College of Natural and Computational Sciences (CNCS)

School of Information Science (SIS)

Student Project Tracking System for School of Information Science (SIS)

Industrial Project II

In partial Fulfillment of Requirements for B.Sc. Degree in Information Systems

Submitted to: School of Information Science

Addis Ababa University

June 2023

Group members

Name	ID
1. Andualem Sebsbe	UGR/7326/12
2. Gelila Tesfaye Gebru	UGR/0903/12
3. Hanan Hussien	UGR/1778/12
4. Huluager Alehegn	UGR/1112/12
5. Kasanesh Ayalew	UGR/8580/12

Information Systems Industrial Project II [INSY4122]

Advisor: Dagmawit Mohammed

Examination Board

Adviser name	**Date**	**Signature**
Examiner name	**Date**	**Signature**
Examiner name	**Date**	**Signature**

Table Content

List of Figures

List of Tables

Acknowledgement

First we would like to express our sincere gratitude to our advisors Dagmawit Mohammed for her guidance, support, and collaboration helped to shape the direction of the project and turn it into a valuable learning experience.

Thanks to our friends for whose support and encouragement goes beyond the successful completion of this project.

We would also like to extend our appreciation to coordinator and to all instructors in the school of information systems for their contribution to our achievements.

Finally we would like to thank school of Information Science employees for their cooperation.

Chapter Five

Object oriented Design

5.1 Review of the first phase of the project

In the first chapter of this project, we have conducted an analysis of the organization's background and history, specific problem that the organization is facing in relation to the methods and ways of managing different student's project in the school, and developed a clear statement of the problem to guide the project. To address the problem, we have defined specific objectives that our system aims to achieve, including measurable outcomes and time lines. We have also established the scope of the project and a feasibility study to assess the viability of the proposed system, considering technical, economic, operational and economic factors. Generally we have addressed the foundation for the new system and provided a clear roadmap for moving forward.

In the second chapter, we have covered the operations of the business areas and its current issues, and the functional and non-functional requirements of the proposed system.

In the third chapter we have developed the user interface (UI) of the new system, use case modeling to identify the various actors, use cases, and scenarios that will be involved in the system, and UML diagrams such as analysis class diagrams to model the system entities and relationships, sequence diagrams to model the system's behavior and interactions between different objects and classes This has helped us to understand the system requirements from the user's perspective and identify the key functionalities that the system needs to provide.

Functional Requirements for Student Project Tracking System:

1. **User Authentication & Registration:** The system should support authentication by means of passwords allowing the users to have access to personalized areas once

identified successfully those users can also register with information like an username name, email address, course number etc..

2. **Submit Title:** Students should be able to submit their project titles through the system. The system should validate the input and ensure that the title is not already taken.

3. **Project Tracking & Management**: Ability to track existing projects assigned to students and faculties at any given point of time; ability to create a project plan outline; ability to create task plans including deliverables & deadlines associated with each task; ability to set milestones & associated goals; ability provide timely notifications on deadlines etc… .

4. The system should allow users with appropriate access permission mark a project has been completed by marking it with a check or other symbol.

5. It should provide feature for past project data search.

6. It should enable lecturers/supervisors assign tasks to students/researchers by due dates.

7. The system should generate project records as reports such as a list of projects running late or according to certain criteria like discipline wise etc.

8. It should provide feature for past project data search.

9.. The system should be able to store the project title, the deadline for completion, tasks assigned to determinate people, researcher's name and in-charge staff.

10. Ability to report on task completion status in real-time.

11. Automated notifications for students about upcoming deadlines.

5.2. Introduction

This section provides the design of the new system. Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. The system requirement document is coming from the previous phase and develops the specific technical details required for the system such as the architecture of the system, system decomposition, component modeling, deployment, and user interface designs.

5.3. Systems architecture design

The proposed system is expected to replace the existing manual system by an automated system in all facets. The architecture of a system explains how distributed systems are actually organized by considering where the software components are placed. It is made by keeping business logic and needs in mind.

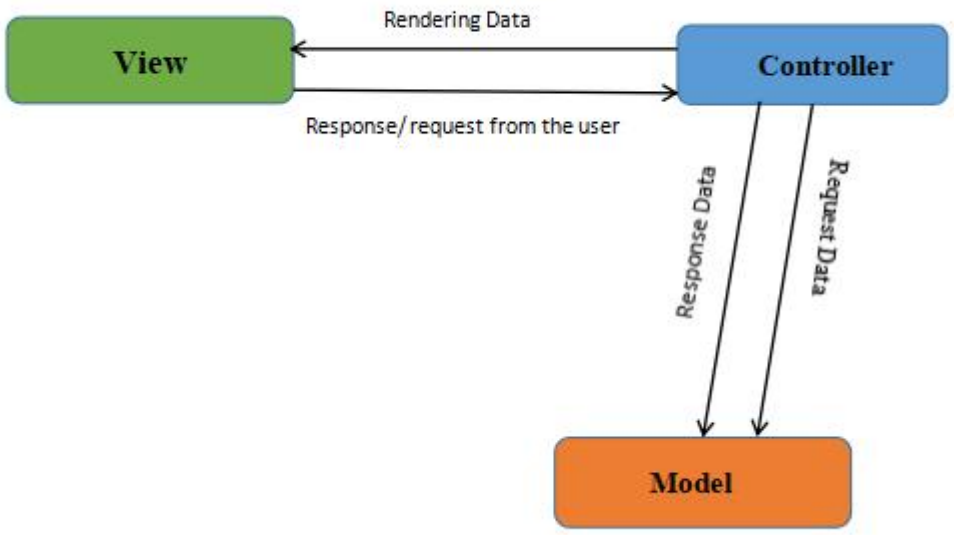


Figure 1 System Architecture

Our system is primarily focused on rendering dynamic views and handling user input and the technology and the framework that we used to develop the system is Next.js and Node.js as which allows for a separation of concerns between the various components of the system. With this regard MVC (Model-View-Controller) architecture is well-suited to our system, (i.e. The Student Project Tracking System)

because it separates the application logic into three distinct components: model, view, and controller. This separation of concerns makes it easier to maintain, extend, and test the system.

- **The model component** is used to represent the data and business logic of the system. This include information such as
  - ✓ Project
  - ✓ User account
  - ✓ Instructor
  - ✓ Project progress
  - ✓ Project evaluation
  - ✓ Student
  - ✓ coordinator
- **The view componen**t represents the user interface, which allows students, instructors and coordinators to interact with the system. This would include forms for submitting project titles, status updates, view progress, and other relevant information.

The following are user interface layers of SPTS.

1. Main menu page
2. Login page
3. Coordinator homepage
4. Instructor page
5. Student home page/ dashboard
6. User registration page
7. Upload project page
8. Title submission page
9. Group form page
10. Search project page
11. Assign adviser page
12. Approve title page
13. Approve project page
14. Add users page
15. Update user page
16. Delete user page
17. Feedback/ comment page
18. View progress page

- **The controller component** handles the flow of data between the model and view components, as well as processing user input and updating the model accordingly.

Compared to other system architectures, such as a monolithic architecture, in which all the application logic is contained in a single codebase, making it difficult to modify or scale the system, MVC architecture, allows for a more modular approach, making it easier to add new features or functionality to the system as needed.

Additionally, the separation of concerns makes it easier to test each component independently, reducing the likelihood of bugs or errors.

The following are the lists of controllers for our project.

1. Login
2. User registration
3. Update user
4. Delete user
5. Upload project
6. Comment project
7. Approve title
8. Submit title
9. Search project
10. Assign adviser
11. Form group

5.4 class diagram modelling

5.4.1 Class diagram

Student project tracking system describes the structure of a student project tracking system classes, their attributes, operations (or methods), and the relationships among objects. The main classes of the system are:-

- ✓ Student
- ✓ Course project
- ✓ Final year project
- ✓ Evaluation
- ✓ Calendar
- ✓ Instructor
- ✓ Coordinator
- ✓ User account
- ✓ Adviser and examiner

