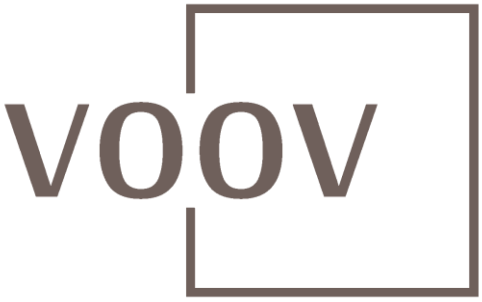
Voov: Project Phase 1



Team Members:

Aquino, Trisha 000916516

Buenavista, Katrina Keen 000922346

Bui, Joseph 000927242

Cliffe, Matthew 000916002

Hao, Yichen 000938290

Thieu, Emily 000915665

Table of Contents

External Client Information:

Name: Dino Fabie

Position: Head Registrar Administrator

Contact Details:

Phone Number: +63822273469 (school administration)

Email Address: evelynefabiecollegeinc@gmail.com

Name: Cecile Fabie

Position: Head Administrator

Contact Details:

Phone Number: +639453191209

Email Address: evelynefabiecollegeinc@gmail.com

Project Topic: School Management System (SMS)

Description: In today's digital age, managing student registrations and related administrative tasks efficiently is crucial for educational institutions. This document outlines the development of a comprehensive School Management System (SMS) focusing primarily on streamlining student registration processes and providing an intuitive website interface through the use of REACT and SQL. The system aims to improve operational efficiency, enhance user experience, and support the institution's administrative needs.

Current System:

- Excel files for data organization
- Facebook for communication and school information
- Google Forms for student registration

Proposed System:

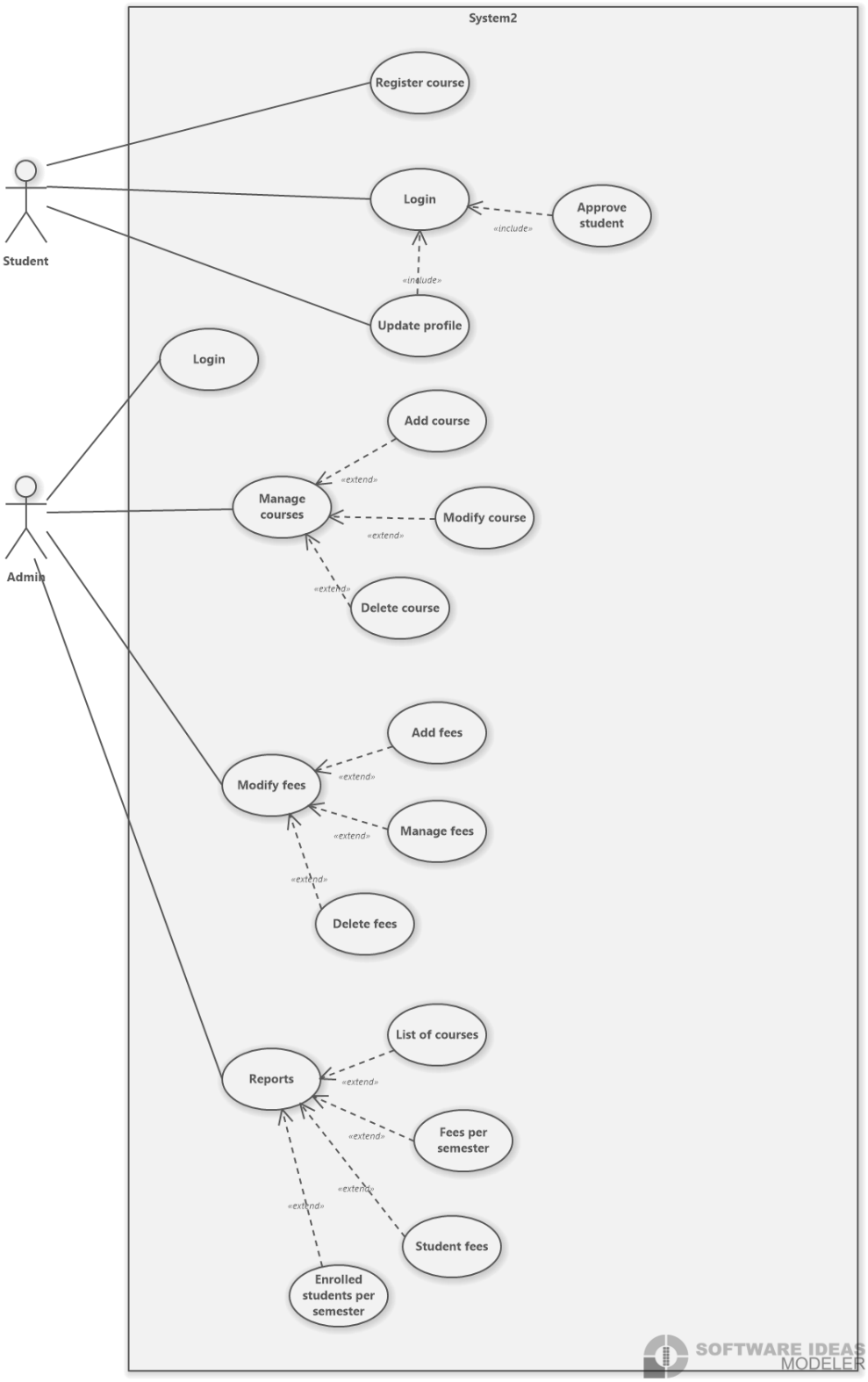
- **Online Registration Form:** An easy-to-fill, online registration form for new students.
- **Document Upload:** Secure upload feature for necessary documents (e.g., birth certificate, previous school records).
- **Automated Verification:** Automated system for verifying submitted documents and data.
- **Status Tracking:** Real-time status updates on registration progress.
- **Notifications:** Automated email/SMS notifications to inform students and parents of registration status and required actions.

- **User Accounts:** Secure login for students, parents, and administrators.
- **Dashboard:** Personalized dashboards displaying relevant information (e.g., registration status, upcoming deadlines).
- **Information Portal:** Access to school policies, academic calendars, and contact information.
- **Interactive Forms:** Forms for additional requests, feedback, and communication with the school.
- **Event Calendar:** Display upcoming events, deadlines, and school activities.
- **Responsive Design:** Mobile-friendly design to ensure accessibility on various devices.
- **Data Management:** Tools for administrators to view, edit, and manage student records.
- **Reporting:** Generate reports on registration statistics, student demographics, and other key metrics.
- **User Management:** Control access levels and permissions for different user roles (e.g., administrators, teachers).

Advantages of Proposed System:

The proposed School Management System aims to revolutionize the student registration process and enhance the overall user experience through a modern, web-based platform. By implementing this system, the institution will achieve greater efficiency, better data management, and improved communication channels.

Use Case Diagram



Use Case Descriptions

Use Case 1: Student Login

Description: Student logs onto account to register for courses

Actors: Student

Preconditions: Requires admin verification (for first signing up) and valid login credentials (username and password).

Postconditions: User is led to the dashboard where they can register for courses, and update student info in their profile.

Triggers: Clicks on the login page

Main Flow:

- Navigate to Login Page: The student accesses the login page.
- Enter Credentials: The student inputs the username and password.
- Click on the "Log in" button
- Credential Validation: The system verifies the username and password. If valid, the system grants access; if invalid, an error message is displayed.
- Access Dashboard: The student is led to the main dashboard of the system.

Alternative Flow: Invalid Login: Displays error message for invalid login info whether for username or password and prompts the user to try again.

Use Case 2: Manage Fees

Description: The admin manages the student fees by adding, modifying, or deleting financial fee information

Actors: Admin

Preconditions: Requires admin to be verified and logged into the system and have access to the manage fee's function

Postconditions: Can view, change, add, or delete fees to update the system

Triggers: Clicks on the manage fees page

Main Flow:

- **Access Manage Fees page:** The admin navigates to the "Manage Fees" section from the dashboard.
- **Add Fees:**

- The admin clicks on the "Add Fees" button.

The admin fills in the required info for student fees (tuition, supplies, etc.)

The admin submits the form.

The system confirms the addition and updates the fees list.

Several different student fees list will be connected to different student users in the system

- **Modify Courses:**

The admin selects a modification to an existing student fee.

The admin edits the necessary financial details. (Setting it to 0 if the fee has been paid for example)

The admin submits the changes.

The system confirms the updates and refreshes the fees list.

- **Delete Courses:**

The admin selects a fee to delete.

The system prompts confirmation.

The admin confirms the deletion.

The system removes the fee from the list and updates the database.

Alternative Flow: Cancellation of Operations: The admin can cancel any add, modify, or delete operation, returning to the manage fees page without changes.

Use Case 3: Reports

Description: Gives data on the total fees, student amount, courses, and other necessary information to the admin.

Actors: Admin

Preconditions: Requires admin to be verified and logged into the system and have access to the reports function.

Postconditions: Can access, change, add, and delete information regarding payments and courses and make a report on it.

Triggers: Clicks on the reports page

Main Flow:

- **Access Reports page:** The admin navigates to the "Reports" section from the dashboard.
- **Add Report info:**

The admin clicks on the "Make report" button.

The admin fills in the required details (e.g., courses, student fees, student amount).

The admin submits the form.

The system confirms the addition and updates the reports list.

- **Modify report info:**

The admin selects an existing report to modify.

The admin edits the necessary report details.

The admin submits the changes.

The system confirms the updates and refreshes the reports list.

- **Delete report:**

The admin selects report info to delete.

The system prompts confirmation.

The admin confirms the deletion.

The system removes that specific info from the list and updates the database.

Alternative Flow: Cancellation of Operations: The admin can cancel any addition, modification, or deletion of operations, returning to the reports management overview without changes.

Use Case 4: Admin Login

Actor: Admin

Description: This use case allows an admin to log into the system to access administrative functionalities securely.

Precondition: The admin must have valid login credentials (username and password).

Postcondition: The admin is logged into the system and redirected to the dashboard.

Main Flow:

- Navigate to Login Page: The admin accesses the application login page.
- Enter Credentials: The admin inputs the username and password.

- **Submit Login:**

The admin clicks the "Log In" button.

Credential Validation: The system verifies the credentials.

- If valid, the system grants access; if invalid, an error message is displayed.
- Access Dashboard: Upon successful login, the admin is redirected to the main dashboard.

Alternative Flow: Invalid Login Attempt: If the admin enters incorrect credentials, the system displays an error message and prompts for re-entry.

