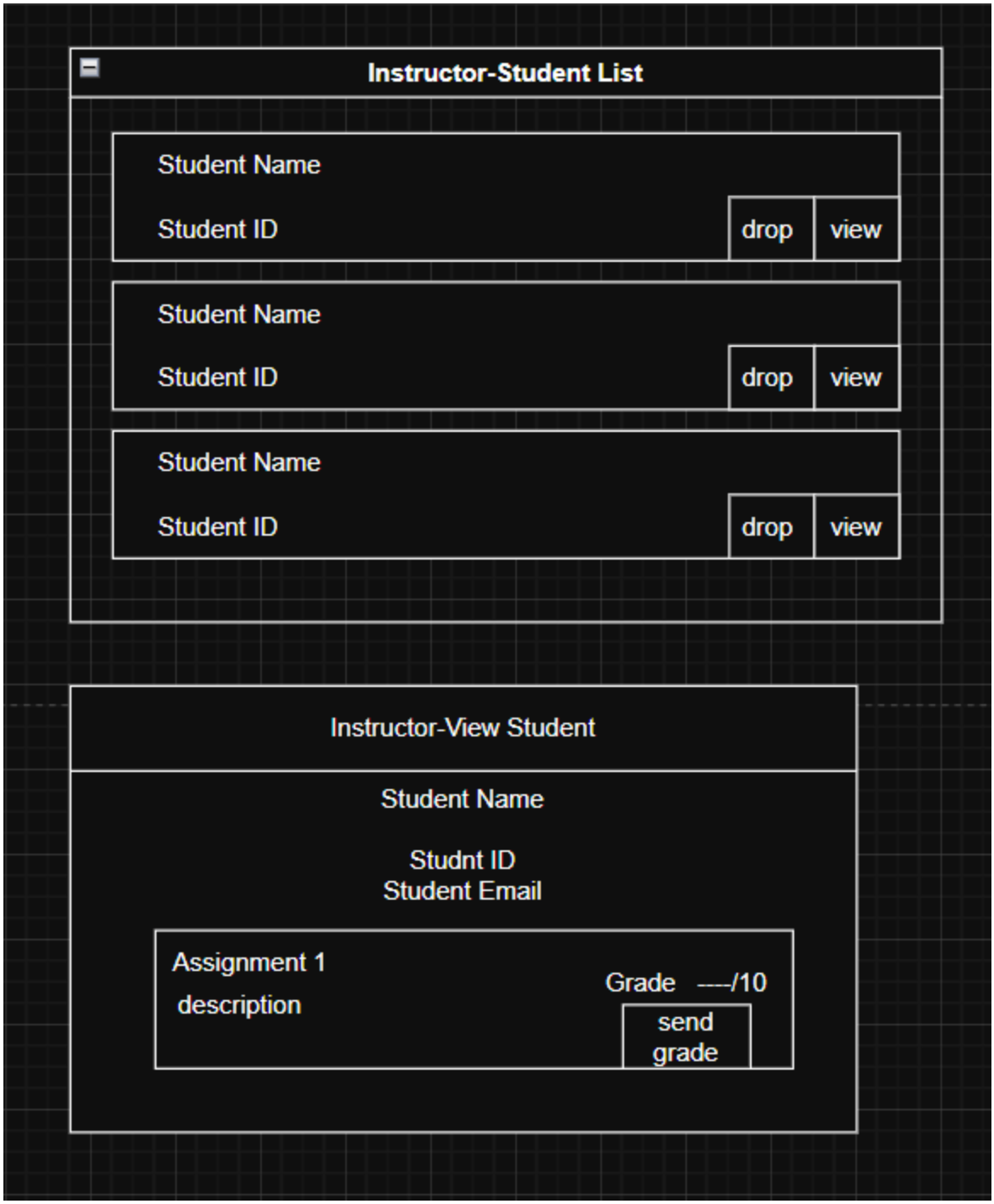

Student- View enrolled Courses
Course Code- Course name
description
instructor name
credits
drop
view
Course Code- Course name
description
instructor name
credits
drop
view
Student-View Course
View Announcments
View Assignments