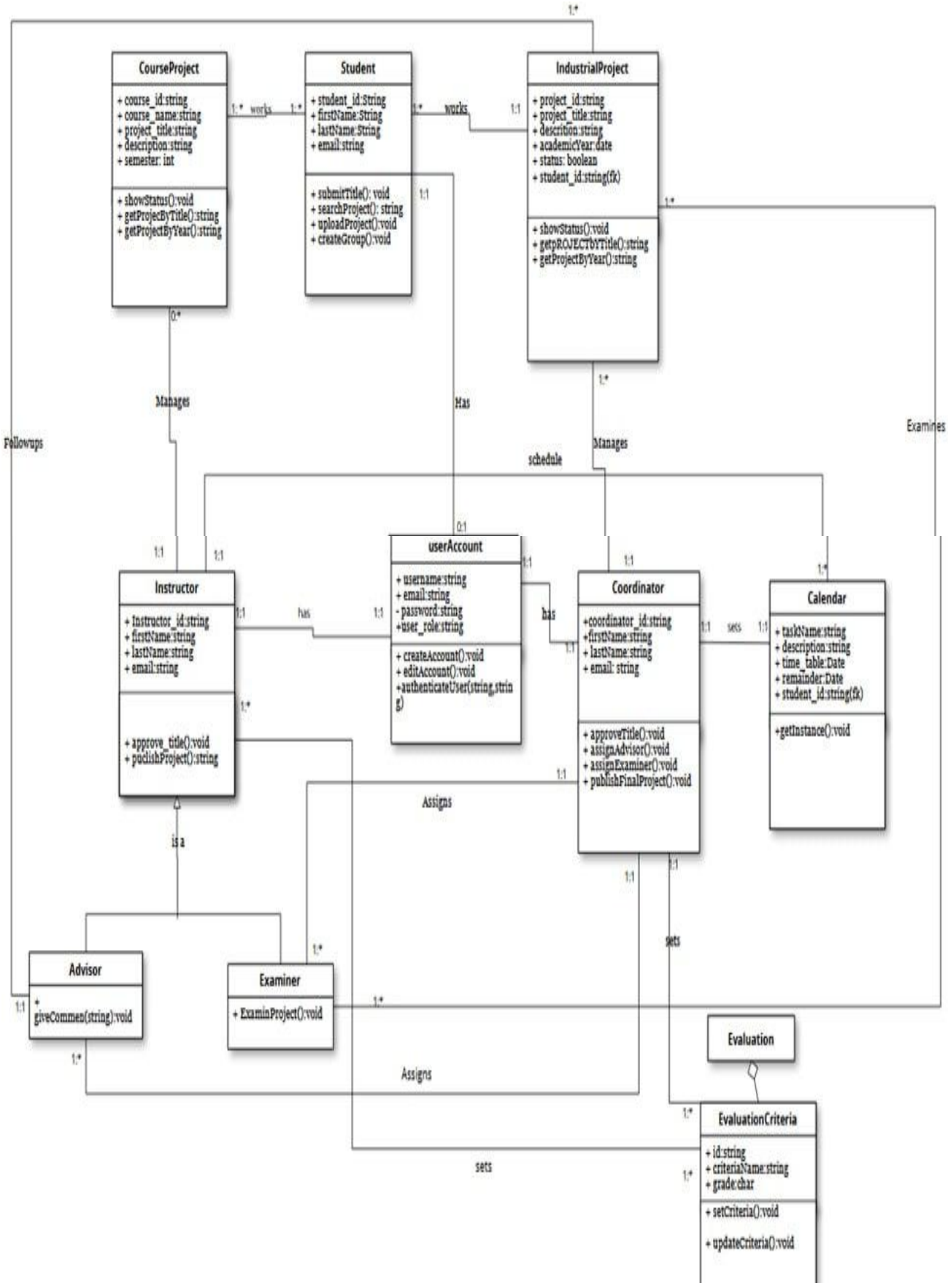
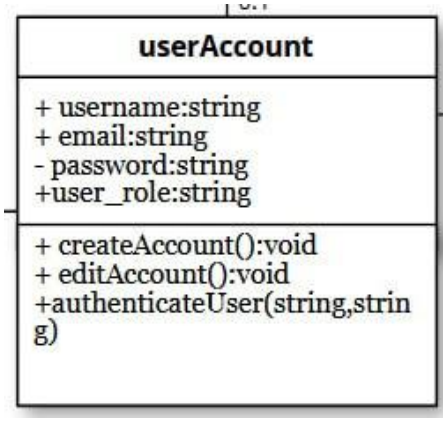


Figure 2. Class Diagram

5.4.2 Class Descriptions

1. Class name: - User_Account

Description: - this class is used to get the user information from the database and is also used for authenticating the users.



Attributes:

- **Username**

  Description: The username is a unique identifier that is assigned to each user account. It is used to log in to the system and to identify the user in the system.

  - ✓ Data type string
  - ✓ Visibility public
  - ✓ Size 50

- **Password**
  - ✓ Description: it is a secret code that is used to authenticate the user and to protect the user's account from unauthorized access.
  - ✓ Data type string
  - ✓ Visibility private
  - ✓ Size 16
- **User role: -** it is used to determine the level of access and permissions that the user has within the system, i.e. the coordinator have its own access to different features and functions of the system that cannot be accessed by the student. Data type is string.
  - ✓ Data type string
  - ✓ Visibility public
  - ✓ Size 20

Methods:-

- **AuthenticateUser()**

Description: - this method is used to authenticate a particular user who has provided the login credentials and wishes to login in the system. It checks the credentials and roles from the database.

- ✓ Visibility public
- ✓ Return type void

Edit_account()

Description: - this method is used to edit account such us password.

- ✓ Return type void
- ✓ Visibility public

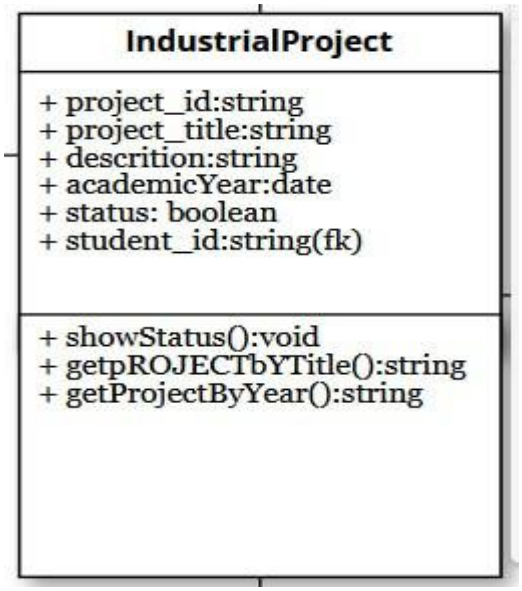
- **Create_accoun**t()

Description: - this method will save the details provided in the database. The user login will be functional immediately after the execution of this method.

- ✓ Visibility public
- ✓ Return type void

2. Class name: industrial Project class

Description: This class is used to process all information regarding the industrial projects



Attributes:

- **project Id**:-

Description: this attribute uniquely identifies the project.

- ✓ Data type string
- ✓ Visibility public

- **project title**

Description: this attribute is used to search and retrieve projects.

- ✓ Data type string
- ✓ Visibility public

- **Project_Description**

Description: this attribute is used to show the overview of the project

- ✓ Data type string
- ✓ Visibility public

- Academic_year

  Description: - is used to identify the year in which the project is done.

  - ✓ Data type string
  - ✓ Visibility public

- Student_id

Description: this attribute references the student id in the student collection.

- ✓ Data type string
- ✓ Visibility public

Methods:-

GetProjectByTitle()

Description: - This role populates the details of a selected project.

- ✓ Return type list<Projects>
- ✓ Visibility: public(+)

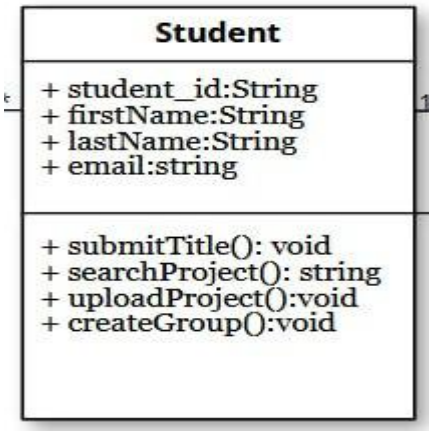
- **GetStatus()**

  Description:-this method used to see the active projects.

  - ✓ Return type void
  - ✓ Visibility public

2. **Class name:-Student**

This class used to access information about a student who is enrolled in the program. They have attributes such as student ID, name, and email address.



Attributes

- **student_id:** it uniquely identifies student.
  - ✓ Visibility public
  - ✓ Type string
  - ✓ Size 11
- **first name and last name**
  - ✓ Type string(30)
  - ✓ Visibility public
- **Email**
  - ✓ Type string(30)
  - ✓ Visibility public

Methods:-

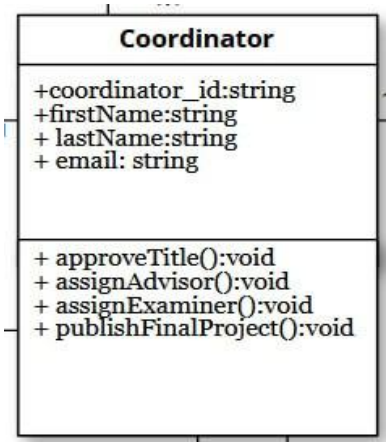
- **Submit_title()**

Description: this method allows the students to submit project title to the system.

  - ✓ Return type void
  - ✓ Visibility public
- **ViewProject()**
  - ✓ Description: this method used to allow the user to view project details
  - ✓ Return type: list<Project>
  - ✓ Visibility: public
- Submitprojects()
  - ✓ Description: this method is used to submit either midterm or final project to the system.

- ✓ Visibility public
- ✓ Return type void

3. **Coordinator**: this is mainly to access the final year project coordinators information.



Attributes: -

- **coordinator id**

  Description: This attribute used to uniquely identify a record.

  - ✓ Data type string
  - ✓ Visibility public
  - ✓ Size 20
- **First_name:-**
  - ✓ Data type string(20)
  - ✓ Visibility public
- **Last_name**
  - ✓ Data type string(20)
  - ✓ Visibility public
- **Email**
  - ✓ Data type string(30)
  - ✓ Visibility public

Methods

- **Assign_adviser ():-**

Description: - this method is used to assign adviser for a final year (industrial - project) from the instructors.

- ✓ Data type string
- ✓ Visibility public

- **Approve_title ()**: this method allows the coordinator to approve a project once it has been submitted by the student.