Use Case 5: Manage Courses

Actor: Admin

Description: This use case allows the admin to manage course offerings by adding, modifying, or deleting courses.

Precondition: The admin must be logged into the system and have access to the course management module.

Postcondition: Courses are updated in the system based on the admin's actions.

Main Flow:

- Access Course Management: The admin navigates to the "Manage Courses" section from the dashboard.
- **Add Courses:**

The admin clicks on the "Add Course" button.

The admin fills in the required course details (e.g., title, description, duration).

The admin submits the form.

The system confirms the addition and updates the course list.

- **Modify Courses:**

The admin selects an existing course to modify.

The admin edits the necessary course details.

The admin submits the changes.

The system confirms the updates and refreshes the course list.

- **Delete Courses:**

The admin selects a course to delete.

The system prompts confirmation.

The admin confirms deletion.

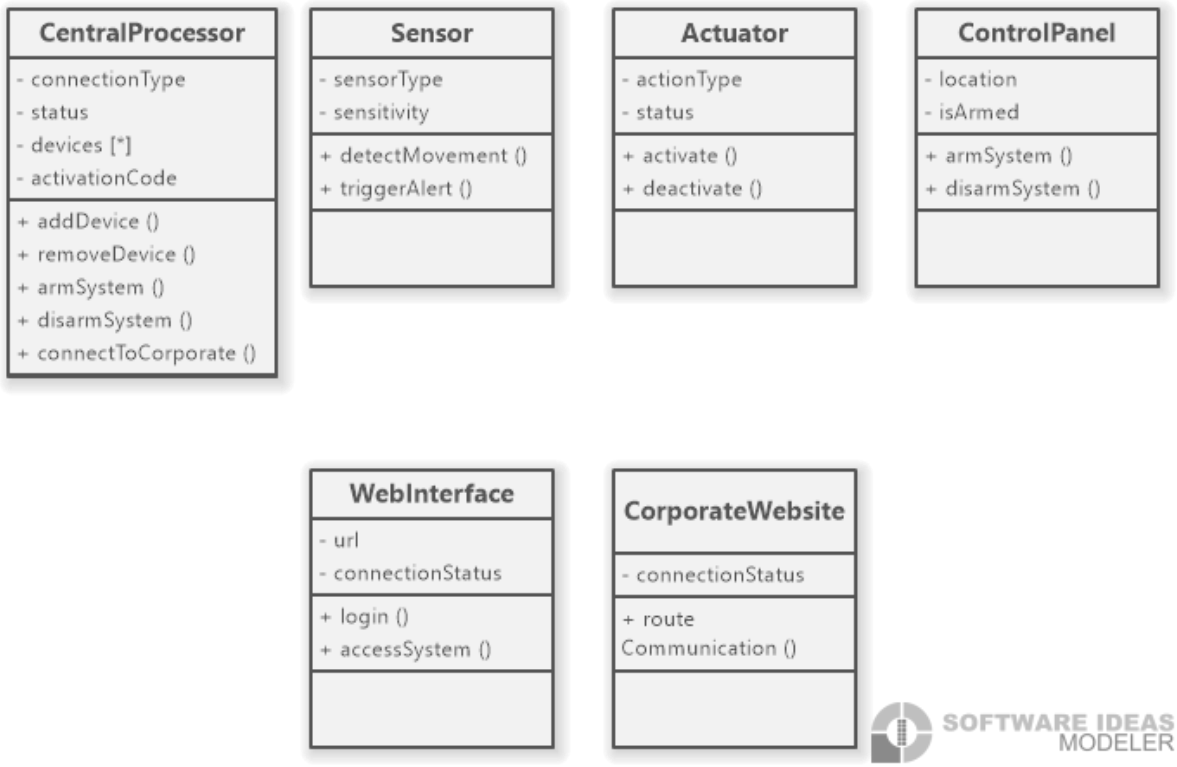
The system removes the course from the list and updates the database.

Alternative Flow: Cancellation of Operations: The admin can cancel any add, modify, or delete operation, returning to the course management overview without changes.

Exceptions:

- The system may display error messages if the admin attempts to add a course with duplicate titles or if there are issues during course management.

Class Diagram



Appendix

Team Constitution

Established Norms and Expectations

Conduct

It is required of every member to abide by the following rules. Penalties for breaking the written regulations may vary depending on the offense committed:

If someone misses a meeting or class without warning, they will have 24 hours to explain. If the group determines that the absence is warranted, the offender will get a verbal warning.

If a group member reaches a deadlock in decision-making and won't give up, a game of Uno (with stacking) will be used to decide the winner.

It is required for members to attend meetings at the appointed time and day. If a member anticipates being late for the meeting, they should let everyone know. "On time" is defined as starting the meeting within ten to fifteen minutes of the stated start time.

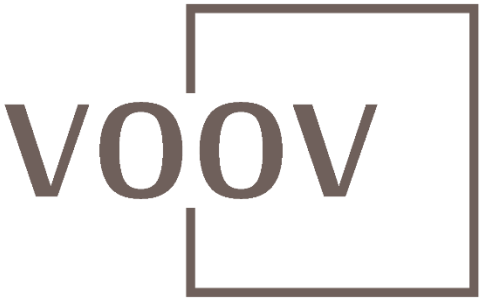
If a member will not be present at the planned meeting, they should let the other members know.

Throughout the project, it is anticipated that each participant will treat the other with appropriate decorum and respect; if not, they will get a verbal warning.

If any modifications should be made to the group members' tasks, they should always be informed. When an issue arises, don't be afraid to tell everyone right away so that a solution may be found.

It is required for every member to finish allocated work on schedule. If necessary, extensions can be worked out.

Voov: Project Phase 2



Team Voov

Aquino, Trisha 000916516

Buenavista, Katrina Keen 000922346

Bui, Joseph 000927242

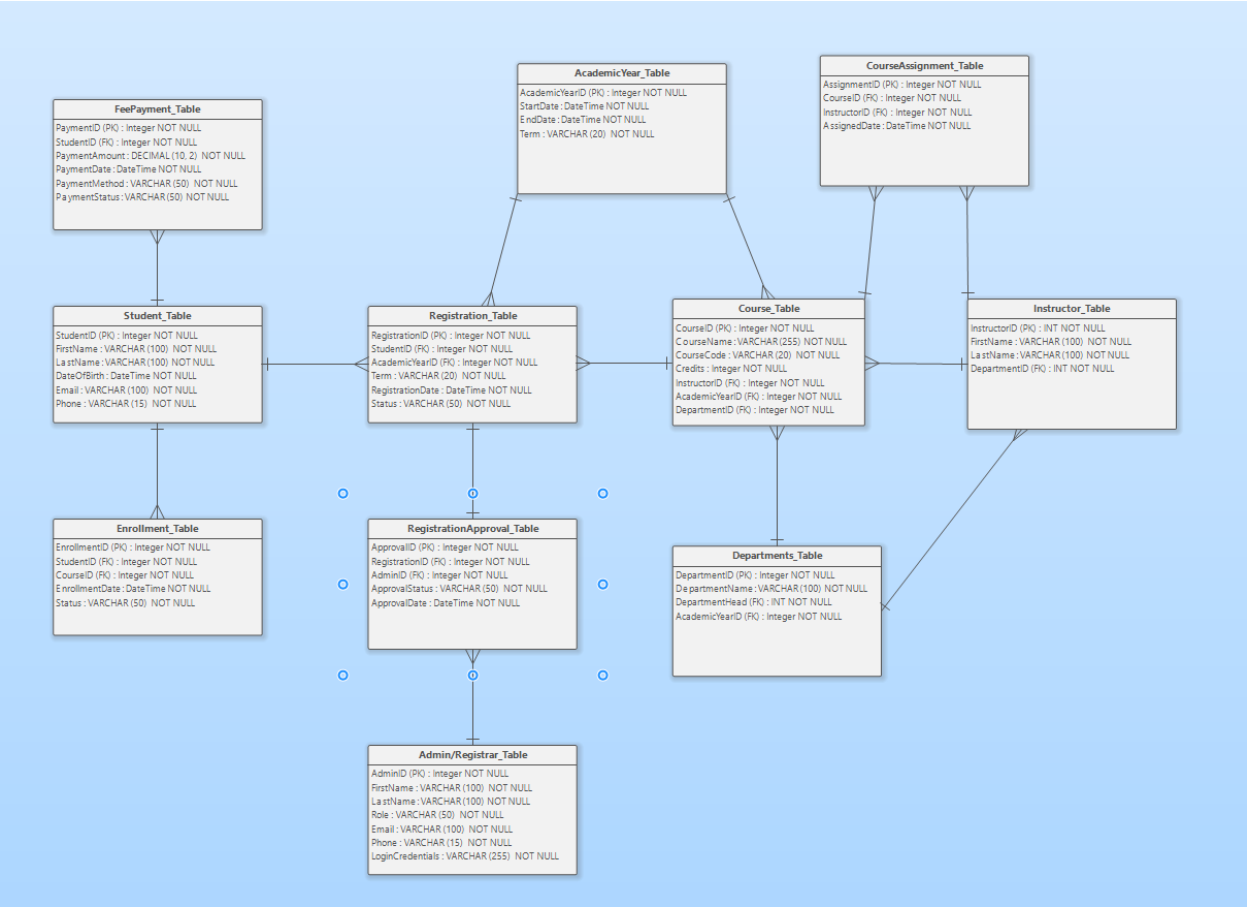
Cliffe, Matthew 000916002

Hao, Yichen 000938290

Thieu, Emily 000915665

Table Of Contents

Physical Model ERD



Physical Model Explanation

1. **Student Table**:
    - **StudentID** is the Primary Key.
    - **Foreign Key**: The StudentID is referenced in the **Registration**, **Enrollment**, and **Fee Payment** tables.
    - **One-to-Many** relationship with **Registration** (one student can have many registrations).
    - **One-to-Many** relationship with **Enrollment** (one student can be enrolled in many courses).
    - **One-to-Many** relationship with **Fee Payment** (one student can make many payments).
2. **Registration Table**:
    - **RegistrationID** is the Primary Key.

- **Foreign Keys**:
  - StudentID references Student.StudentID (relating registrations to specific students).
  - AcademicYearID references AcademicYear.AcademicYearID (connecting registrations to a specific academic year).
- **One-to-Many** relationship with **Student** (one student can have many registrations).
- **Many-to-One** relationship with **AcademicYear** (many registrations can belong to one academic year).