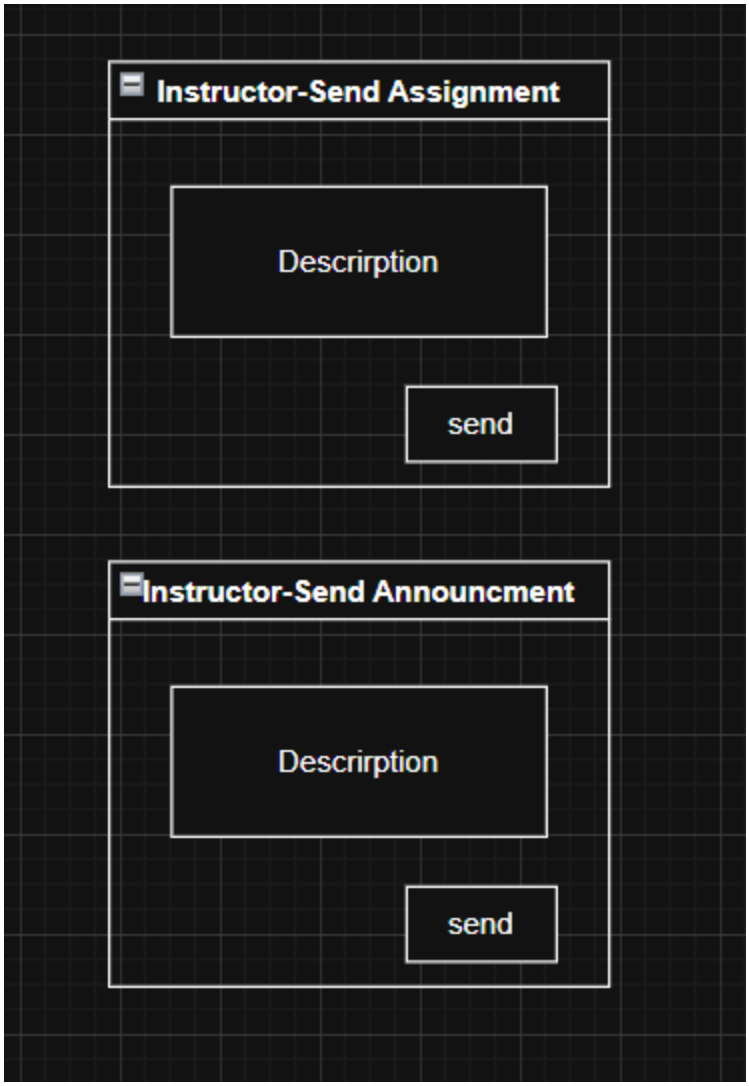

Student-View Assignments
Assignment 1
description
TA name
Grade ----/10
Submit
Assignment 2
description
TA name
Grade ----/10
Submit
Student-View Announcments
Announcment 1
description
Announcment 2
description