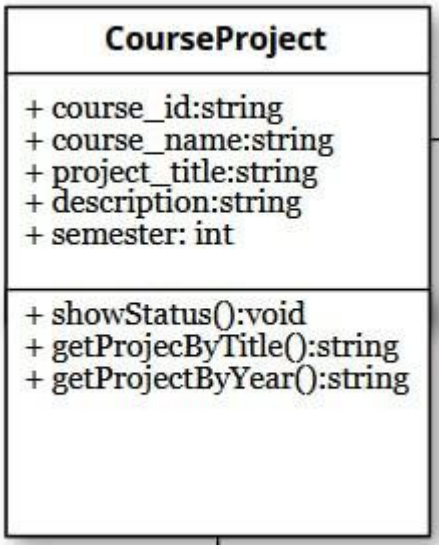
- ✓ Data type string
- ✓ Visibility public

- **Assign_examiner():** helps the coordinator to select instructors and assign as an adviser for a project.
- ✓

- **Publish_project ():** this method is used to publish the final project.

4. **Class name: Course Project class**

Description: this class is used to track different course projects.



Attributes: -

- **Course_id**
  - ✓ Data type string
  - ✓ Visibility public
- **Course_name**
  - ✓ Date type string
  - ✓ Visibility public
- **Semester**
  - ✓ Date type integer
  - ✓ Visibility public

Methods

GetProjectByTitle (): string

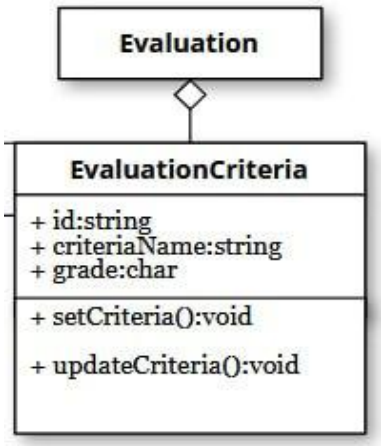
Description: - this role populates the details of a selected project.

- ✓ Return type list<Course projects>
- ✓ Visibility public(+)

- **GetStatus()**
  - ✓ Description: this method used to see the active projects.
  - ✓ Return type void
  - ✓ Visibility public

5. **Class name :Evaluation**

Description: the evaluation class is an aggregate of evaluation criteria because it represents a collection of individual evaluation criteria that are combined to form a comprehensive evaluation of the software system.



Attributes :-

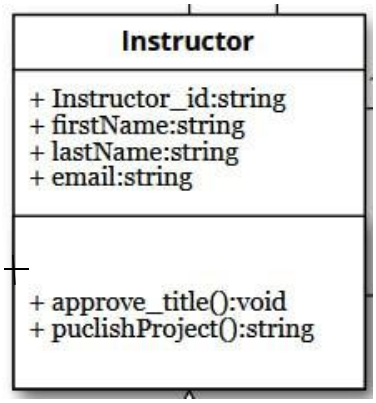
- Id:
  - ✓ Description: to identify each evaluation criteria's.
  - ✓ Data type: string
  - ✓ Visibility: public
- Criteria_name:
  - ✓ Description: the specific name or label given to a particular evaluation criterion.
  - ✓ Data type: string
  - ✓ Visibility: public
- Grade:

- ✓ Description:user do store the score or rating assigned to a particular evaluation result.
- ✓ Data type: char
- ✓ Visibility: public

Methods :

- Set_criteria():
  - ✓ Description: this method is used to set an evaluation criteria.
  - ✓ Return type: void
  - ✓ Visibility: public
- Update_criteria

6. **Class name:-Instructor**



Attributes

- **Instructor_id**

  Description: uniquely identify each Instructor records.
  - ✓ Type string
  - ✓ Visibility public
- **Full_name**

  Description: this attribute is used to store the name of the instructor.
  - ✓ Type string
  - ✓ Visibility public
- **Email**
  - ✓ Type string
  - ✓ Visibility public

Methods

- **Examine_project ()**

  Description: this method is used to evaluate the course project.

  ✓ Return type void

  ✓ Visibility public

- **Approve_project ()**

  Description: this method allows the instructor to approve course projects.

  ✓ Return type string

  ✓ Visibility public

Follow_up_projects ()

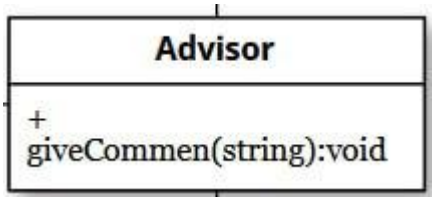
Description: this method allows the instructor to manage and follow the progress of the project activities.

✓ Return type void

✓ Visibility public

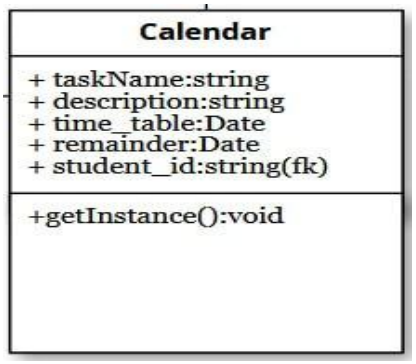
7. **Class name: Adviser**

Description: This class is a child class inherited from the instructor class.



8. **Class name: Examiner,** this class also a child class inherited from the instructor class.

9. **Class name:- calendar**



Description: It provides a wide range of functionality for performing common date and time operations, and can be used in a variety of applications, such as scheduling systems, reminders, and time-tracking applications

Attributes:

- **Task_name:** this attribute used to show the task assigned in the given deadline.
  - ✓ Data Type string
  - ✓ Visibility public
- **Start_date and end_date**
  - ✓ Date type Date
  - ✓ Visibility public
- **Description**: shows the description of the task.
- **Remainder:** to notify the deadline.
  - ✓ Data type: date
  - ✓ visibility: public

5.5. Relational persistent model

A relational persistent model is a way of organizing data in a software application using a relational database management system that provides persistent storage capabilities, allowing data to be stored and accessed efficiently and reliably over time. In relational persistent modeling, normalization is a crucial step in designing a database schema. The process helps to eliminate redundant data, reduce data anomalies, and improve data consistency and integrity. The resulting tables are then linked together through the use of foreign keys, which help to maintain data integrity and consistency.

Hence were used a MongoDB database which is a non-relational database management system that stores data in a document-oriented format, rather than table-based structure of relational databases. So, in a NoSQL database normalization is not required because it uses a document-oriented data model that allows for flexible and dynamic schema design. In MongoDB, data is stored in documents in a JSON-like format, which can contain nested structures, arrays, and key-value pairs, making it easy to represent complex data structures. This flexibility allows for easy and efficient handling of complex data structures and relationships.

Since MongoDB does not rely on the tabular structure of relational databases, we don't need to draw a relational persistent model, Instead, Normalization is the process of organizing data in a database to reduce redundancy and dependency, which helps to improve data consistency, reduce data anomalies, and.

In order to draw our systems non-relational persistent model we had followed the following steps

- Identify the entities and relationships in our system.
- Design the document schema
- Define the indexes
- Optimize the schema for query performance: Since MongoDB is a schema less database, it is important to optimize the document schema for query performance. Including, Deformalizing data, embedding related data, using reference documents to represent complex relationships.
- Draw the diagram: Once we designed the document schema and defined the indexes, then we draw the diagram using moon modeler tool.