3. **Course Table**:
   - **CourseID** is the Primary Key.
   - **Foreign Keys**:
     - InstructorID references Instructor.InstructorID (associating each course with an instructor).
     - DepartmentID references Department.DepartmentID (linking each course to a department).
     - AcademicYearID references AcademicYear.AcademicYearID (linking courses to a specific academic year).
   - **One-to-Many** relationship with **Instructor** (one instructor can teach many courses).
   - **Many-to-One** relationship with **Department** (many courses can belong to one department).
   - **Many-to-One** relationship with **AcademicYear** (many courses can belong to one academic year).
4. **Enrollment Table**:
   - **EnrollmentID** is the Primary Key.
   - **Foreign Keys**:
     - StudentID references Student.StudentID (connecting the enrollment to a student).
     - CourseID references Course.CourseID (linking the enrollment to a specific course).
   - **One-to-Many** relationship with **Student** (one student can be enrolled in many courses).

- **One-to-Many** relationship with **Course** (one course can have many students enrolled).

5. **Admin/Registrar Table**:
   - **AdminID** is the Primary Key.
   - **Foreign Key**: AdminID in **RegistrationApproval** table.
   - Admins/Registrars are responsible for managing registrations and approvals.
6. **Fee Payment Table**:
   - **PaymentID** is the Primary Key.
   - **Foreign Key**: StudentID references Student.StudentID (linking payments to a student).
   - **Many-to-One** relationship with **Student** (many payments can be made by one student).
7. **Academic Year Table**:
   - **AcademicYearID** is the Primary Key.
   - **One-to-Many** relationship with **Course** (one academic year can have many courses).
   - **One-to-Many** relationship with **Registration** (one academic year can have many registrations).
   - **One-to-Many** relationship with **Course Assignment** (one academic year can have many assignments).
8. **Registration Approval Table**:
   - **ApprovalID** is the Primary Key.
   - **Foreign Keys**:
     - RegistrationID references Registration.RegistrationID (connecting the approval to a registration).
     - AdminID references Admin.AdminID (associating the approval with the admin).
   - **One-to-One** relationship with **Registration** (a registration can only have one approval).
9. **Course Assignment Table**:
   - **AssignmentID** is the Primary Key.
   - **Foreign Keys**:

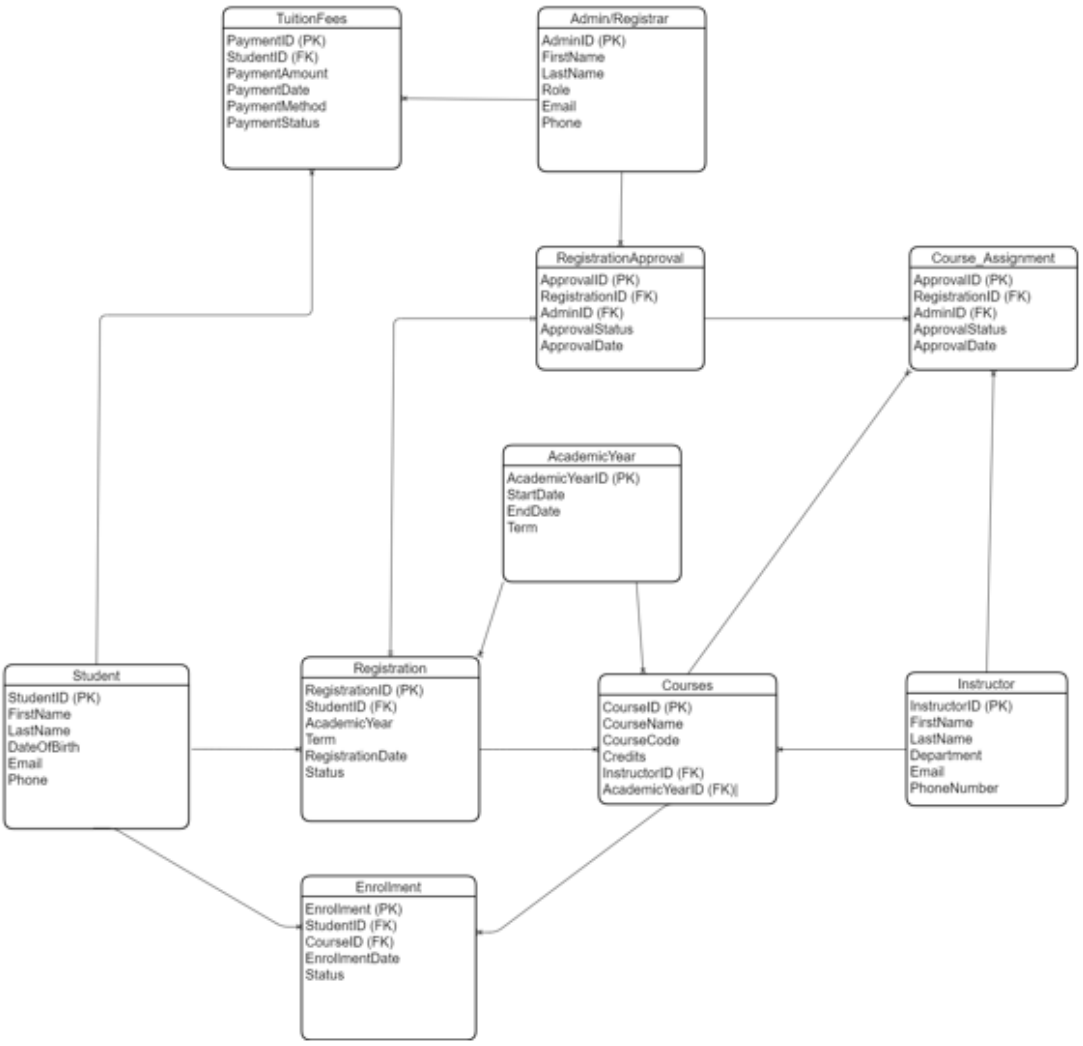
        - CourseID references Course.CourseID (linking the assignment to a course).
        - InstructorID references Instructor.InstructorID (associating the assignment with an instructor).
    - **One-to-Many** relationship with **Course** (one course can have many assignments).
    - **One-to-Many** relationship with **Instructor** (one instructor can have many assignments).

10. **Department Table**:
    - **DepartmentID** is the Primary Key.
    - **DepartmentName** is the name of the department (e.g., Computer Science, Mathematics).
    - **One-to-Many** relationship with **Course** (one department can offer many courses).
    - **Many-to-One** relationship with **Instructor** (many instructors can belong to one department).

11. **Instructor Table**:
    - **InstructorID** is the Primary Key.
    - **Foreign Key**: DepartmentID references Department.DepartmentID (associating an instructor with a specific department).
    - **One-to-Many** relationship with **Course** (one instructor can teach many courses).
    - **Many-to-One** relationship with **Department** (many instructors can belong to one department).

Logical Model ERD

LOGICAL MODEL DIAGRAM



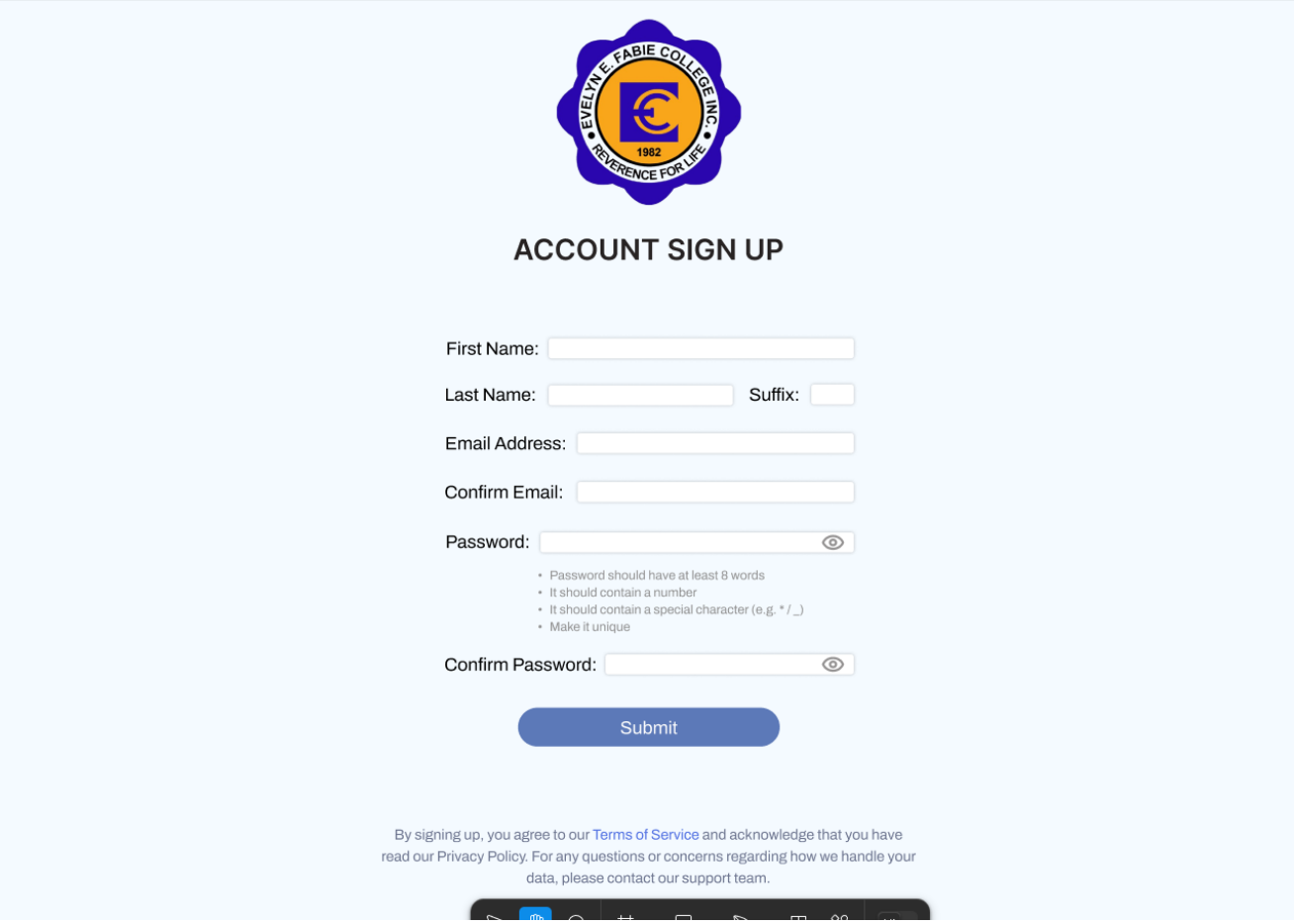
Logical Model Explanation

1. Student to Registration: One-to-many relationship (one student can have many registrations).
2. Course to Registration: One-to-many relationship (one course can have many registrations).
3. Student to Enrollment: One-to-many relationship (one student can be enrolled in many courses).
4. Course to Enrollment: One-to-many relationship (one course can have many enrolled students).