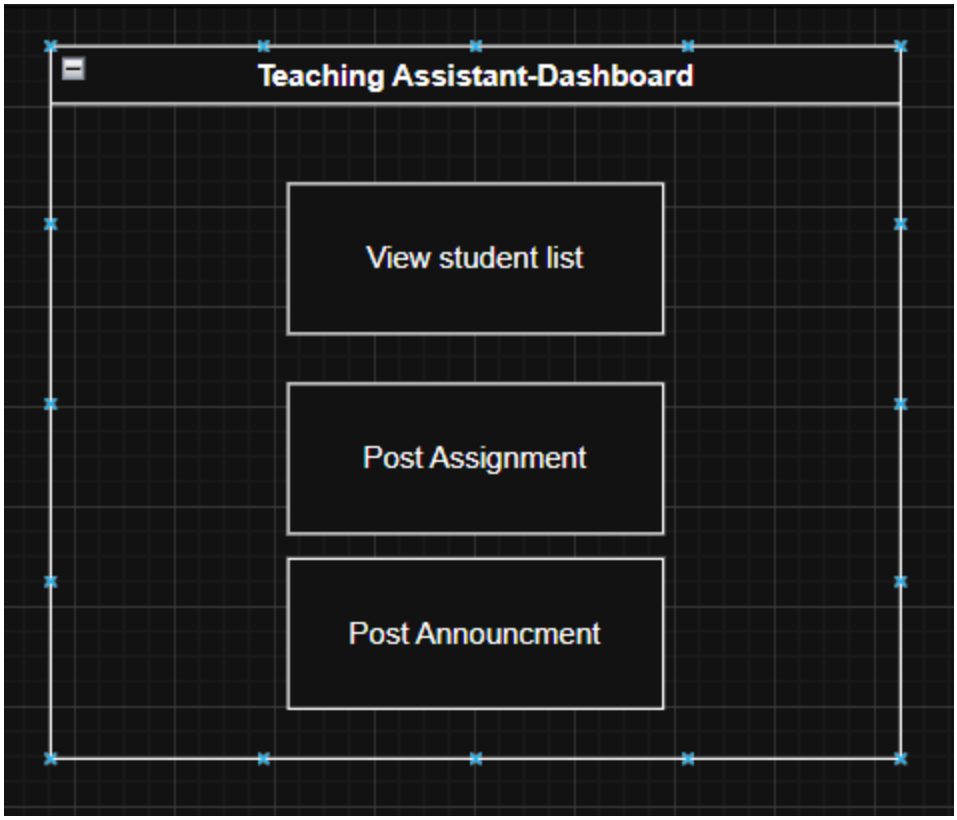

Student-Submit Assignment
Descrirption
Submit

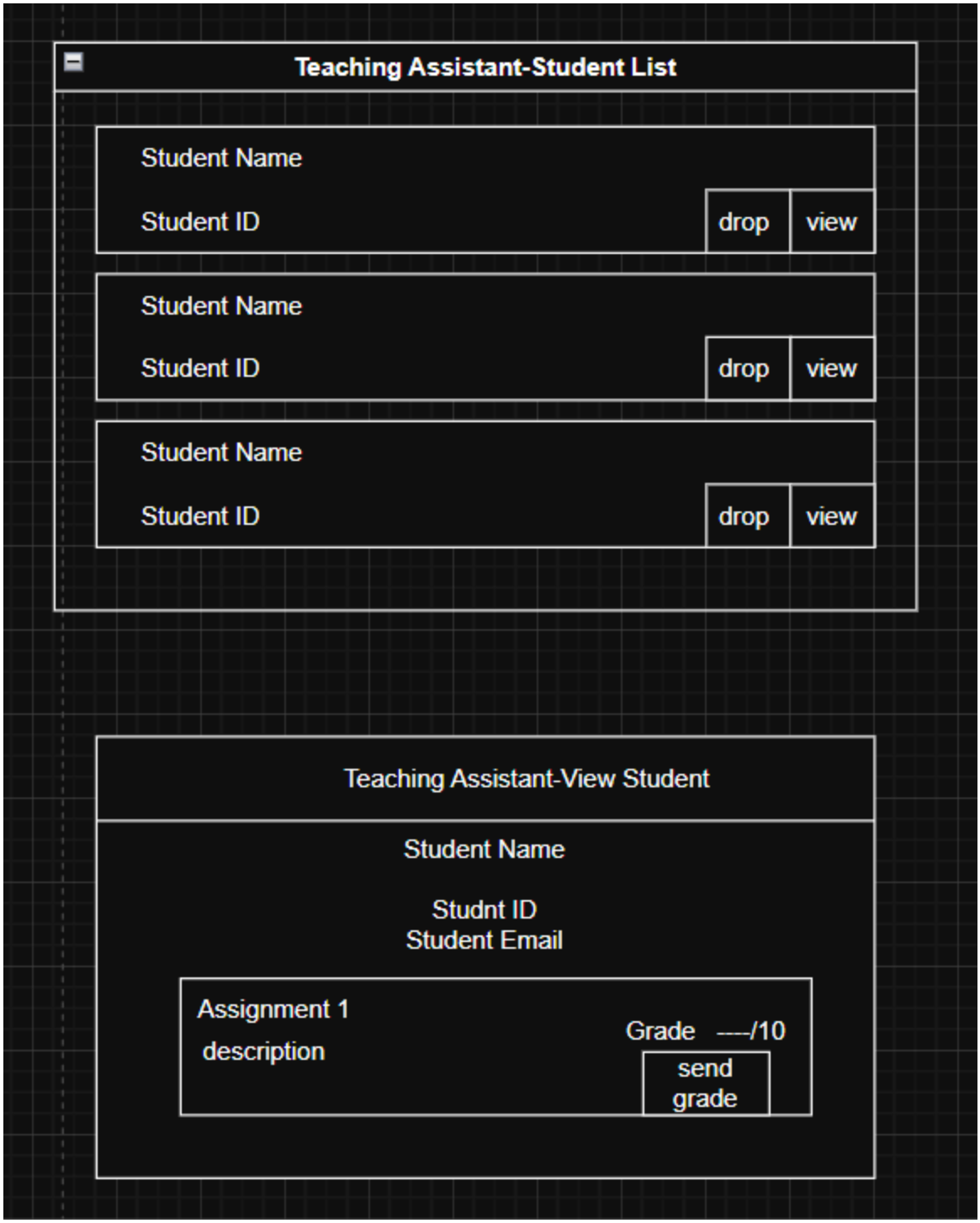

Instructor-DashBoard
Course List
Course Code- Course name
open
Course Code- Course name
open