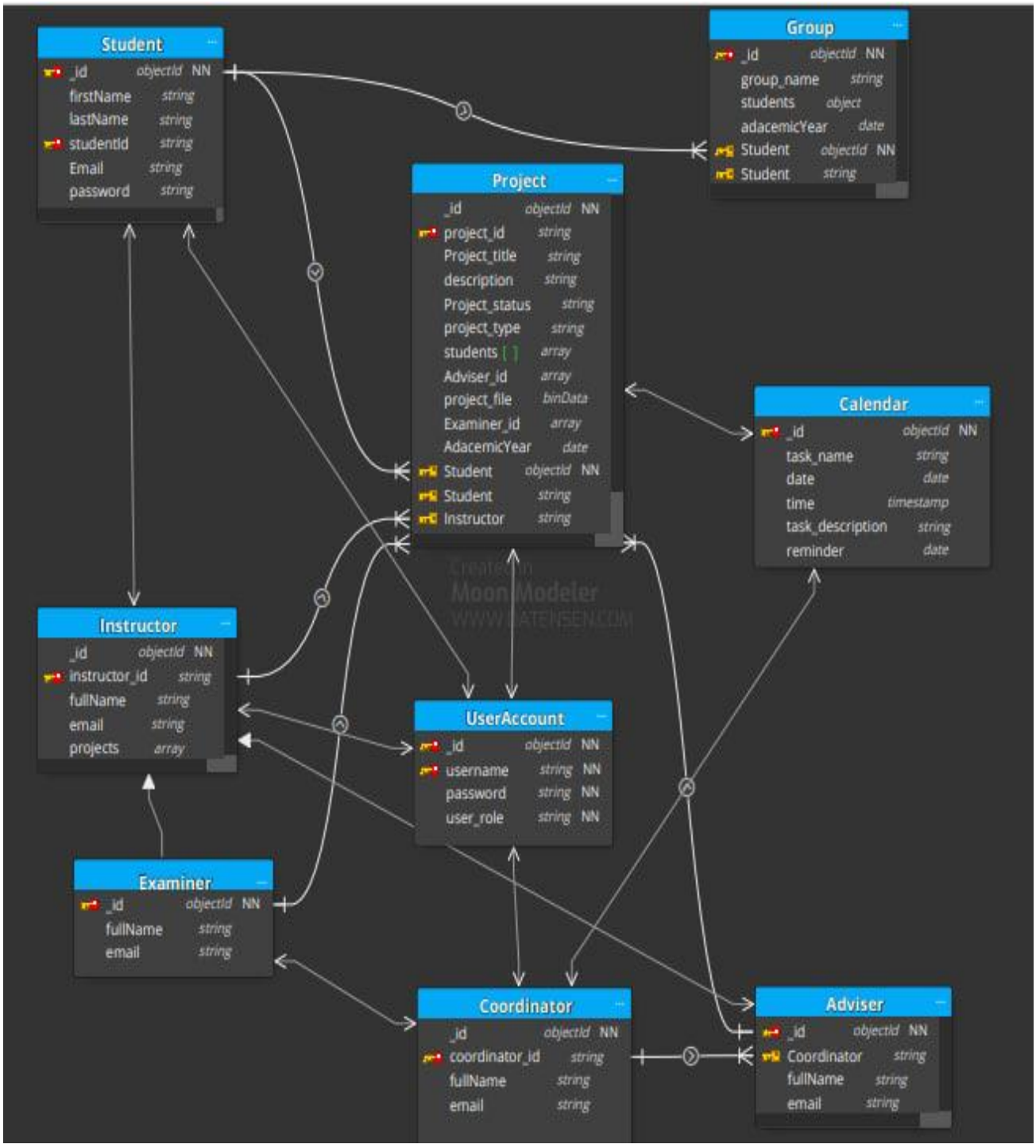


Figure 3. Non-relational persistent model

In this model, each entity (Coordinator, Student, Instructor, Adviser, Examiner, Project, and Calendar) is represented as a separate document or collection in our mongo DB database. The "projects" field in each entity is an array that contains references to the projects that the entity is associated with. The "calendar" field in the Project entity is also an array that contains references to the Calendar events associated with the project.

5.5 Component Diagram

A UML component diagram is a type of diagram in Unified Modeling language (UML) that shows the structure of a software system by representing the components and their dependencies. Components in a UML component diagram are software modules, libraries, executables, or other parts of a system that can be independently deployed and replaced.

A UML component diagram typically includes components, ports, interfaces, connectors, and relationships between them. The components are shown as rectangles with the name inside, and the ports are depicted as small squares at the edges of the components, which serve as communication endpoints for the components.

The relationships between components are depicted as connectors, which can be either simple lines or more complex shapes such as arrows or diamonds. UML component diagrams are helpful in understanding the architecture of complex systems and in designing new software systems.

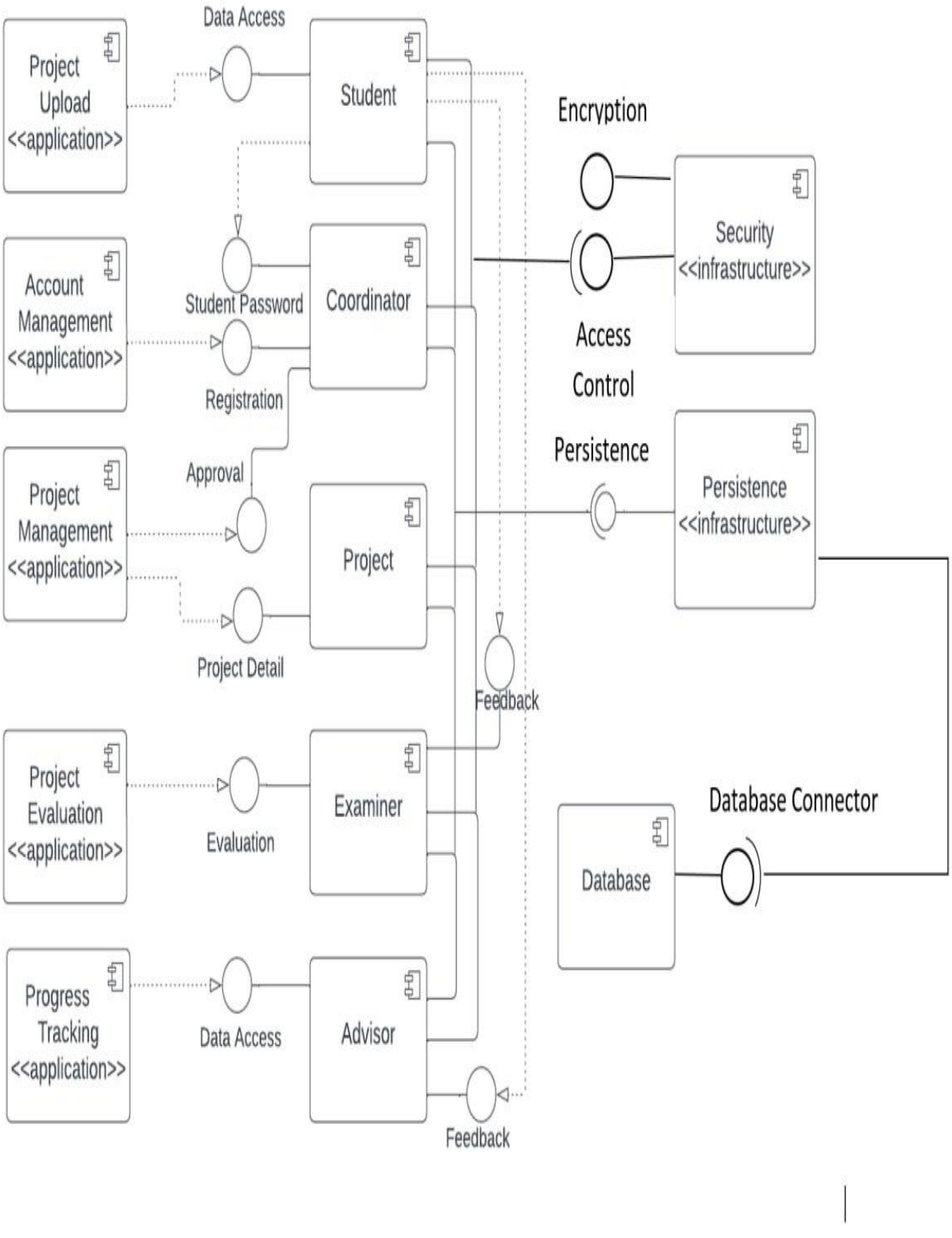


Figure 4. Component Diagram

5.6 Deployment diagram

The deployment diagram is a type of UML diagram that shows the physical deployment of software components on hardware nodes. It consists of nodes, which represent the hardware elements, and components, which represent the software elements. The nodes can be physical devices or virtual machines that are used to host the components. The components can be software applications, web services.

The deployment diagram for the student project tracking system shows the distribution of the processes using the system's physical architecture. It shows the relationships between software and hardware (physical architecture) that the project tracking process. This system includes three separate nodes as shown in the following diagram.