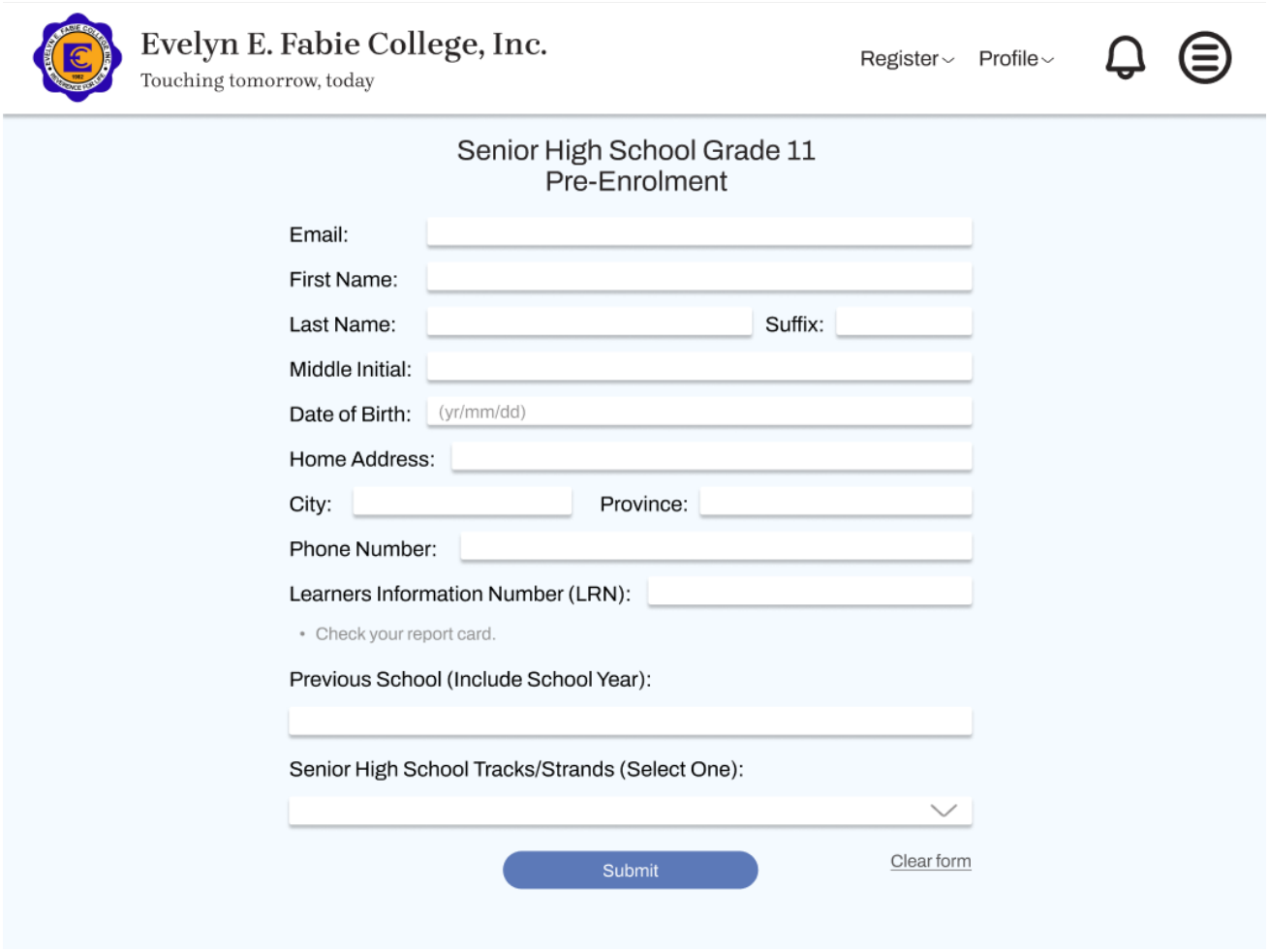
5. Admin/Registrar to Registration Approval: One-to-many relationship (an admin can approve many registrations).
6. Registration to Registration Approval: One-to-one relationship (a registration is approved only once).
7. Course to Course Assignment: One-to-many relationship (one course can have many assignments).
8. Instructor to Course Assignment: One-to-many relationship (one instructor can be assigned to many courses).
9. Fee Payment is linked to Student: Many-to-one relationship (many payments can be made by one student).
10. Course to Academic Year: Many-to-One
A Course belongs to one Academic Year (i.e., a course is taught in a specific academic year).
11. Registration to Academic Year: Many-to-One
Each Registration is tied to one Academic Year (indicating which academic year the registration pertains to).

Preliminary User Interface Design:

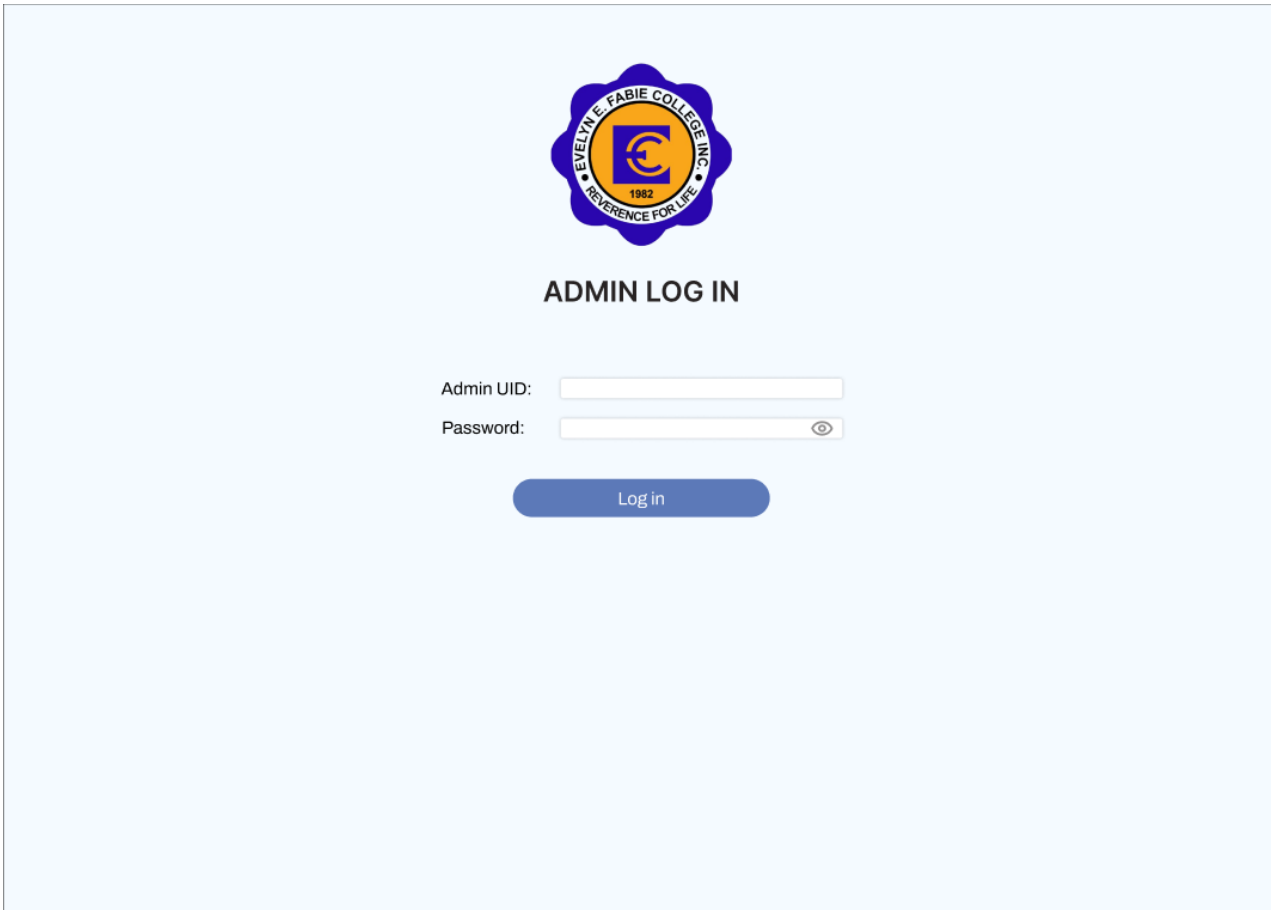
Student Sign-Up Page



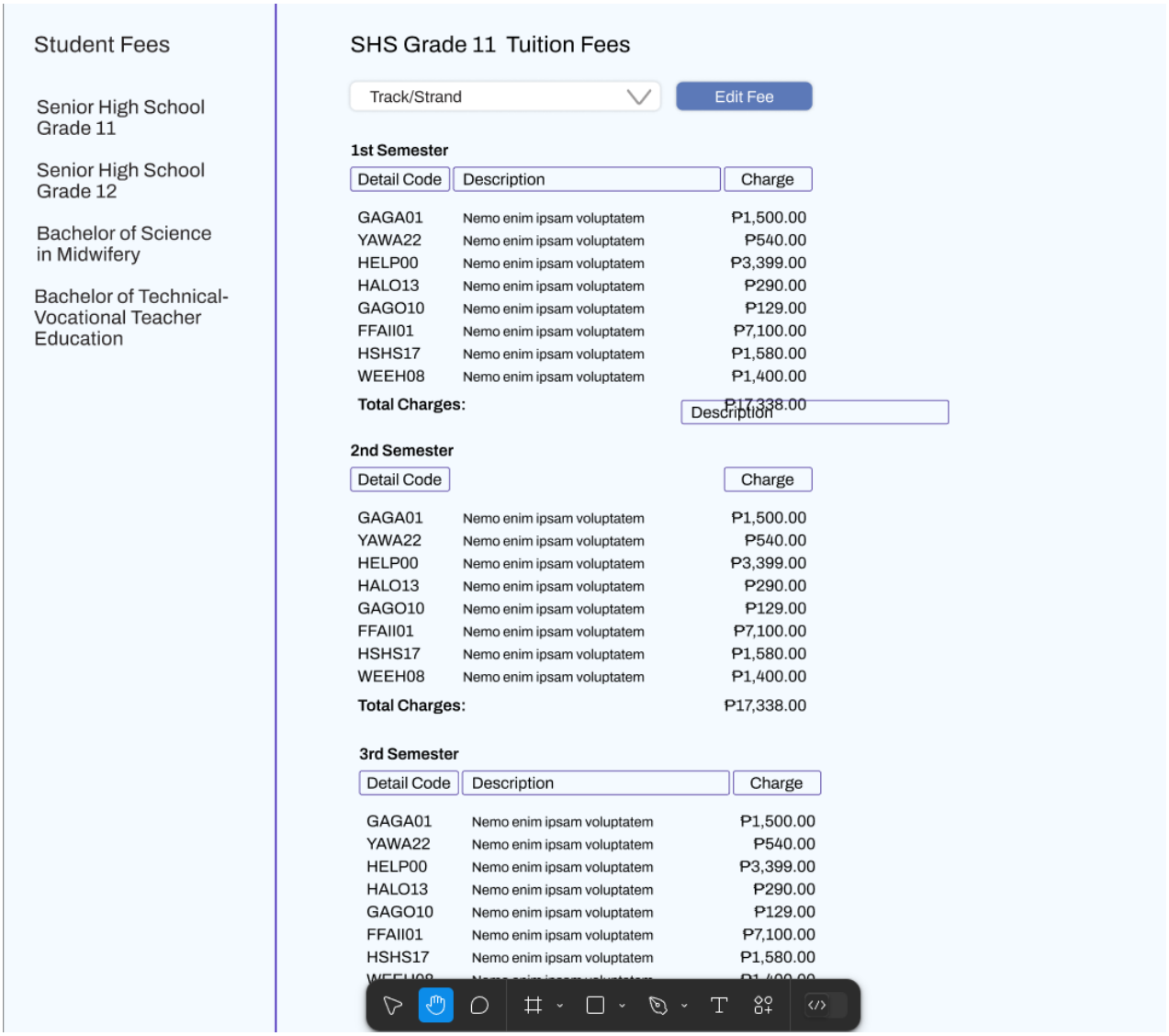
Pre-enrollment Page



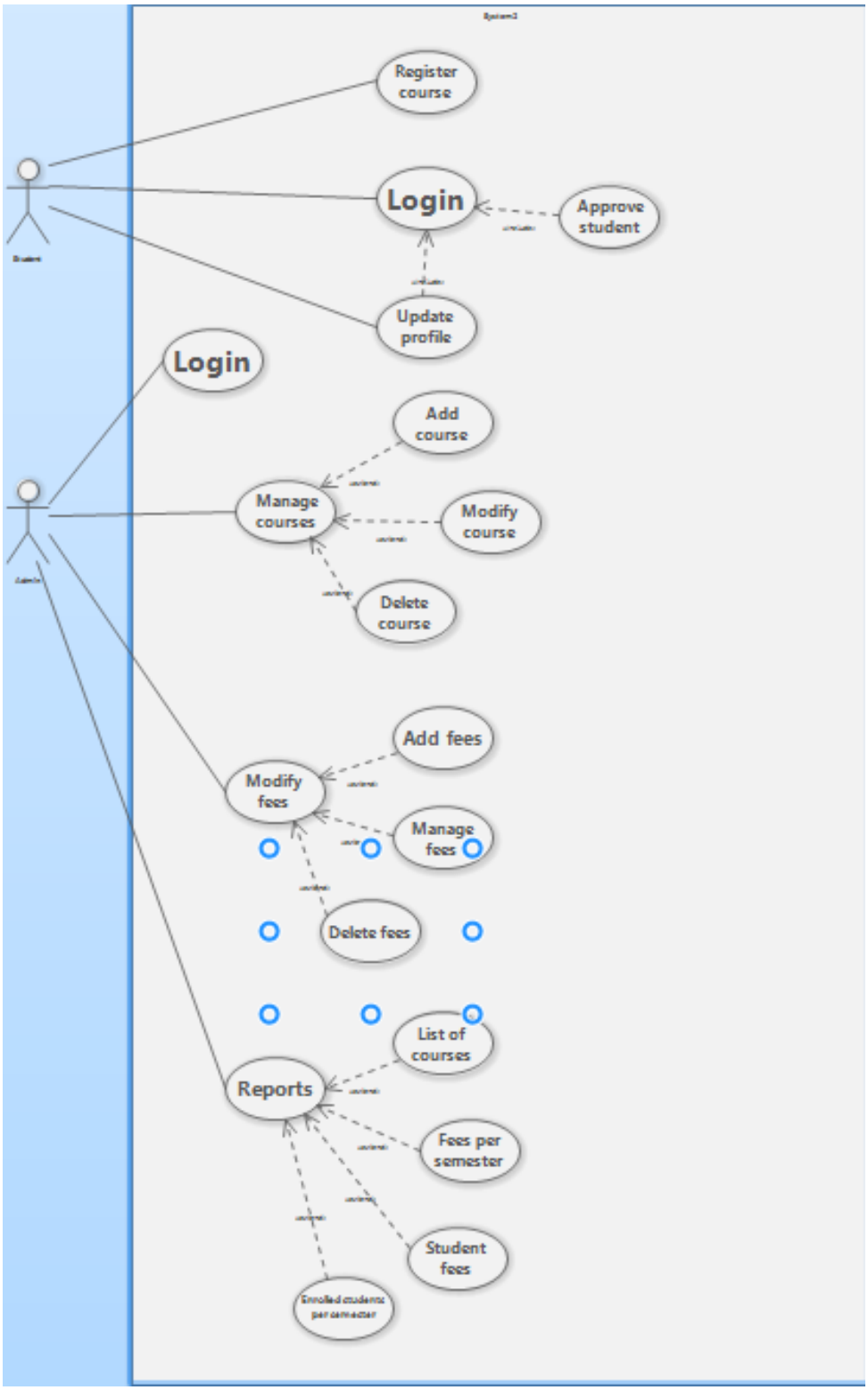
Admin Log In


EVELYN E. FABIE COLLEGE INC.
1982
REVERENCE FOR LIFE
ADMIN LOG IN
Admin UID:
Password:
Log in

Tuition Fee Page



Use Case Diagram



Team Voov Use Case Description

Name: Student Login

Description: Student logs onto account to register for courses

Actors: Student

Preconditions: Requires admin verification (for first signing up) and valid login credentials (username and password).

Postconditions: User is led to the dashboard where they can register for courses, and update student info in their profile.

Limitations: Limited access to the database and can only edit information related to the users themselves.

Triggers: Clicks on the login page

Main Flow:

- Navigate to Login Page: The student accesses the login page.
- Enter Credentials: The student inputs the username and password.
- Click on the "Log in" button.
- Credential Validation: The system verifies the username and password. If valid, the system grants access; if invalid, an error message is displayed.
- Access Dashboard: The student is led to the main dashboard of the system.

Alternative Flow:

- Invalid Login: Displays error message for invalid login info whether for a username or `password and prompts the user to try again.

Name: Manage fees

Description: The admin manages the student fees by adding, modifying, or deleting financial fee information

Actors: Admin

Preconditions: Requires admin to be verified and logged into the system and have access to the manage fee's function.

Postconditions: Can view, change, add, or delete fees to update the system.

Triggers: Clicks on the manage fees page

Main Flow:

- **Access Manage Fees page:** The admin navigates to the "Manage Fees" section from the dashboard.

- **Add Fees:**
  - The admin clicks on the "Add Fees" button.
  - The admin fills in the required info for student fees (tuition, supplies, etc.)
  - The admin submits the form.
  - The system confirms the addition and updates the fees list.
  - Several different student fee lists will be connected to different student users in the system.

- **Modify Courses:**
  - The admin selects an existing student fee to modify.
  - The admin edits the necessary financial details. (Setting it to 0 if the fee has been paid for example)
  - The admin submits the changes.
  - The system confirms the updates and refreshes the fees list.

- **Delete Courses:**
  - The admin selects a fee to delete.
  - The system prompts for confirmation.
  - The admin confirms deletion.
  - The system removes the fee from the list and updates the database.

Alternative Flow:

- Cancellation of Operations: The admin can cancel any add, modify, or delete operation, returning to the manage fees page without changes.

Name: Reports

Description: Gives data on the total fees, student amount, courses, and other necessary information to the admin.

Actors: Admin

Preconditions: Requires admin to be verified and logged into the system and have access to the reports function.

Postconditions: Can access, change, add, and delete information regarding payments and courses and make a report on it.

Triggers: Clicks on the reports page

Main Flow:

- **Access Reports page:** The admin navigates to the "Reports" section from the dashboard.
- **Add Report info:**
  - The admin clicks on the "Make report" button.
  - The admin fills in the required details (e.g., courses, student fees, student amount).
  - The admin submits the form.
  - The system confirms the addition and updates the reports list.
- **Modify report info:**
  - The admin selects an existing report to modify.
  - The admin edits the necessary report details.
  - The admin submits the changes.
  - The system confirms the updates and refreshes the reports list.
- **Delete report:**
  - The admin selects report info to delete.
  - The system prompts for confirmation.
  - The admin confirms deletion.
  - The system removes that specific info from the list and updates the database.

Alternative Flow:

- Cancellation of Operations: The admin can cancel any add, modify, or delete operation, returning to the reports management overview without changes.

Use Case Name: Admin login

Actor: Admin

Description: This use case allows an admin to securely log into the system to access administrative functionalities.

Precondition: The admin must have valid login credentials (username and password).

Postcondition: The admin is logged into the system and redirected to the dashboard.

Main Flow:

- Navigate to Login Page: The admin accesses the application login page.
- Enter Credentials: The admin inputs the username and password.
- **Submit Login:**
  - The admin clicks the "Log In" button.

- Credential Validation: The system verifies the credentials.

- If valid, the system grants access; if invalid, an error message is displayed.
- Access Dashboard: Upon successful login, the admin is redirected to the main dashboard.

Alternative Flow:

- Invalid Login Attempt: If the admin enters incorrect credentials, the system displays an error message and prompts for re-entry.

Use Case Name: Manage Courses

Actor: Admin

Description: This use case allows the admin to manage course offerings by adding, modifying, or deleting courses.