Course Code- Course name
open

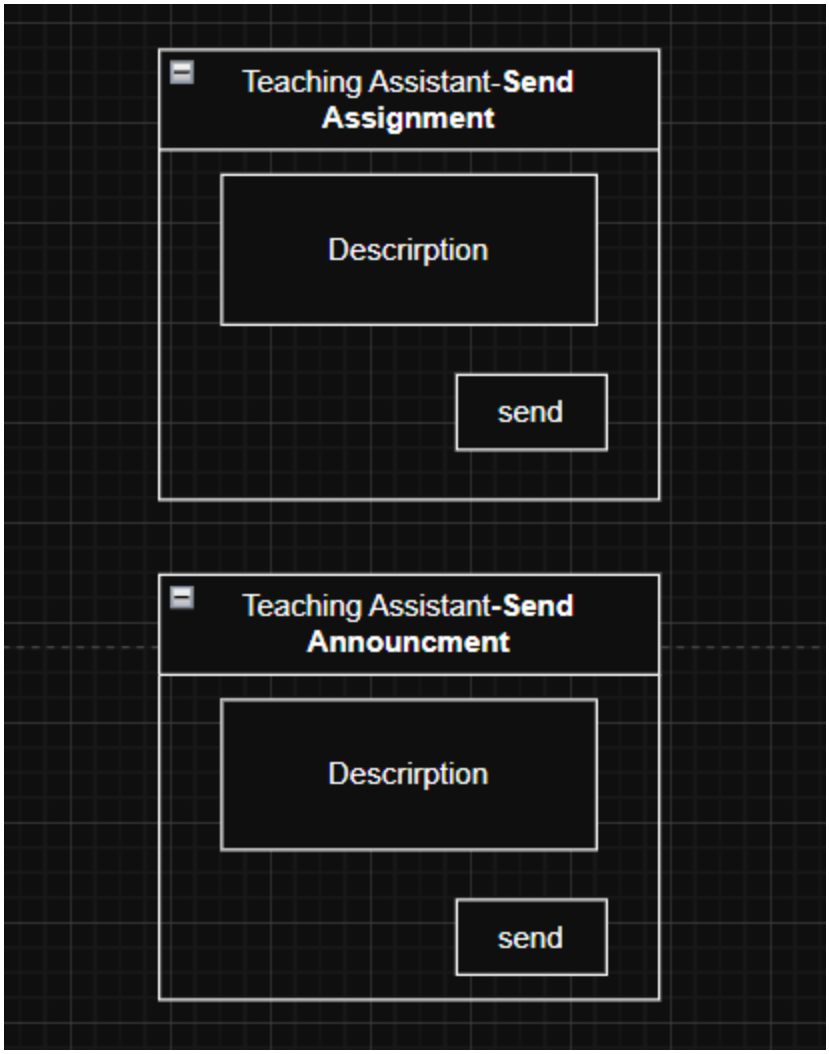

Instructor-Open Course
Course Name
View Student List
Edit Course Name
Send Assignment
Send Announcment
Instructor- Edit coursename
Enter new name
send