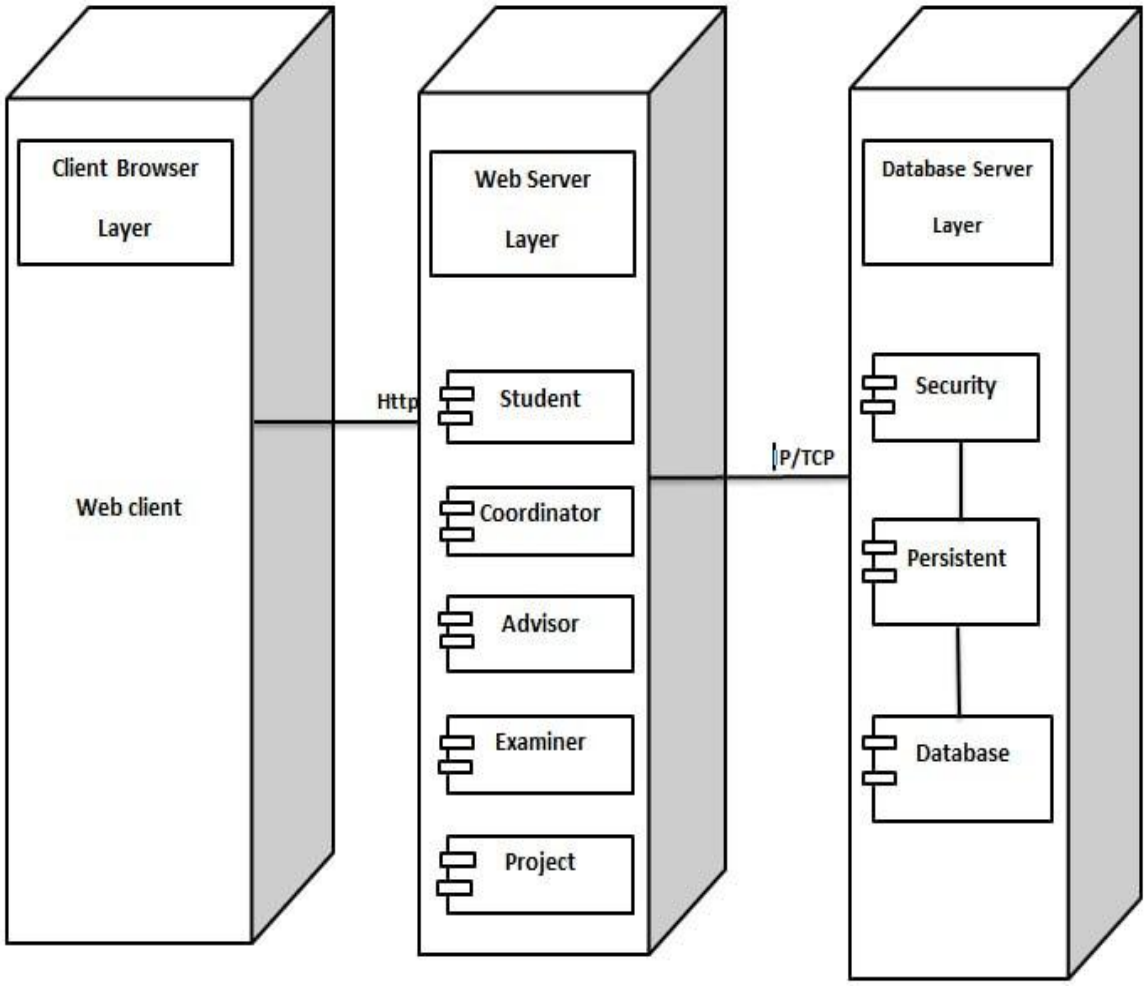


Figure 5. Deployment Diagram

5.7 User interface

5.7.1 User Interface Flow Diagram

User flow diagram is used primarily by product and UX teams to figure out the flow of a website or application after we have thought about the users experience and user needs. To best understand these needs and the experience we want our customers to have, it is important to map and visualize them. The following diagram shows user interface flow diagram.

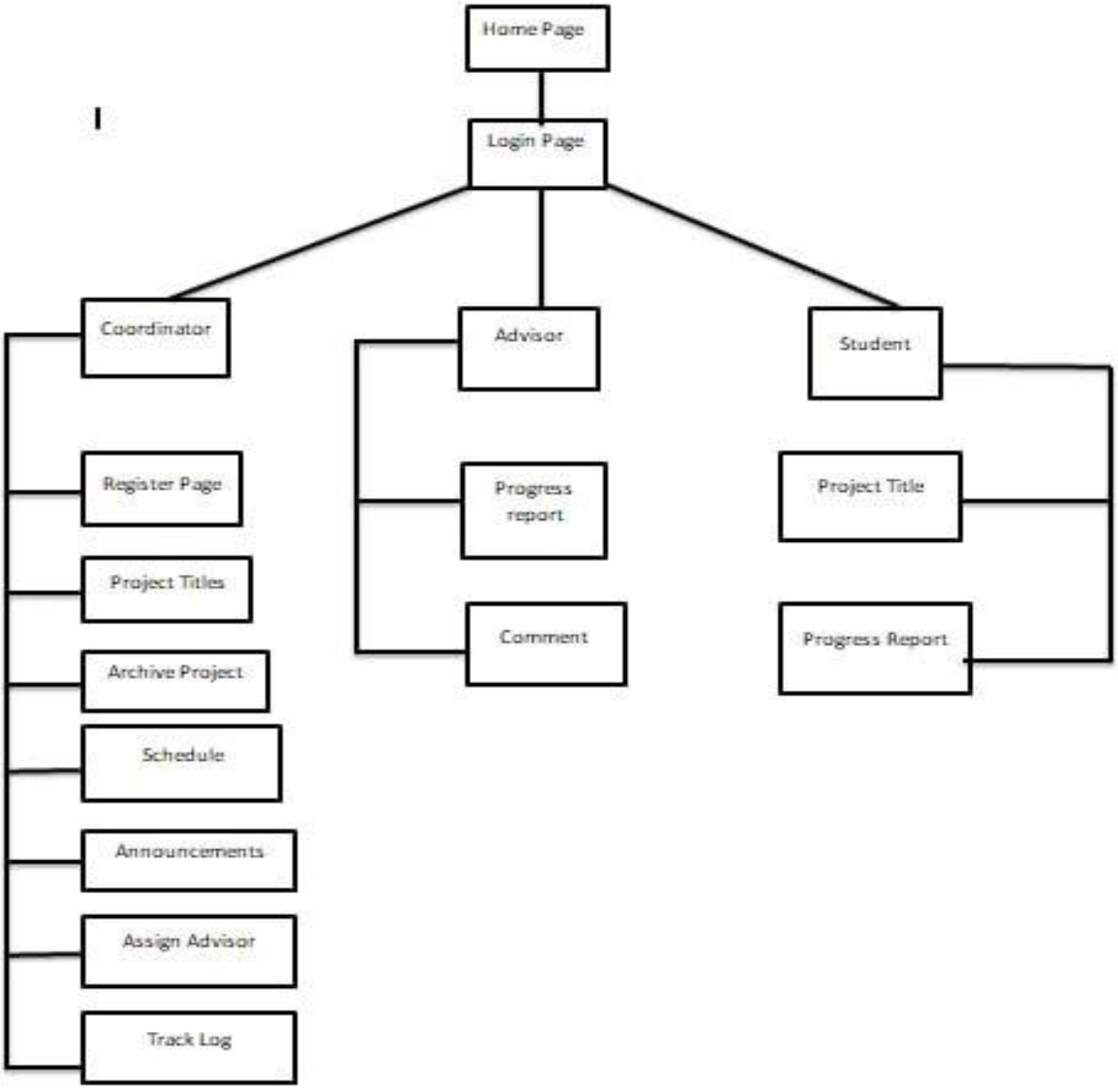


Figure 6. User Interface Flow Diagram

5.7.2 User interface design

Home Page

This page provides an overview of the system's features and navigation, and also displays the highlight information of some active project for users.