Precondition: The admin must be logged into the system and have access to the course management module.

Postcondition: Courses are updated in the system based on the admin's actions.

Main Flow:

- Access Course Management: The admin navigates to the "Manage Courses" section from the dashboard.
- **Add Courses:**
  - The admin clicks on the "Add Course" button.
  - The admin fills in the required course details (e.g., title, description, duration).
  - The admin submits the form.
  - The system confirms the addition and updates the course list.
- **Modify Courses:**
  - The admin selects an existing course to modify.
  - The admin edits the necessary course details.
  - The admin submits the changes.
  - The system confirms the updates and refreshes the course list.
- **Delete Courses:**
  - The admin selects a course to delete.
  - The system prompts for confirmation.
  - The admin confirms deletion.
  - The system removes the course from the list and updates the database.

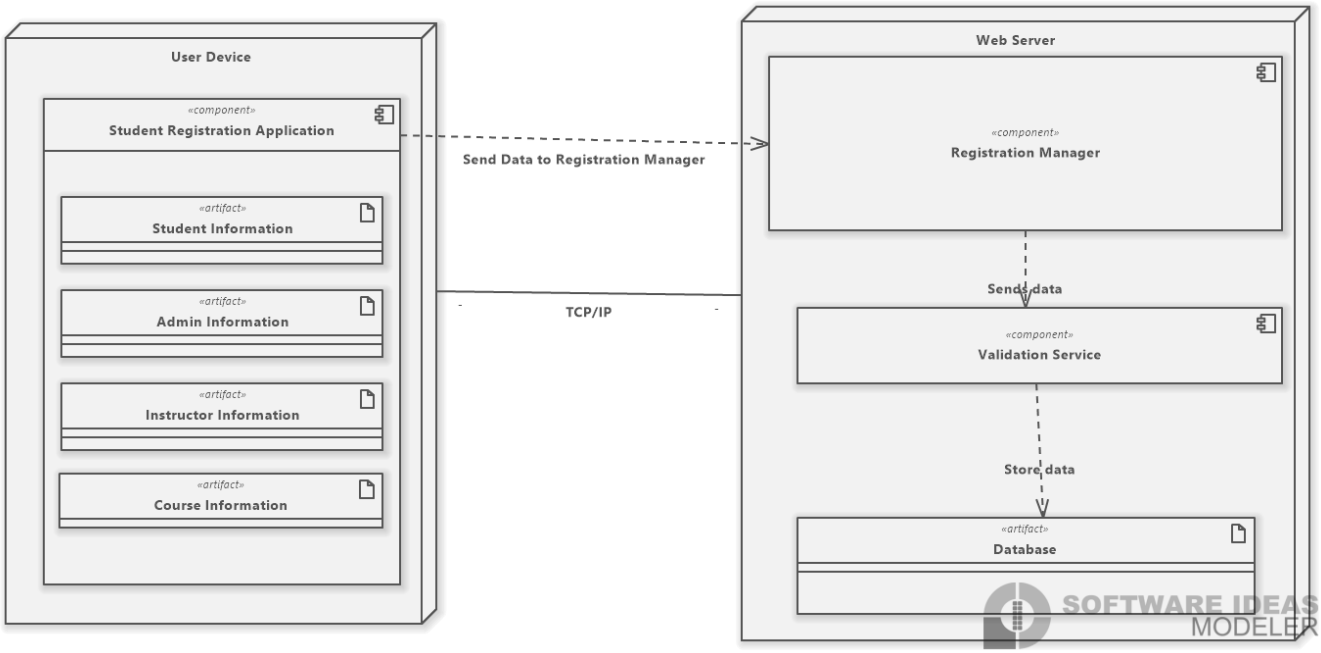
Alternative Flow:

- Cancellation of Operations: The admin can cancel any add, modify, or delete operation, returning to the course management overview without changes.

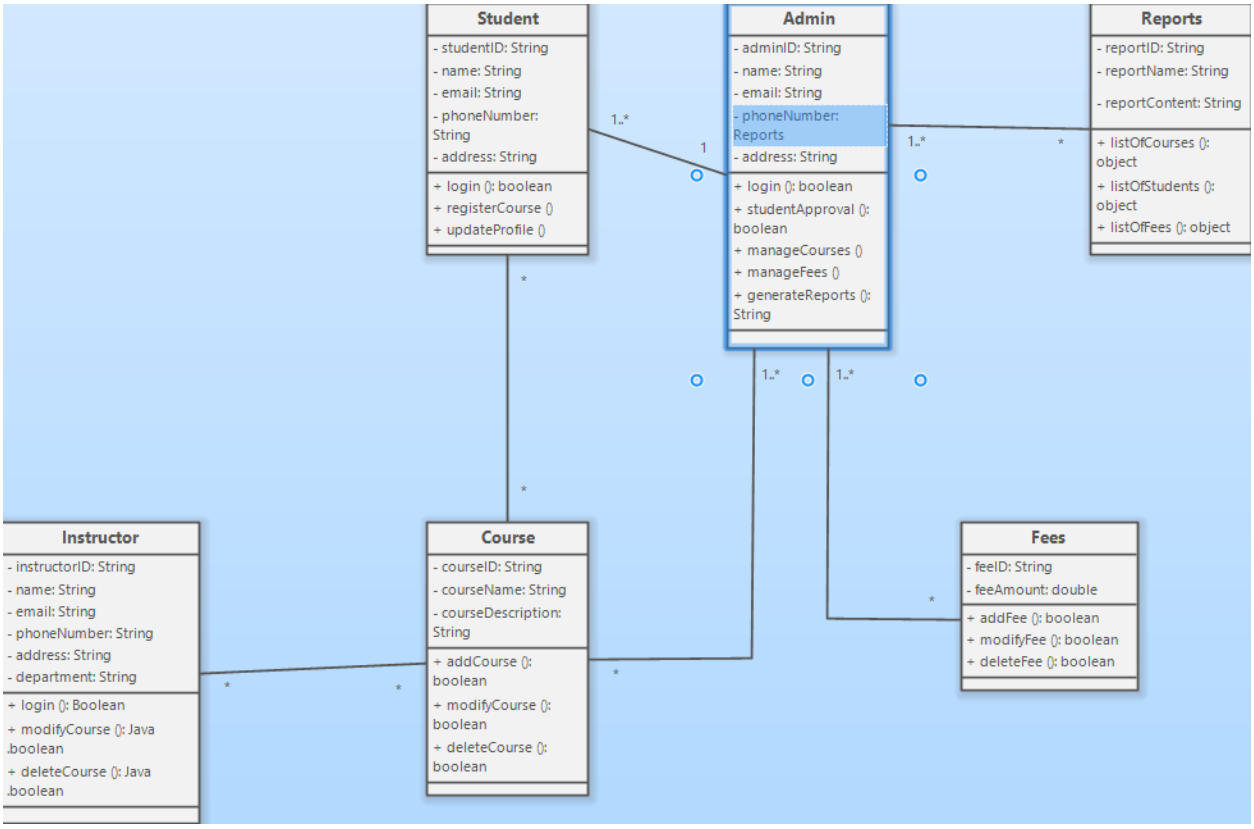
Exceptions:

- The system may display error messages if the admin attempts to add a course with duplicate titles or if there are issues during course management.

Deployment Diagram:

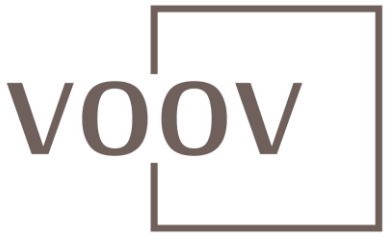


Class Diagram:



Team Contract

Team Name: Voov
Team Logo:



Purpose:
This agreement serves as a road map for our trip. The team contract serves the function of outlining each member's obligations and setting limits based on conversations and expectations.
Mission:
Our goal is to utilize our expertise and professional abilities to achieve a successful outcome.
Vision:
We aim to create a sustainable and impactful App/website/Application software that supports local businesses.
Team Goals

- To practice confidence and professionalism during the whole course of the project.
- Meet the course study deadlines with an outstanding grade not lower than 95%.
- Expand on our knowledge and skills through group problem solving

Skills we aim to gain through the process:

- Effective communication
- Cooperation and respect for each other
- Time management
- Being a team player and collaborator
- Flexibility in workspace

Team Member Contact Information

Team Member	Contact Info	Constraints
Mathew Cliffe	403.909.5846	N/A
	matthew.cliffe@edu.sait.ca	
Trisha Aquino	825.733.4339 /	None – use either Discord or Teams
	trishayvonne.aquino@edu.sait.ca	
Joseph Bui	825.437.5527 /	N/A
	joseph.bui@edu.sait.ca	

Katrina Keen Buenavista	403.827.7912 / katrinakeen.buenavista@edu.sait.ca	None – use either Discord, WhatsApp or Teams
Yichen Hao	587.429.5134 / yichen.hao1@edu.sait.ca	Cannot be reached Friday evenings
Emily Thieu	403-383-8739/ emily.thieu@edu.sait.ca	None – use either Discord or teams

Team Members with Professional Biography

Katrina Keen Buenavista

Educational Background	• Bachelor's in industrial technology major in Computer Technology
Employment History	• Coop/On-the-job-Training – IT helpdesk
Summary of Skills	• Programming Languages: • Front-end: React.js, next.js • Back-end: C++, C#, Python, HTML, CSS • Database: MySQL, non-SQL

Intellectual Property Statement

- Unauthorized use or disclosure of any intellectual property is strictly prohibited.
- **Voov** is committed to protecting its intellectual property rights and ensuring that they are used solely for the company's and its stakeholders' benefit.
- Team members should not share work and related intellectual property that belongs to the client without the client's prior written consent.