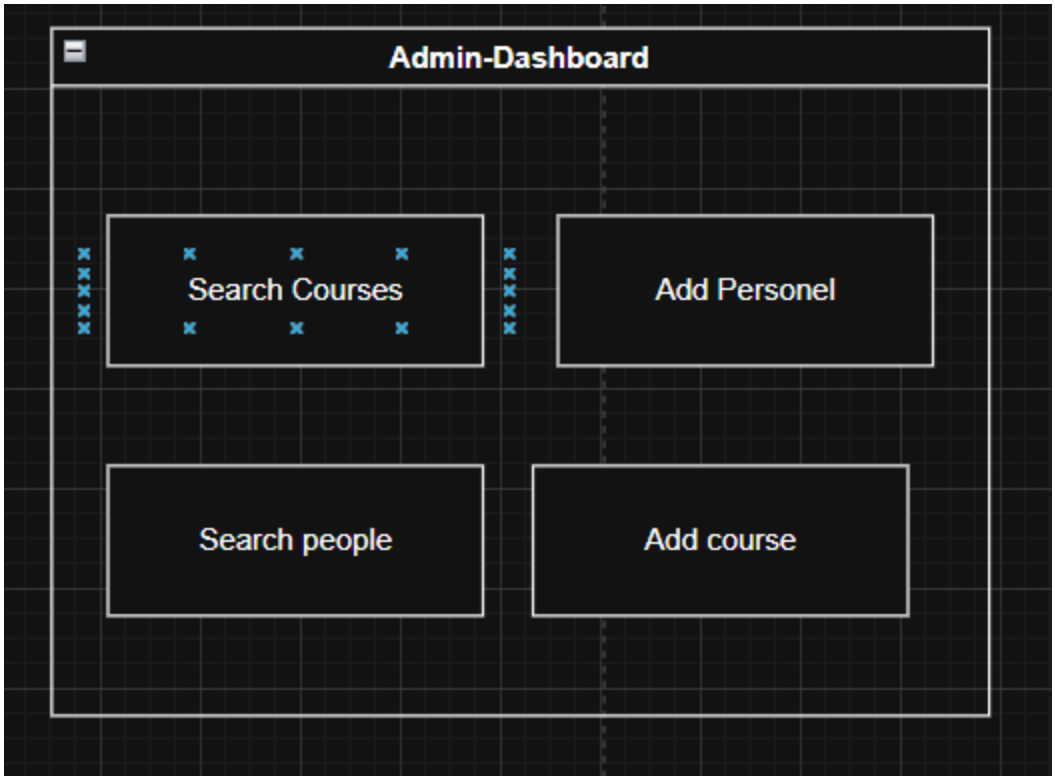

Instructor-Student List
Student Name
Student ID
drop
view
Student Name
Student ID
drop
view
Student Name
Student ID
drop
view
Instructor-View Student
Student Name
Studnt ID
Student Email
Assignment 1
description
Grade ----/10
send grade

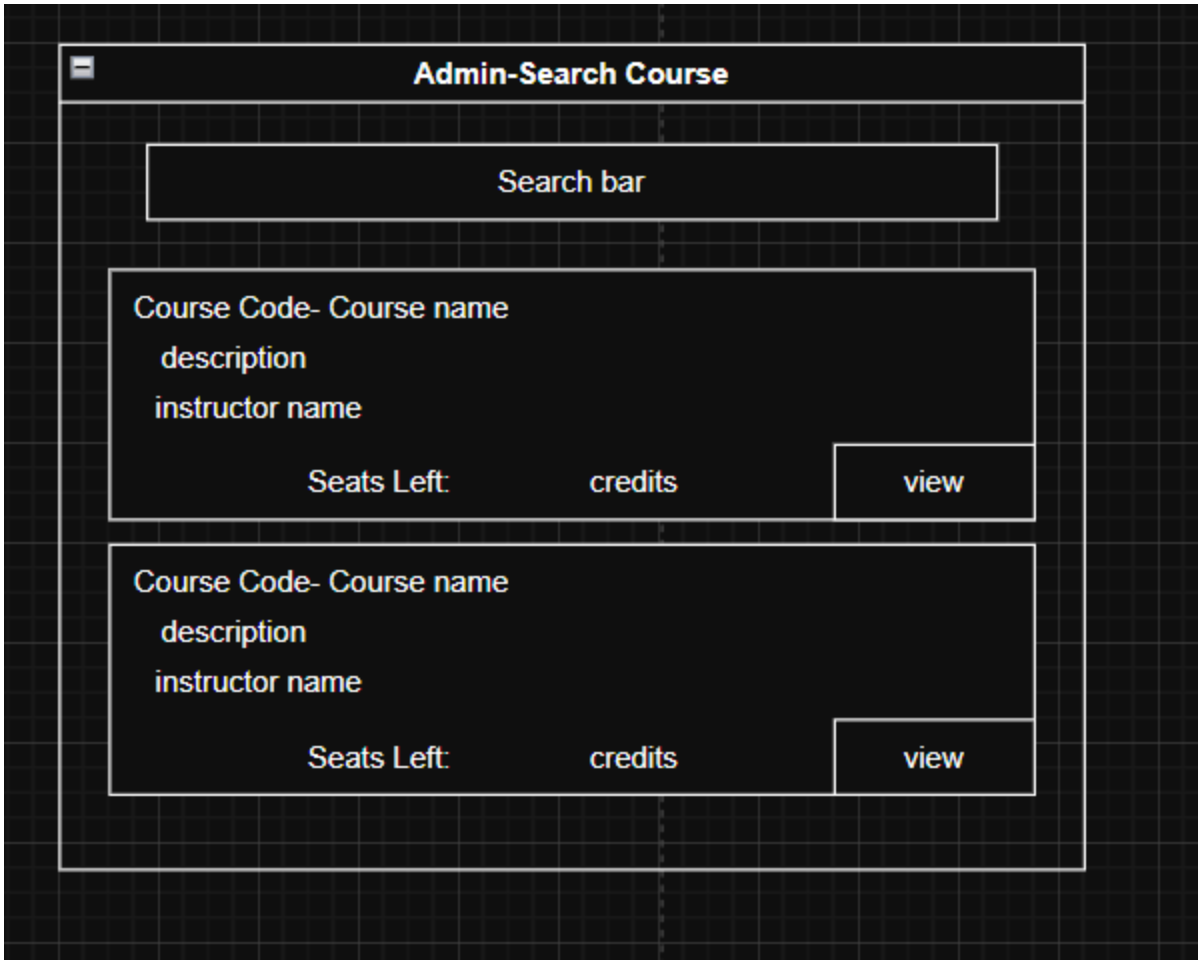

Instructor-Send Assignment
Descrirption
send
Instructor-Send Announcment
Descrirption
send