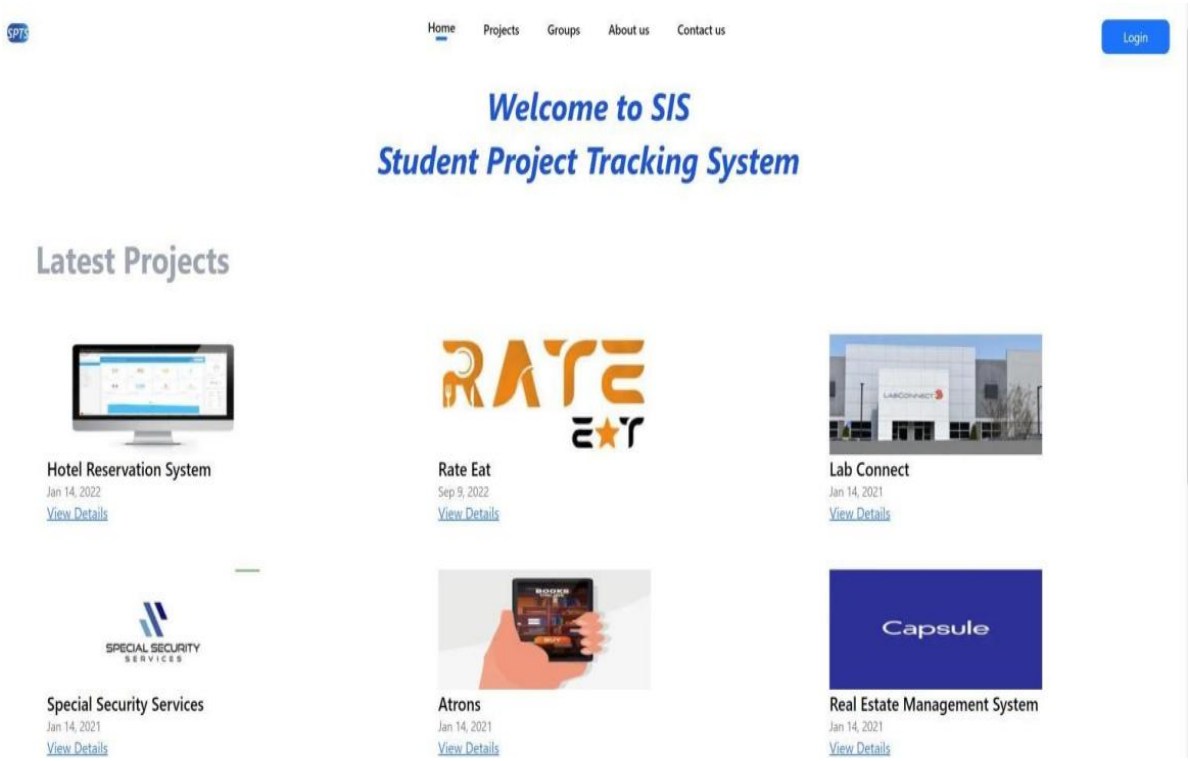


Figure 7. Home Page UI

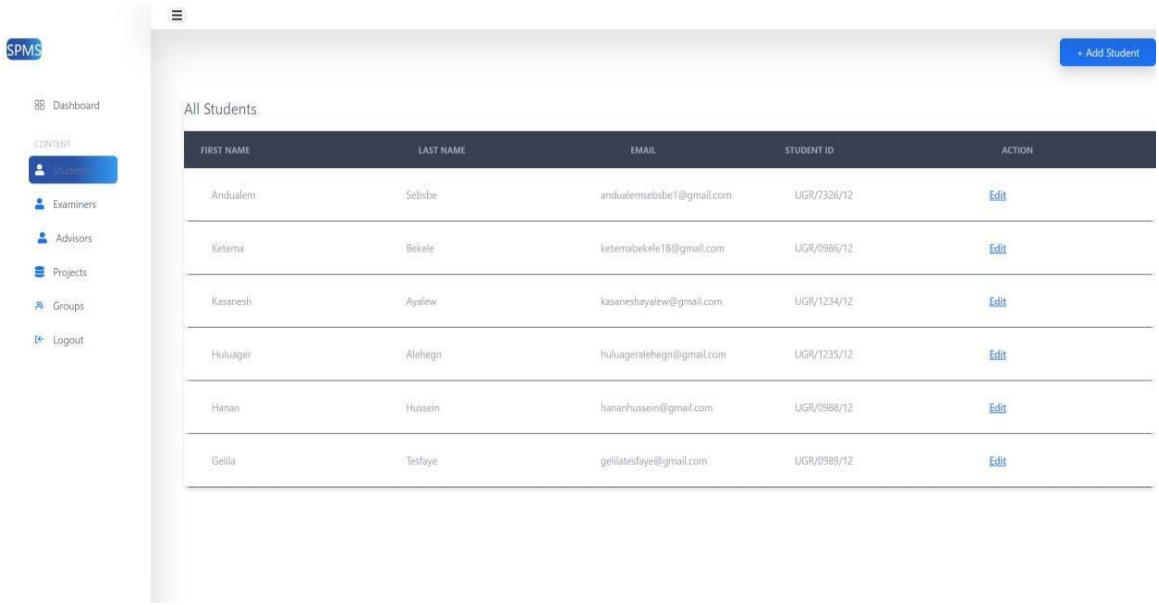


Figure 8. Coordinator Dashboard

Description: All registered student are viewed and modified with one screen.

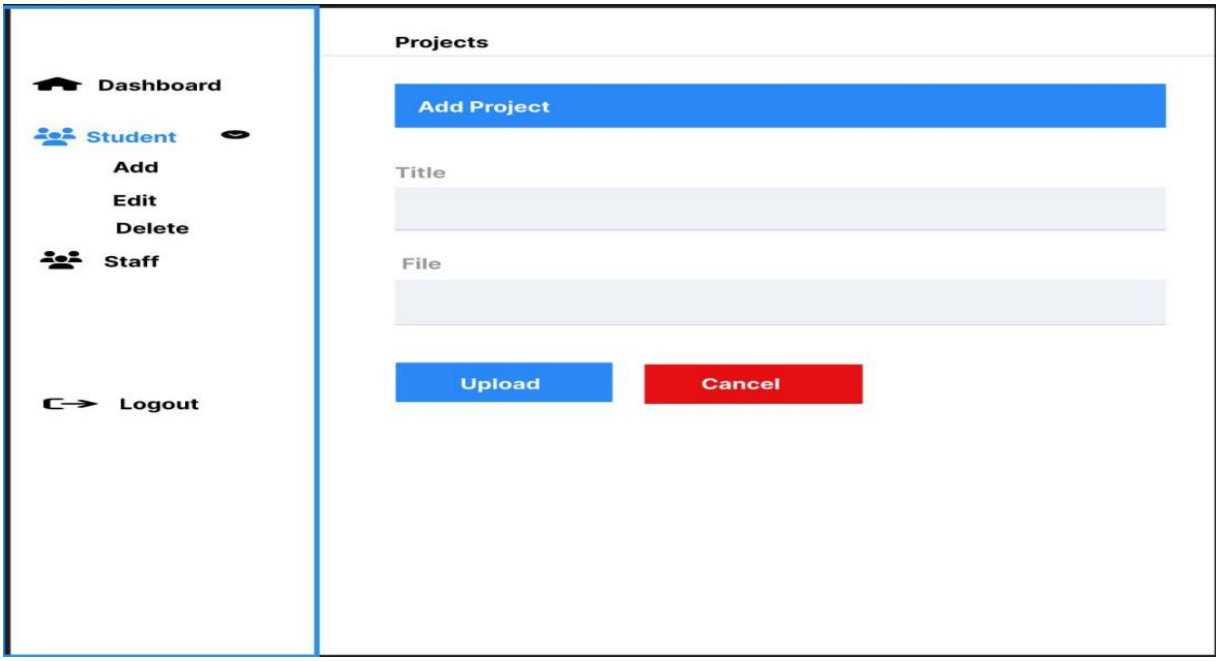


Figure 9. Student dashboard

Description: this interface allows the student to upload project title with description.

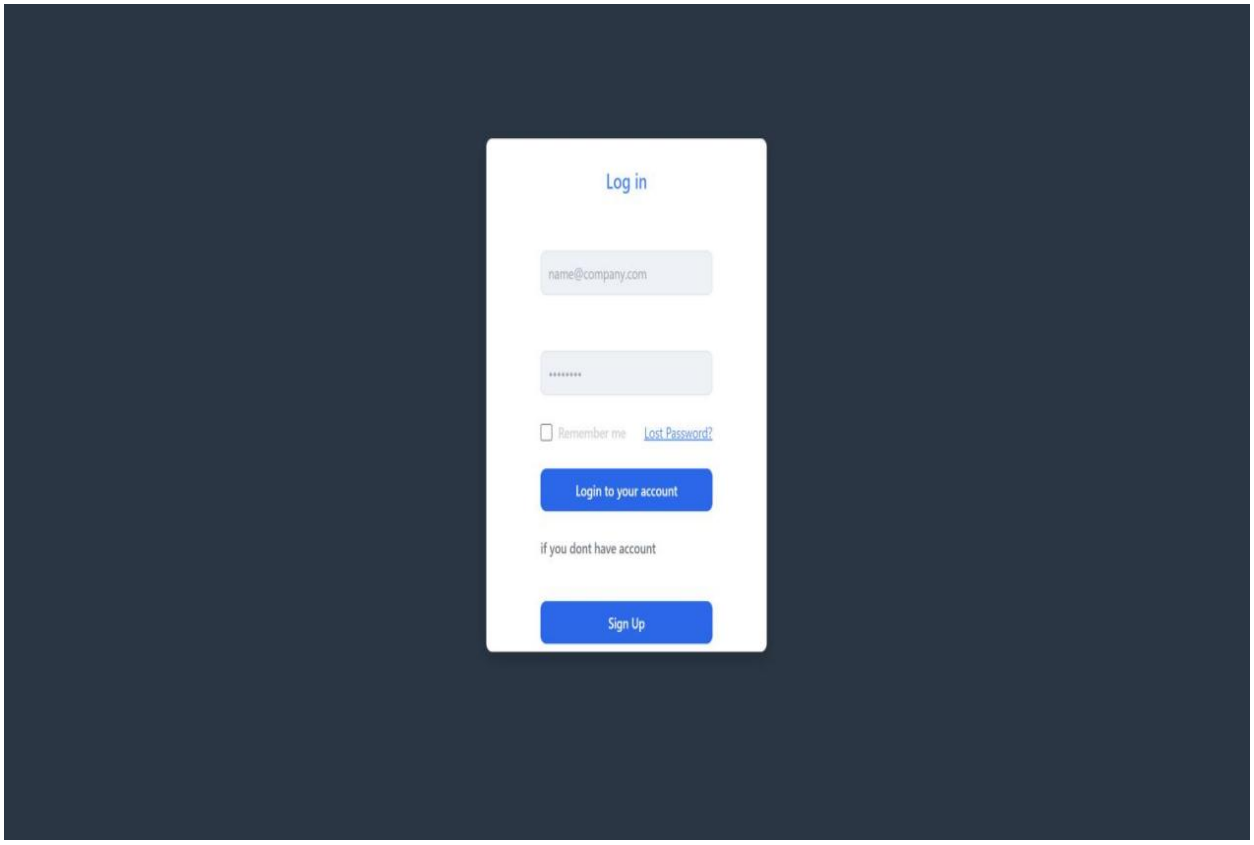


Figure 10. Login Page

Description: this screen shows the login page to authenticate the user and checks the credential.

Chapter six

Object oriented implementation

6.1 Introduction

System implementation is a process that converts the system requirements and design into program codes. This phase at times involve some modifications to previous design. The development environment has certain impact on the development of a system. System development consists of hardware and soft configurations. Using the suitable hardware and software is an important factor in determining the success of the project.

6.2 Testing and testing procedures

The primary purpose of testing is to ensure the resulting component of program as well as the program as a whole fulfills the requirements specification and to eliminate errors in the program, thus, our team tried to perform testing procedure to ensure the systems requirement and functionality.

We have performed certain procedures for testing software, beginning with unit or component testing and working towards integration and system testing as a whole.

6.2.1 Unit testing

This test is conducted to check whether the internal logic is functioning properly and program inputs produce valid outputs that compare with the expected results. It is done after the completion of an individual unit. These tests are performing at component level and specific business process, application, and system configuration. In this testing module interface is tested to assure that information properly and correctly flows into and out of the module. This testing involves the testing of data truncation, the structure of the data, and whether the program correctly accepts the input data. The whole validation of the program is encountered in this testing.

Sample unit test cases

Table 6-1The landing page testing case specification

Landing page test	
Test case identifier	Landing page
Test location	Home page
Feature to be tested	The first page appears on the very first call to the systems webpage.
Test Cases	To check if the home page is always the first page that appears when the webpage is opened, to ensure that users are directed to the correct landing page and that the website functions as expected.
Result	Finely work

Table 6-2:The navigation testing case specification.