Team Member Role Assignment

Phase 1:

Name	Role	Description
Mathew Cliffe	Project Manager/Team Leader	• Project coordination and scheduling • Communication with team members and stakeholders • Managing deadlines and resources Organizes and manages team members
Emily Thieu, Trisha Aquino	Front-end developer and UI/UX designer	• Creating the structure and layout of the entire web application • Design and implement the user interface
Katrina Keen Buenavista	Database Designer	• Defining the detailed database design, including tables and other database-specific constructs • Connecting the database to the entire application
Yichen Hao, Joseph Bui	Back-End Developer	• Develop API for bridging Back-End & Front-End

Phase 2:

Name	Role	Description
Mathew Cliffe	Project Manager/Team Leader	• Project coordination and scheduling • Communication with team members and stakeholders • Managing deadlines and resources Organizes and manages team members
Emily Thieu, Trisha Aquino	Front-end developer and UI/UX designer	• Creating the structure and layout of the entire web application • Design and implement the user interface
Katrina Keen Buenavista, Trisha Aquino	Database Designer	• Defining the detailed database design, including tables and other database-specific constructs Connecting the database to the entire application
Yichen Hao	UI/UX designer, Back-End Developer	• Developed base use case and planning for UI
Matthew Cliffe, Joseph Bui	Back-End Developer	• Develop API for bridging Back-End & Front-End

Established Norms and Expectations

Conduct

It is required of every member to abide by the following rules. Penalties for breaking the written regulations may vary depending on the offense committed:

- If someone misses a meeting or class without warning, they will have 24 hours to explain. If the group determines that the absence is warranted, the offender will get a verbal warning.
- If a group member reaches a deadlock in decision-making and won't give up, a game of Uno (with stacking) will be used to decide the winner.
- It is required of members to attend meetings at the appointed time and day. If a member anticipates being late for the meeting, they should let everyone know. "On time" is defined as starting the meeting within ten to fifteen minutes of the stated start time.
- If a member will not be present at the planned meeting, they should let the other members know.
- Throughout the project, it is anticipated that each participant will treat the other with appropriate decorum and respect; if not, they will get a verbal warning.
- If any modifications should be made to the group members' tasks, they should always be informed. When an issue arises, don't be afraid to tell everyone right away so that a solution may be found.
- It is required for every member to finish allocated work on schedule. If necessary, extensions can be worked out.

Communication

- In-College / Video conference (primarily through MS Teams)
- Primary mode of communication: Teams
- Secondary mode of communication: Discord
- Normal working hours: 12:00 PM to 12:00 AM
- Required time to reply: Within the 5-hour period

Meeting Times

- Regular Meeting: Teams, Saturday @ 1-2:50 PM
- Optional Meeting #2: Teams, Saturday after class for 1-2 hours to deliver updates on the work progress
- Optional Meeting #3: Discord, Mondays and/or Thursdays at 8 PM

Decision-Making Process

- Major decisions will be made through consensus among team members.
- In disagreements, the project manager will facilitate discussions to resolve the issues.

Conflict Handling

Here are the lists of conflict-handling strategies with potential resolutions during the project:

- Establish regular check-ins and make sure that lines of communication are open and transparent to avoid misunderstandings and to keep everyone updated.
- After acknowledging that everyone has different preferences, decide on a cooperative workflow that combines everyone's skills while guaranteeing flexibility and compromise.
- Define fair job distribution and clear expectations. Make sure that the leader implements progress checks to ensure that all team members are held accountable.
- To establish an agreement, promote candid conversations and brainstorming sessions where all ideas are considered. Keep the project's goals front and center.

Expelling Plans

If a group member does not adhere to the rules stated on **Conduct**, they will be addressed according to their offense:

First Offense

The person responsible is given a verbal warning. He or she is required to provide each group member with their preferred beverage.

Second Offense

A formal apology letter is written and sent to every member of the group, along with a 20% deduction from the final project grade as a second warning. The offense, an admission of guilt for their behavior, and an apology should all be included in the letter.

Third Offense

Expulsion from the group.

Agreement

We all agree to follow this team contract throughout the CPSY-301 project and are all responsible for enforcing the rules outlined above.

Mathew Cliffe

Trisha Aquino

Yichen Hao

Emily Thieu

Joseph Bui

Katrina Keen Buenavista

360 Peer Feedback

Instructions

Each member of each group will fill out the following form.

The number of questions should be adjusted based on the group size. A sample form is available on the 2nd page of this document.

Collect the feedback from all teammates and calculate the average rating per student.

VITAL: THESE ARE NOT “FREE GRADES”. Your final grade may be impacted by this peer assessment. Your un-adjusted grade will be the final assessed grade of the project. If your individual contribution was deemed fair, you will be assigned the project grade. Your grade MAY be adjusted upwards if your contribution was deemed greater than your peers. Your grade MAY be decreased INCLUDING FAILURE if your contribution was deemed substantially less than your peers.

Throughout the course, your instructors will note how the team has been working together and monitoring attendance.

Group #: 4 Your name: Group 4

1. the following scale:
    - 5 stars - Contributed to every part of the assessment
    - 4 stars - Actively contributed to most parts of the assessment
    - 3 stars - Contributed about 50% to the assessment but was not actively participating in all parts of the assessment
    - 2 stars - Was present when the group was working on the assessment, but the participation was minimal
    - 1 star - Was not present when the group worked on any part of the assessment

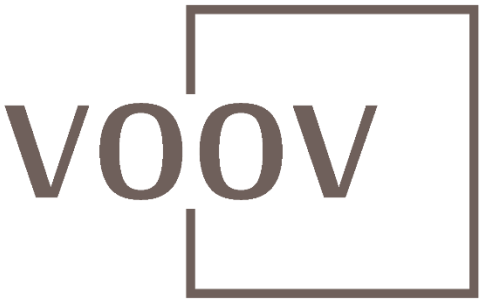
Peer review	Rating	Notes
Yichen Hao	**4**	**Fixed class diagram and completed use case description**
Emily Thieu	**5**	**Worked with Trisha on the Preliminary User Interface Design**
Trisha Aquino	**5**	**Worked with Emily on the Preliminary User Interface Design and helped Katrina on the Physical ERD**

Matthew Cliffe	**5**	**Fixed class diagram and collaborated with Joseph with the deployment diagram**
Katrina Buenavista	**5**	**Collaborated with Trisha on physical ERD and created the logical ERD**
Joseph Bui	**4**	**Collaborated with Matthew on the deployment diagram**

3. Is there anything else I should know regarding your team dynamics or anyone's contribution?

- Everyone was cooperative when doing their assigned work. There were a few misunderstandings but was managed to communicate it within the team.

Voov: Project Phase 3



Team Voov

Aquino, Trisha 000916516

Buenavista, Katrina Keen 000922346

Bui, Joseph 000927242

Cliffe, Matthew 000916002

Hao, Yichen 000938290

Thieu, Emily 000915665

External Client Information:

Name: Dino Fabie

Position: Head Registrar Administrator

Contact Details:

Phone Number: +63822273469 (school administration)

Email Address: evelynefabiecollegeinc@gmail.com

Name: Cecile Fabie

Position: Head Administrator

Contact Details:

Phone Number: +639453191209

Email Address: evelynefabiecollegeinc@gmail.com

Project Topic: School Management System (SMS)

Description: In today's digital age, managing student registrations and related administrative tasks efficiently is crucial for educational institutions. This document outlines the development of a comprehensive School Management System (SMS) focusing primarily on streamlining student registration processes and providing an intuitive website interface through the use of REACT and SQL. The system aims to improve operational efficiency, enhance user experience, and support the institution's administrative needs.

Current System:

- Excel files for data organization
- Facebook for communication and school information
- Google Forms for student registration

Proposed System:

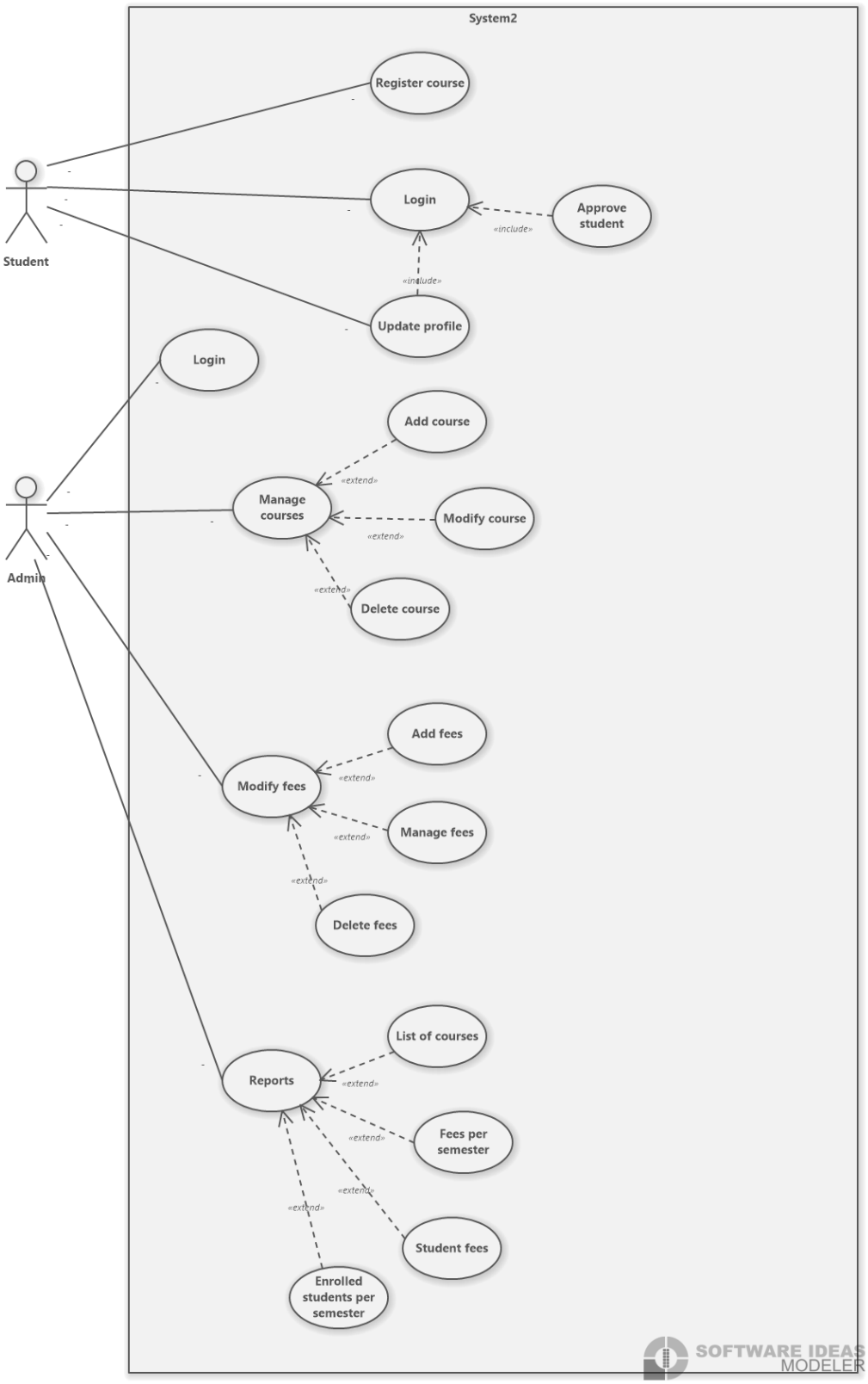
- **Online Registration Form:** An easy-to-fill, online registration form for new students.
- **Document Upload:** Secure upload feature for necessary documents (e.g., birth certificate, previous school records).
- **Automated Verification:** Automated system for verifying submitted documents and data.
- **Status Tracking:** Real-time status updates on registration progress.
- **Notifications:** Automated email/SMS notifications to inform students and parents of registration status and required actions.
- **User Accounts:** Secure login for students, parents, and administrators.
- **Dashboard:** Personalized dashboards displaying relevant information (e.g., registration status, upcoming deadlines).
- **Information Portal:** Access to school policies, academic calendars, and contact information.
- **Interactive Forms:** Forms for additional requests, feedback, and communication with the school.
- **Event Calendar:** Display upcoming events, deadlines, and school activities.
- **Responsive Design:** Mobile-friendly design to ensure accessibility on various devices.
- **Data Management:** Tools for administrators to view, edit, and manage student records.