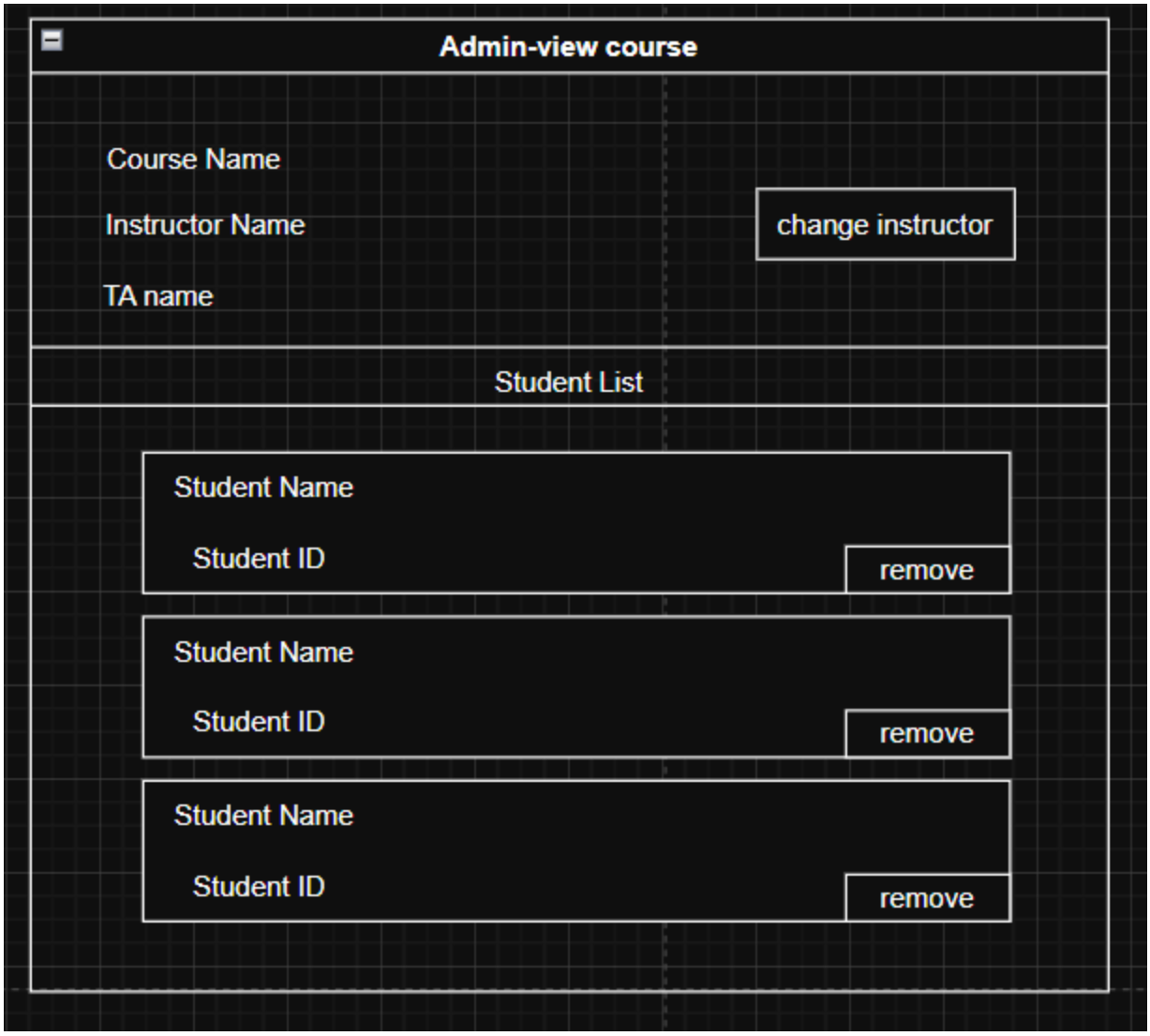

Teaching Assistant-Dashboard
View student list
Post Assignment
Post Announcment

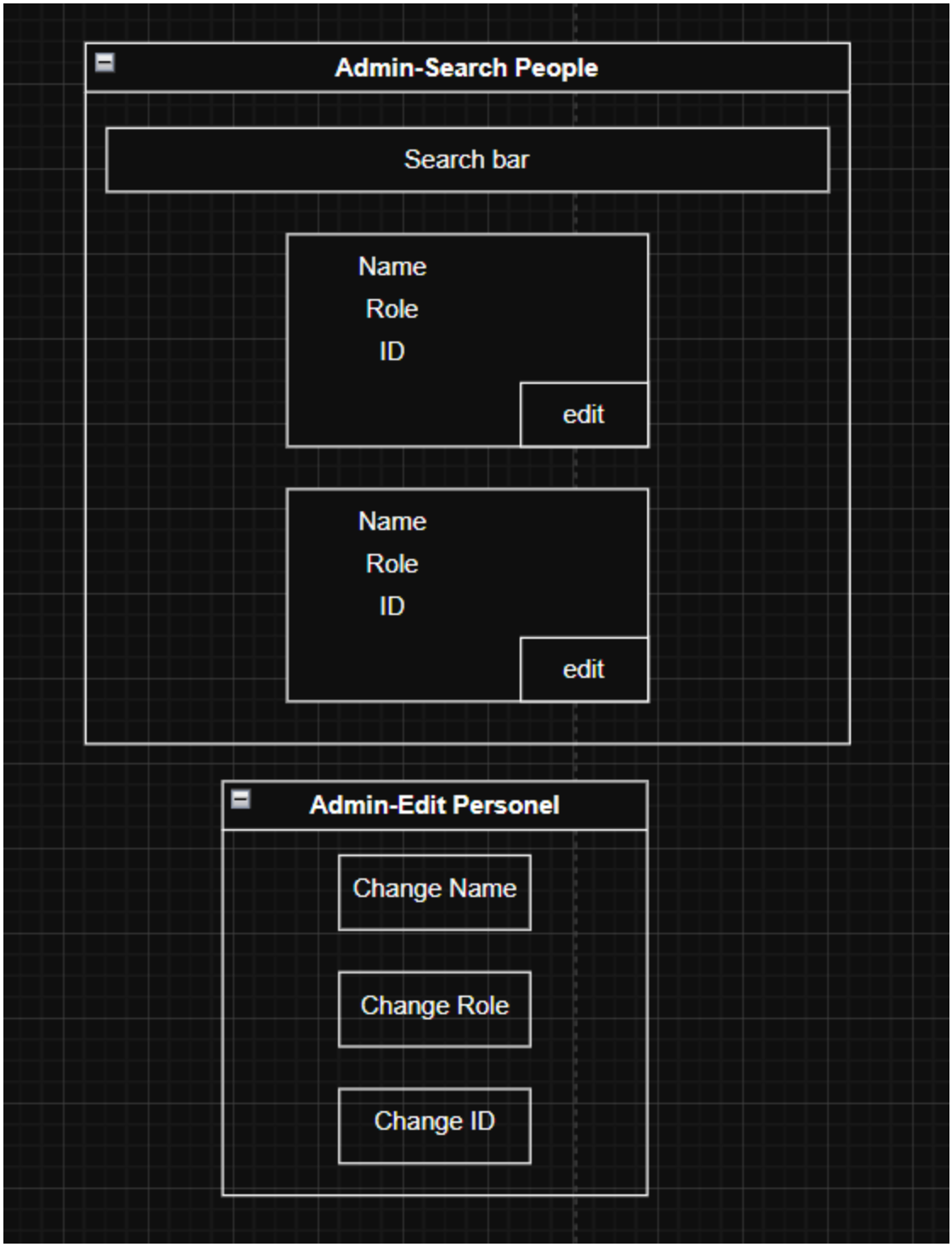

Teaching Assistant-Student List
Student Name
Student ID
drop
view
Student Name
Student ID
drop
view
Student Name
Student ID
drop
view
Teaching Assistant-View Student
Student Name
Studnt ID
Student Email
Assignment 1
description
Grade ----/10
send grade