Navigation test	
Test case identifier	Page Navigation
Test location	In every page
Feature to be tested	Consistent navigation to the respective pages.
Test Cases	If the system allows the user to navigate throughout the privileged page seamlessly.
Result	Finely work

6.2.2 Integration testing

Integration testing is combining modules testing to verify the proper functioning of the system's integrated components. The purpose of integration testing is to ensure that the different modules and components of the system work together seamlessly and that the system as a whole meets its functional and non-functional requirements.
In the context of our system, before starting the integration testing we have tried to integrate individual modules created under unit testing above to create integrated module. After that we have tried to check that if different modules are communicating with each other or not since it is critical to test every modules interaction effect on the entire program model. Based on our test result, we have found that modules are working well.

6.2.3 System testing

System testing is a critical part of the software development process, as it helps to ensure that the system is working correctly. When the unit testing and the integration testing have been completed, the whole system would be tested to ensure the software product does not fail. This testing procedure ensures complete implementation of our system without unexpected and unnecessary bugs and errors that may occur and affect the performance of the system. Therefore to apply it we run the whole system and tested for logical errors and as a result showed that there is no logical as well as syntax errors. From the functionality perspective of our system we have made the following testing's.

- **Functional testing**: Functional testing ensures that the system meets the functional requirements specified for the project. This includes testing features such as project title submission, project approval work flow, task assignment, progress tracking, and evaluation. We can designed different test cases to verify that the system can handle the full range of scenarios that students and faculty members might encounter during the project life cycle.
- **Usability testing**: Usability testing is designed to ensure that the system is easy to use and intuitive for all users of the system. It is mainly used to analyze the potential area of improvement to the system. The can include testing the user interface, navigation, and overall user experience.

System testing samples

Table 6-3: Login testing case specification

A: Login	
Test case identifier	Login
Test location	Login page
Feature to be tested	Authentication and Authorization of the user
Data	Password and username information in the login page.
Test Cases	If the system correctly accepts and validates inputs, if the system validates a user by comparing the decrypted form of the related password of the user with the entered password. If the system displays only the resources that are authorized for that user only.
Result	Validate user

Table 6-4: User registration testing case specification

B: Register	
Test case identifier	Register
Test location	Register page
Feature to be tested	Applicant Registration should be stored on database properly
Data	Applicant information should be available
Test Cases	If the system checks that applicant's information completely entered and they all are valid. If the system checks for available user name in the database, in order not to allow username duplication.
Result	Registered applicant's

Table 6-5: Upload project testing case specification

C: Upload project	
Test case identifier	Upload project
Test location	Upload project page
Feature to be tested	Project works should be stored on database properly
Data	Final project works should be available
Test Cases	If the system stored project works. If the system checks for available project titles in the database, in order not to allow project title duplication
Result	Upload project works.

Table 6-6: Search project testing case specification

D. Search project	
Test case identifier	Search
Test location	search project page
Feature to be tested	Project works should be retrieved from the database properly.
Data	Information about projects
Test Cases	If the system retrieves project If the system checks for active project in the database, and displays the relevant project information.
Result	Search projects work.

Table 0-7: Approve title testing case specification

E. Approve title	
Test case identifier	Title_approval
Test location	approve project page
Feature to be tested	Approval status of the project title.
Data	Project title approval status information
Test Cases	If the title is approved by coordinator, If the system changes approval status in the respective student page.
Result	Approval status changed

6.3 deployment / installation process

The deployment process for a software application requires careful planning, testing, and monitoring to ensure that the application is stable, secure, and meets the requirements of end-users. The deployment process for a software application involves the steps and procedures for releasing the application into a production environment. The deployment process can vary depending on the complexity of the application, the deployment environment, and the requirements of the organization.

The deployment and installation process for a web based system can vary depending on the specific hosting environment and deployment strategy. Hence our system is developed with the Next.js there are different tools offer different features and pricing structure. In our case we will intend to use Vercel which is a cloud-based platform that specializes in hosting Next.js applications. It offers a seamless deployment process, automatic scaling, and a range of features for optimizing performance and security.

6.3.1 Hardware and software acquisition

For a typical, full deployment of the system, the hardware and software acquisition is necessary. The hardware and software requirement for this system installation are derived from industry standard technologies and widely available commercial products.

6.3.1.1 Hardware

- Computer
- Application server
- Processor
- Hard disk
- Monitor
- Pointing device: Mouse
- Database
- Backup storage device
- CD/DVD or flash disks

6.4.1.2 Software

Software plays a very crucial role in the development of any system. For the new system we have developed the appropriate operating system in Microsoft Window 10

- Internet browser
- Mongo DB Atlas
- Moon modeler application software

Chapter 7

Conclusion and recommendation

7.1 Conclusions

This paper shows the overall step for developing a web based Student Project Tracking System for the school of information science (SIS). It provides an automated form of submitting, approving, retrieving and storing student projects. Moving from a current manual work system to the automated system helps the SIS to reduce costs, increase project security and minimizing redundant project works. To design and format the web system different kinds of tools are used like: tailwind instead of CSS, Next.js, and some other tools. The database was created using MongoDB database management system. Generally, our System will benefit the school Employee's, and the students. It reduces the time spent in managing projects and ensures that the system students will find important information about different projects timely.

As project come up to the end we pass all software development life cycles. Starting from system specification up to system testing and at each stage system development we understand clearly what to be done and what need to be done for the future. For example, in analysis, we tried to model the new and proposed system using UML diagrams: - use case diagrams, sequence diagrams, class diagrams and component diagrams. And from this we understood how to model the system.

In general, the project (system) that the team developed will benefit the enterprise in by changing its business range from manual to online level.

The overall benefits of the system to the school is to have a repository for projects, minimize the time required to perform task, and reducing human power and cost that are spent on the manual system

7.2 Recommendations

Our system web-based system so, it is easy to use and requires only basic computer skills, experience on how to browse. while we were doing this project the team members has faced some challenges, but by the cooperation of all the group members and the advisor, the team is now able to reach to the final result. I.e. all the group members strongly fight these challenge and take the turn to the front.

What were the gaps of our project?

Here are the gaps in our project that would benefit for the future work.

1. The system does not notify students via email about the project submission deadline.
2. The system is not integrated with other systems like the student information system of the school.
3. Some gaps in time management, making it difficult for us to finish the projects before the deadline.
4. Difficulties in selecting project titles and early understanding the business logic behind the project.

So now depending on the limitations and gaps that we have identified in our project, all the group members jot down some recommendations for the coming students who want to improve this system and work better projects than the present.

- Should implement an email notification system: To address the issue of students not being notified when the project submission deadline is approaching, consider implementing an email notification system that sends automatic reminders to students as the deadline approaches. This will help students stay on track and to ensure that they submit their projects on time.
- The should work to integrate this system with other systems that it needs to interact with, such as student information systems, clearance management system, and learning management systems etc.
- Should have good time management to finish before the deadlines,by breaking down tasks into smaller steps, and prioritizing tasks based on importance and urgency.
- Should develop clear guidelines and business logic: In order not to be confused when selecting project titles and understanding the business logic and should have a clear on what they are intending to develop before starting on it.

Appendix

1. Data base connection

```
import mongoose from "mongoose";

const { MONGODB_URI } = process.env

if (!MONGODB_URI) {
   throw new Error("Invalid environment variable: MONGODB_URI");
}

export const connectToMongoDB = async () => {
   try {
      const { connection } = await mongoose.connect(MONGODB_URI)

      if (connection.readyState === 1) {
         return Promise.resolve(true)
      }

   } catch (error) {
      return Promise.reject(error)
   }
}
```

2. Upload project

```
import { useState } from 'react';
import axios from 'axios';

export default function ProjectForm() {
  const [project, setProject] = useState({
    title: '',
    description: '',
    academic_year: '',
    image: null,
    file: null,
  });

  function handleInputChange(event) {
    const { name, value } = event.target;
    setProject({ ...project, [name]: value });
  }

  function handleImageChange(event) {
    const file = event.target.files[0];
    const reader = new FileReader();
    reader.onloadend = () => {
      setProject({
        ...project,
        image: { data: Buffer.from(reader.result), contentType: file.type },
```

```
    });
  };
  reader.readAsArrayBuffer(file);
 }

 function handleFileChange(event) {
  const file = event.target.files[0];
  const reader = new FileReader();
  reader.onloadend = () => {
   setProject({
    ...project,
    file: { data: Buffer.from(reader.result), contentType: file.type },
   });
  };
  reader.readAsArrayBuffer(file);
 }

 function handleSubmit(event) {
  event.preventDefault();
  axios
   .post('/api/projects', project)
   .then((response) => {
    console.log(response.data);
   })
   .catch((error) => {
    console.error(error);
   });
 }

 return (
  <div className='bg-[#293645]px-4 p-10 px-24 text-center justify-center ml-36'>
  <div className=" absolute py-24 space-y-4 bg-gray-50 text-center justify-center
">
   <form onSubmit={handleSubmit} className='max-w-md mx-auto'>
    <div className='mb-4'>

    <label htmlFor="title">Title:</label>
    </div>

    <input
     type="text"
     id="title"
     name="title"
     value={project.title}
     onChange={handleInputChange}
     className='w-full border-gray-400 border-2 rounded-md py-2 px-4 text-gray-
700 focus:outline-none focus:border-blue-500'
    />
    <br />
    <br />
```

```
      <label htmlFor="description">Description:</label>
      <textarea
        id="description"
        name="description"
        value={project.description}
        onChange={handleInputChange}
        className="w-full border-gray-400 border-2 rounded-md py-2 px-4 text-gray-
700 focus:outline-none focus:border-blue-500"
      />
      <br />
      <br />

      <label htmlFor="academic_year">Academic Year:</label>
      <input
        type="text"
        id="academic_year"
        name="academic_year"
        value={project.academic_year}
        onChange={handleInputChange}
        className="w-full border-gray-400 border-2 rounded-md py-2 px-4 text-gray-
700 focus:outline-none focus:border-blue-500"
      />
      <br />
      <br />

      <label htmlFor="image">Image:</label>
      <input
        type="file"
        id="image"
        name="image"
        accept="image/*"
        onChange={handleImageChange}
      />
      <br />
      <br />

      <label htmlFor="file">File:</label>
      <input
        type="file"
        id="file"
        name="file"
        accept=".pdf"
        onChange={handleFileChange}
      />
      <br />
      <br />

      <button type="submit">Submit</button>
    </form>
```

```
    </div>
    </div>
  );
}
```