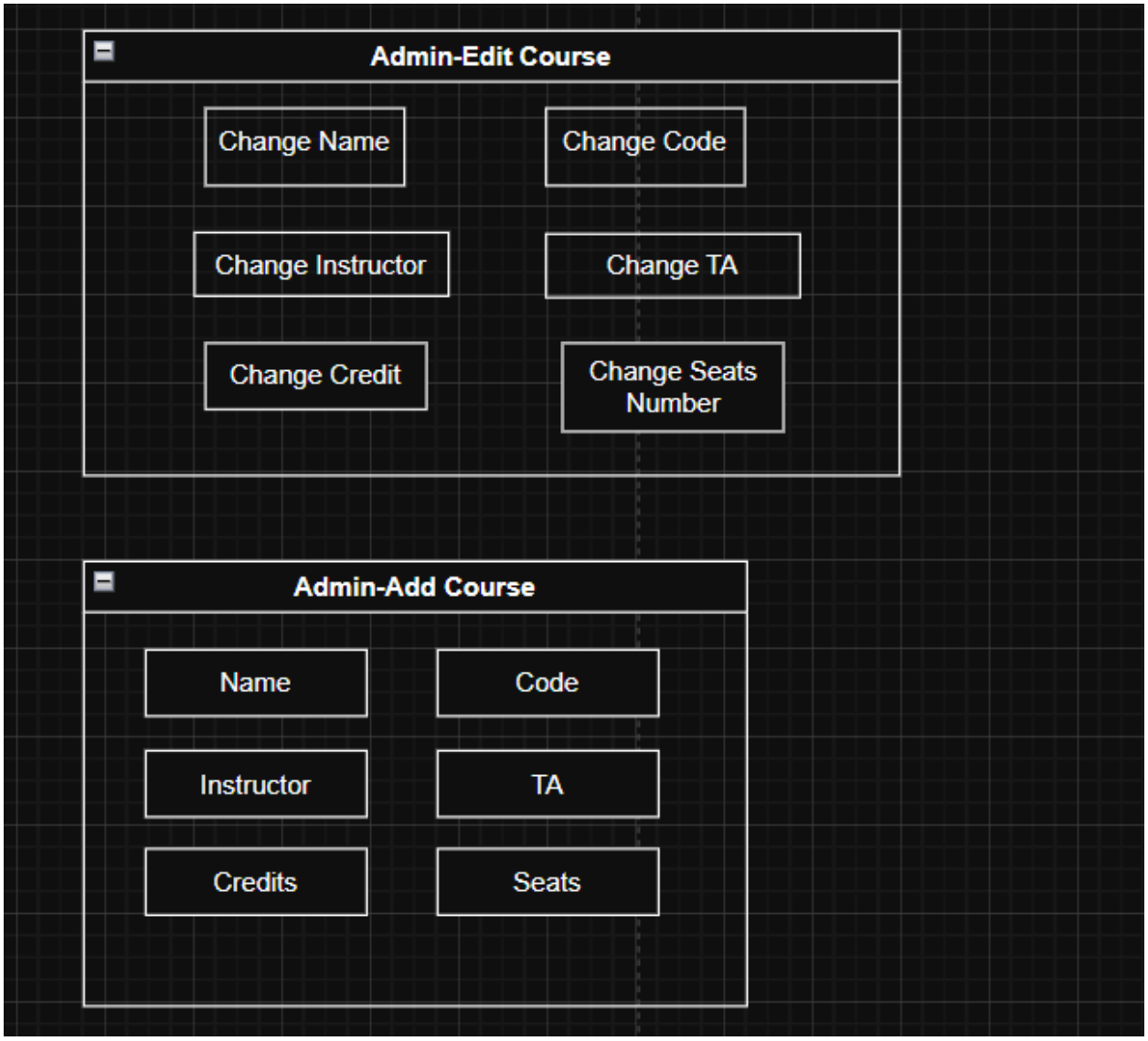

Teaching Assistant-**Send Assignment**
Descrirption
send
Teaching Assistant-**Send Announcment**
Descrirption
send

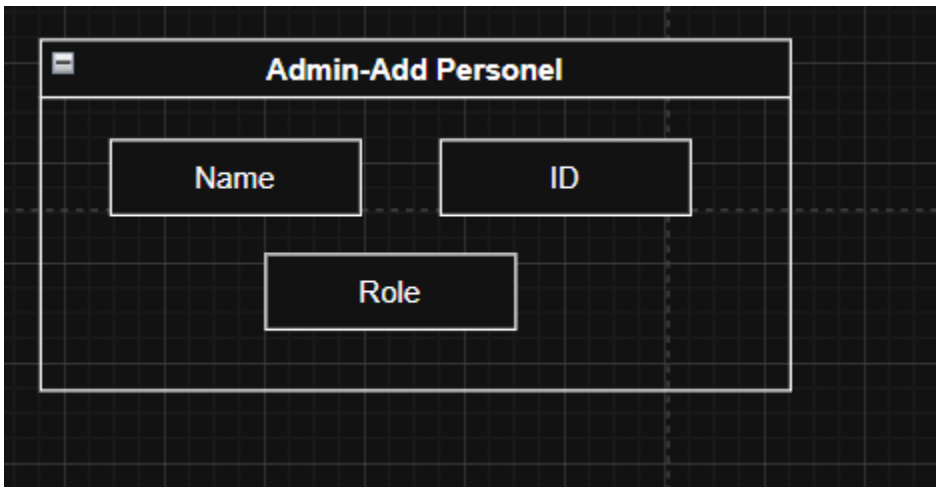

Admin-Dashboard
Search Courses
Add Personel
Search people
Add course


Admin-Search Course
Search bar
Course Code- Course name
description
instructor name
Seats Left:
credits
view
Course Code- Course name
description
instructor name
Seats Left:
credits
view


Admin-view course
Course Name
Instructor Name
change instructor
TA name
Student List
Student Name
Student ID
remove
Student Name
Student ID
remove
Student Name
Student ID
remove