Upload project model and api

```
//api to add the project to the database
import connectMongo from "../../../lib/mongoconnect";
import Project from "../../../models/projects";
import { NextApiRequest, NextApiResponse } from "next";

export default async function uploadTest(req:NextApiRequest, res:NextApiResponse)
{
  const {project_title, description}=req.body;
  console.log('connecting to mongo');
  await connectMongo()
  console.log('connected to mongo');

  console.log('creating a document');
  const project = await Project.createCollection(req.body);
  console.log('created document');

  res.json({project});

import { Schema, models, model} from "mongoose";

const projectSchema = new Schema({
  Project_title: {
    type: String,
    required: true
  },
  description: {
    type: String,
    required: true
  },
  academic_year: {
    type: String,
    required: true
  },
  image: {
    data: Buffer,
    contentType: String
  },
  file: {
    data: Buffer,
    contentType: String
  }
});
const Final_project =models.Final_project || model('Final_project',projectSchema)
export default Final_project
```

//api

3. Create student modal

```
'use client'
import { useRouter } from "next/navigation"
import { useState } from "react"
import axios, {AxiosError} from "axios"
import { InputErrors } from '../../types/error'
import { getErrorMsg, loginUser } from '../../helpers'
import { FormData } from "../../types"
interface Students{
  showModal: boolean
  setShowModal: (showModal: boolean) => void
}

const initialFormValues: FormData = {
  fullName: '',
  email: '',
  username: '',
  password: '',
  role: 'student',
  confirmPassword: ''
}

export const CreateStudentModal = ({
  showModal,
  setShowModal,
//   students
} : Students) => {

    const [form, setForm] = useState<FormData>(initialFormValues)
    const [submitError, setSubmitError] = useState<string>('');
    const router = useRouter()


    const [validationErrors, setValidationErrors] = useState<InputErrors[]>([])
    const [loading, setLoading] = useState(false)
    // console.log(role)
    const validateData = (): boolean => {
        const err = []

        if (form.fullName?.length < 4) {
            err.push({ fullName: "Full name must be atleast 4 characters long" })
        }
        else if (form.fullName?.length > 30) {
            err.push({ fullName: "Full name should be less than 30 characters" })
        }
        else if (form.password?.length < 8) {
            err.push({ password: "Password should be atleast 8 characters long" })
```

```
    }
    else if (form.password !== form.confirmPassword) {
      err.push({ confirmPassword: "user id don't match" })
    }

    setValidationErrors(err)

    if (err.length > 0) {
      return false
    }
    else {
      return true
    }
  }

  const handleSignup = async (event: React.FormEvent<HTMLFormElement>) => {
    event.preventDefault()

    const isValid = validateData()

    if (isValid) {
      // sign up
      try {
        setLoading(true)
        const response = await fetch('/api/auth/signup', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify(form),
        });

        // const apiRes = await response.json();
        // console.log(apiRes);

        if(response.ok){
          console.log("Form Submitted Successfully")
        }
        else{
          const data = await response.json();
          setSubmitError(data.error);
          // console.log(data.error);
        }
      } catch (error) {
        console.error('An error occurred while submitting the form:', error);
        setSubmitError('An error ocurred while submitting the form');
      }setForm(initialFormValues)
      setLoading(false)
    }
  }

  return (
```

```
      <>
    <div className="justify-center items-center flex overflow-x-hidden overflow-y-
auto fixed inset-0 z-50 outline-none focus:outline-none">
      <div className="relative w-auto my-6 mx-auto max-w-3xl">
        <div className="border-0 rounded-lg shadow-lg relative flex flex-col w-full
bg-white outline-none focus:outline-none">
        <div className="relative p-6 flex-auto">
          <div className="flex flex-col justify-evenly bg-white px-7">

           <form
            className="text-secondary-text"
            encType="multipart/form-data"
            onSubmit={handleSignup}
            >
            <div className="mb-4">
               <h1 className="text-xl text-center mb-1 text-primary-text">
               Add New Student
               </h1>

            </div>
               <div className="mb-4">
                  <label htmlFor="firstName" className="text-sm text-gray-700 font-
semibold">
                  Full Name
                  </label>
                  <input
                  name="fullName"
                  className="text-sm  block border-secondary-text-100  rounded
border-2 border-solid w-full p-1"
                  value={form.fullName}
                  onChange={e => setForm({...form, fullName: e.target.value})}
                  required
                  />
               </div>

               <div>
               <div className="mb-4">
                  <label htmlFor="email" className="text-sm text-gray-700 font-
semibold">
                  Email
                  </label>
                  <input
                  name="email"
                  className="text-s block border-secondary-text-100  rounded border-2
border-solid w-full p-1"
                  value={form.email}
                  onChange={e => setForm({...form, email: e.target.value})}
                  required
                  />
                  <div className="text-red-400 text-sm py-1">
```

```
                </div>
              </div>

              <div className="mb-4">
                <label htmlFor="username" className="text-sm text-gray-700 font-
semibold">
                Student ID
                </label>
                <input
                name="username"
                className="text-s block border-secondary-text-100  rounded border-2
border-solid w-full p-1"
                value={form.username}
                onChange={e => setForm({...form, username: e.target.value})}
                required
                />
                <div className="text-red-400 text-sm py-1">

                </div>
              </div>
              <div className="flex gap-10">
               <div className="mb-4">
                  <label htmlFor="password" className="text-sm text-gray-700 font-
semibold">
                  Password
                  </label>
                  <input
                  name="password"
                  className="text-s block border-secondary-text-100  rounded border-
2 border-solid w-full p-1"
                  value={form.password}
                  onChange={e => setForm({...form, password: e.target.value})}
                  required
                  />
                  <div className="text-red-400 text-sm py-1">

                  </div>
                </div>
                <div className="mb-4">
<label htmlFor="confirmPassword" className="text-sm text-gray-700 font-
semibold">
                  Confirm Password
                  </label>
                  <input
                  name="confirmPassword"
                  className="text-s block border-secondary-text-100  rounded border-
2 border-solid w-full p-1"
                  value={form.confirmPassword}
```

```
                    onChange={e => setForm({...form, confirmPassword:
e.target.value})}
                    required
                    />
                    <div className="text-red-400 text-sm py-1">

                    </div>
                  </div>
                </div>
                </div>

                <div className="flex items-center justify-end gap-4 ">
                    {submitError && <p>{submitError}</p>}
                    <button
                    className="bg-blue-500 rounded text-white text-sm px-4 py-2
border-2 border-primary shadow hover:shadow-lg outline-none focus:outline-none"
                    type="submit"
                    >
                    Submit
                    </button>

                    <button
                    className="bg-red-500 font-bold px-4 py-2 text-sm border-2 rounded
border-secondary-text "
                    type="button"
                    onClick={() => setShowModal(false)}
                    >
                    Cancel
                    </button>
                </div>
              </form>
            </div>
            </div>
          </div>
        </div>
      </div>
      <div className="opacity-25 fixed inset-0 z-40 bg-black"></div>
    </>
    )
}
```

3. Fetching student api

```
import User, {UserDocument} from "../../models/user";
import { connectToMongoDB } from "../../lib/mongodb";
import { NextApiRequest, NextApiResponse } from "next";

export default async function handler (req: NextApiRequest, res: NextApiResponse) {
    if (req.method !== 'GET') {
        res.status(405).json({ message: 'Method Not Allowed' });
```

```
        return;
    }

    try {
        const db = await connectToMongoDB()
        const students: UserDocument[] = await User.find({role: 'student'})
        res.status(200).json({students})
    } catch (error) {
        return new Response("Failed to fetch all prompts", { status: 500 })
    }
}
```

References

[1] Ambler, S.W. 2004. The Object Primer 3rd Edition - The Application Ambler, S.W. 2001. Ambler, S. W. June 2001. The object primer 2 edition.

[2] Developer's Guide to Object Orientation and The UML, Cambridge University Press, 2001

[3] www.ambysoft.com/theObjectPrimer.html Developer's Guide to Object Orientation and The UML, Cambridge University Press, 2004,

[4] https://en.wikipedia.org/wiki/Document-oriented_database

[5] https://josipmisko.com/best-next-js-books. Next.js quick start guide

[6] http://repository.smuc.edu.et/bitstream/123456789/5296/1/ELECTRONIC%20DOCUMENT%20MANAGEMENT%20SYSTEM-converted%20%281%29.pdf