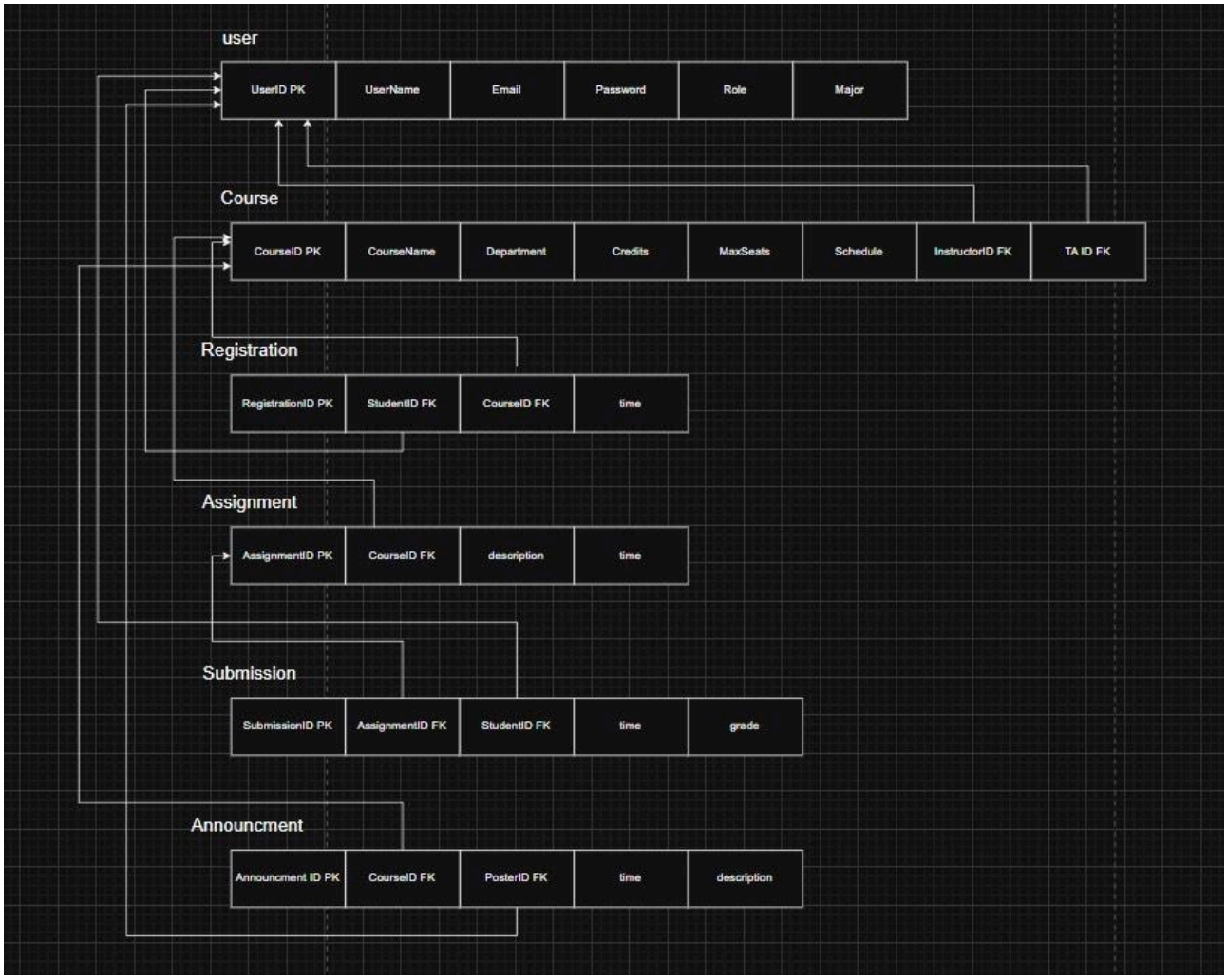

Admin-Search People
Search bar
Name
Role
ID
edit
Name
Role
ID
edit
Admin-Edit Personel
Change Name
Change Role
Change ID


Admin-Edit Course
Change Name
Change Code
Change Instructor
Change TA
Change Credit
Change Seats Number
Admin-Add Course
Name
Code
Instructor
TA
Credits
Seats


Admin-Add Personel
Name
ID
Role

4.4 Data Design

4.4.1 Database Schema/ER diagram



4.4.2 Data Storage Model

The Smart Course Registration System uses a relational SQL-based data storage model where all system data is stored in a set of normalized database tables, each responsible for a specific part of the system functionality.
The storage model :

1. Primary Keys (PK) that uniquely identify records
2. Foreign Keys to maintain relationships between tables
3. NOT NULL, UNIQUE, and CHECK constraints for data integrity
4. 1-to-Many and Many-to-Many relationships as defined in the ERD

The SCRS database contains the following tables:

1. users – stores all user accounts (student, instructor, TA, admin)
2. roles – stores predefined system roles

3. courses - course information, instructor assignment, timings and capacity
4. registrations: stores student enrollments in courses
5. assignments – stores assignments posted to each course
6. submissions – stores uploaded assignment submissions from students
7. announcements - stores posts made by instructors or TAs

8. ta_assignments - stores TA permissions and the courses they are assigned to

These tables taken together represent the complete database storage layer of SCRS and follow the SQL schema shown in section 4.4.1.

4.4.3 Data Dictionary

Roles :

Field	*Type*	*Description*
id	*INTEGER*	*Primary key, unique role identifier*
role_name	*TEXT*	*Role name (student / ta / instructor / admin)*

User :

Field	*Type*	*Description*
UserID	*INTEGER*	*PK ,unique user ID*
UserName	*TEXT*	*Full name of the user*
Email	*TEXT*	*University email (unique)*
Password	*TEXT*	*Encrypted password*
Role	*TEXT*	*User role in the system (Student, Instructor, TA,*

		Admin). Determines the permissions and access level of the user.
Major	*TEXT*	*Student's major or field of study.* ***NULL*** *for users who are not students.*

Courses:

Field	*Type*	*Description*
CourseID	*INTEGER*	*Primary key, unique course ID*
CourseName	*TEXT*	*Full course name*
Department	*TEXT*	*Course department*
Credits	*INTEGER*	*Number of credit hours*
MaxSeats	*INTEGER*	*Maximum seats for the course*
Schedule	*TEXT*	
InstructorId	*TEXT*	*FK → users(id): instructor teaching the course*
TAId	*INTEGER*	*FK → users(id): TA assisting the course*

Registrations:

Field	*Type*	*Description*
RegistrationId	*INTEGER*	*PK*
StudentId	*INTEGER*	*FK → users(id), the student enrolled*
CourseId	*INTEGER*	*FK → courses(id), the course enrolled*
time	*DATETIME*	*Timestamp of enrollment*

Assignments:

Field	*Type*	*Description*
Assignmentid	*INTEGER*	*PK*
CourseId	*INTEGER*	*FK → courses(id)*
description	*TEXT*	*Assignment description/content*
time	*DATETIME*	*Assignment time*

Submissions:

Field	*Type*	*Description*
SubmissionsId	*INTEGER*	*PK*
AssignmentId	*INTEGER*	*FK → assignments(id)*
StudentId	*INTEGER*	*FK → users(id)*
time	*DATETIME*	*Submissions time*
grade	*REAL*	*Grade assigned to submission (nullable)*

Announcements:

Field	*Type*	*Description*
AnnouncementId	*INTEGER*	*PK*
CourseId	*INTEGER*	*FK → courses(id)*
PosterID	*INTEGER*	*FK → users(id), instructor or TA*
time	*DATETIME*	*Time of announcement*
description	*TEXT*	*Announcements description*

5. Conclusion

5.1 Summary of Design Phase

This design document has successfully outlined the comprehensive architectural and detailed design for the Smart Course Registration System (SCRS). The system will be developed as a monolithic Flask application, adhering to the Model-View-Controller (MVC) architectural pattern to ensure a clear separation of concerns, maintainability, and scalability.