- **Reporting:** Generate reports on registration statistics, student demographics, and other key metrics.
- **User Management:** Control access levels and permissions for different user roles (e.g., administrators, teachers).

Advantages of Proposed System:

The proposed School Management System aims to revolutionize the student registration process and enhance the overall user experience through a modern, web-based platform. By implementing this system, the institution will achieve greater efficiency, better data management, and improved communication channels.

Use Case Diagram



Team Voov Use Case Description

Name: Student Login

Description: Student logs onto account to register for courses

Actors: Student

Preconditions: Requires admin verification (for first signing up) and valid login credentials (username and password).

Postconditions: User is led to the dashboard where they can register for courses, and update student info in their profile.

Limitations: Limited access to the database and can only edit information related to the users themselves.

Triggers: Clicks on the login page

Main Flow:

- Navigate to Login Page: The student accesses the login page.
- Enter Credentials: The student inputs the username and password.
- Click on the "Log in" button.
- Credential Validation: The system verifies the username and password. If valid, the system grants access; if invalid, an error message is displayed.
- Access Dashboard: The student is led to the main dashboard of the system.

Alternative Flow:

- Invalid Login: Displays error message for invalid login info whether for a username or `password and prompts the user to try again.

Name: Manage fees

Description: The admin manages the student fees by adding, modifying, or deleting financial fee information

Actors: Admin

Preconditions: Requires admin to be verified and logged into the system and have access to the manage fee's function

Postconditions: Can view, change, add, or delete fees to update into the system.

Triggers: Clicks on the manage fees page

Main Flow:

- **Access Manage Fees page:** The admin navigates to the "Manage Fees" section from the dashboard.

- **Add Fees:**

  - The admin clicks on the "Add Fees" button.
  - The admin fills in the required info for student fees (tuition, supplies, etc.)
  - The admin submits the form.
  - The system confirms the addition and updates the fees list.
  - Several different student fee lists will be connected to different student users in the system.

- **Modify Courses:**

  - The admin selects an existing student fee to modify.
  - The admin edits the necessary financial details. (Setting it to 0 if the fee has been paid for example)
  - The admin submits the changes.
  - The system confirms the updates and refreshes the fees list.

- **Delete Courses:**

  - The admin selects a fee to delete.
  - The system prompts for confirmation.
  - The admin confirms deletion.
  - The system removes the fee from the list and updates the database.

Alternative Flow:

- Cancellation of Operations: The admin can cancel any add, modify, or delete operation, returning to the manage fees page without changes.

Name: Reports

Description: Gives data on the total fees, student amount, courses, and other necessary information to the admin.

Actors: Admin

Preconditions: Requires admin to be verified and logged into the system and have access to the reports function.

Postconditions: Can access, change, add, and delete information regarding payments and courses and make a report on it.

Triggers: Clicks on the reports page

Main Flow:

- **Access Reports page:** The admin navigates to the "Reports" section from the dashboard.
    - **Add Report info:**
    - The admin clicks on the "Make report" button.
    - The admin fills in the required details (e.g., courses, student fees, student amount).
    - The admin submits the form.
    - The system confirms the addition and updates the reports list.
- **Modify report info:**
    - The admin selects an existing report to modify.
    - The admin edits the necessary report details.
    - The admin submits the changes.
    - The system confirms the updates and refreshes the reports list.
- **Delete report:**
    - The admin selects report info to delete.
    - The system prompts for confirmation.
    - The admin confirms deletion.
    - The system removes that specific info from the list and updates the database.

Alternative Flow:

- Cancellation of Operations: The admin can cancel any add, modify, or delete operation, returning to the reports management overview without changes.

Use Case Name: Admin login

Actor: Admin

Description: This use case allows an admin to securely log into the system to access administrative functionalities.

Precondition: The admin must have valid login credentials (username and password).

Postcondition: The admin is logged into the system and redirected to the dashboard.

Main Flow:

- Navigate to Login Page: The admin accesses the application login page.
- Enter Credentials: The admin inputs the username and password.
- **Submit Login:**

- The admin clicks the "Log In" button.
- Credential Validation: The system verifies the credentials.

- If valid, the system grants access; if invalid, an error message is displayed.
- Access Dashboard: Upon successful login, the admin is redirected to the main dashboard.

Alternative Flow:

- Invalid Login Attempt: If the admin enters incorrect credentials, the system displays an error message and prompts for re-entry.

Use Case Name: Manage Courses

Actor: Admin

Description: This use case allows the admin to manage course offerings by adding, modifying, or deleting courses.

Precondition: The admin must be logged into the system and have access to the course management module.

Postcondition: Courses are updated in the system based on the admin's actions.

Main Flow:

- Access Course Management: The admin navigates to the "Manage Courses" section from the dashboard.
- **Add Courses:**
  - The admin clicks on the "Add Course" button.
  - The admin fills in the required course details (e.g., title, description, duration).
  - The admin submits the form.
  - The system confirms the addition and updates the course list.
- **Modify Courses:**
  - The admin selects an existing course to modify.
  - The admin edits the necessary course details.
  - The admin submits the changes.
  - The system confirms the updates and refreshes the course list.
- **Delete Courses:**
  - The admin selects a course to delete.
  - The system prompts for confirmation.

- The admin confirms deletion.
- The system removes the course from the list and updates the database.

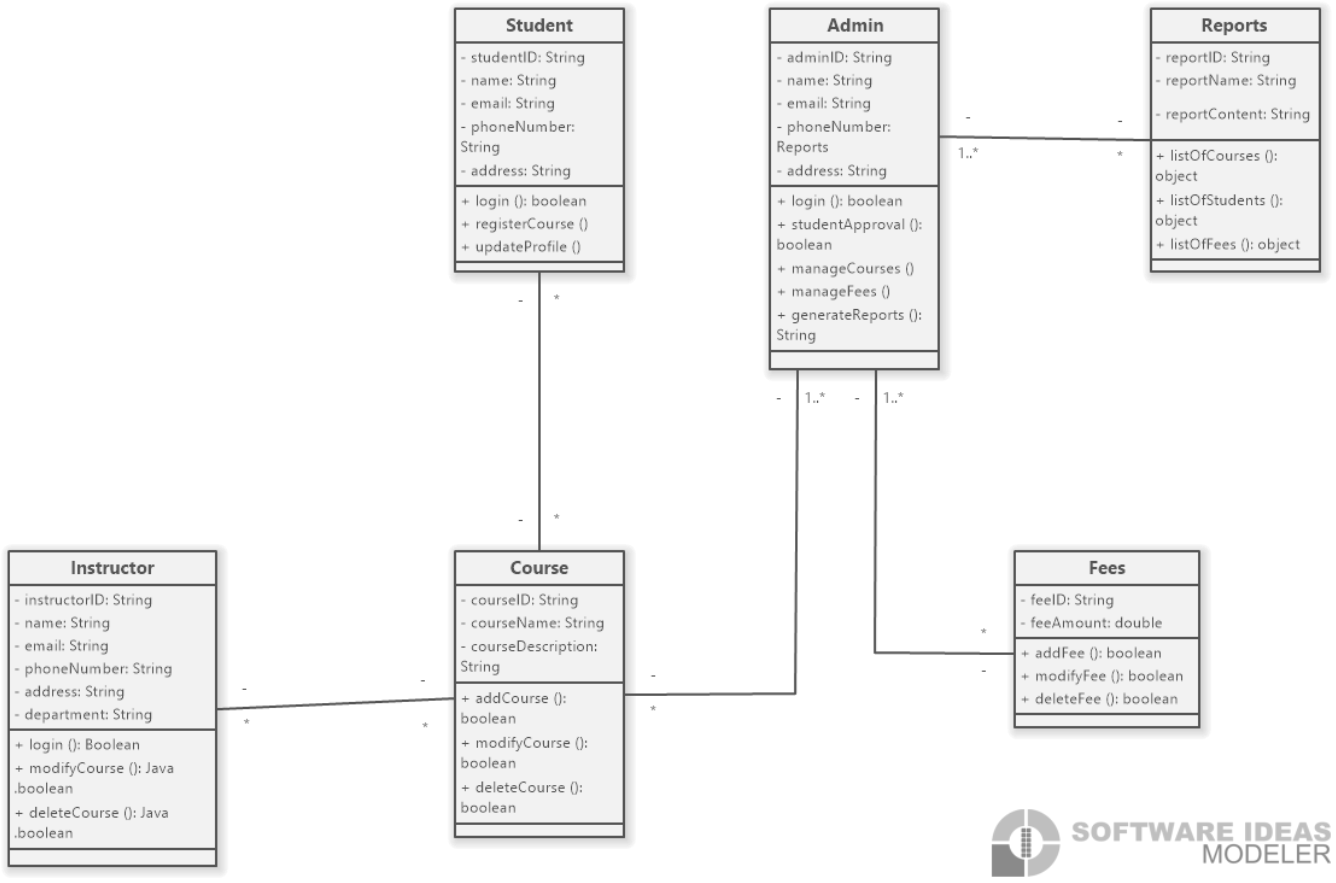
Alternative Flow:

- Cancellation of Operations: The admin can cancel any add, modify, or delete operation, returning to the course management overview without changes.

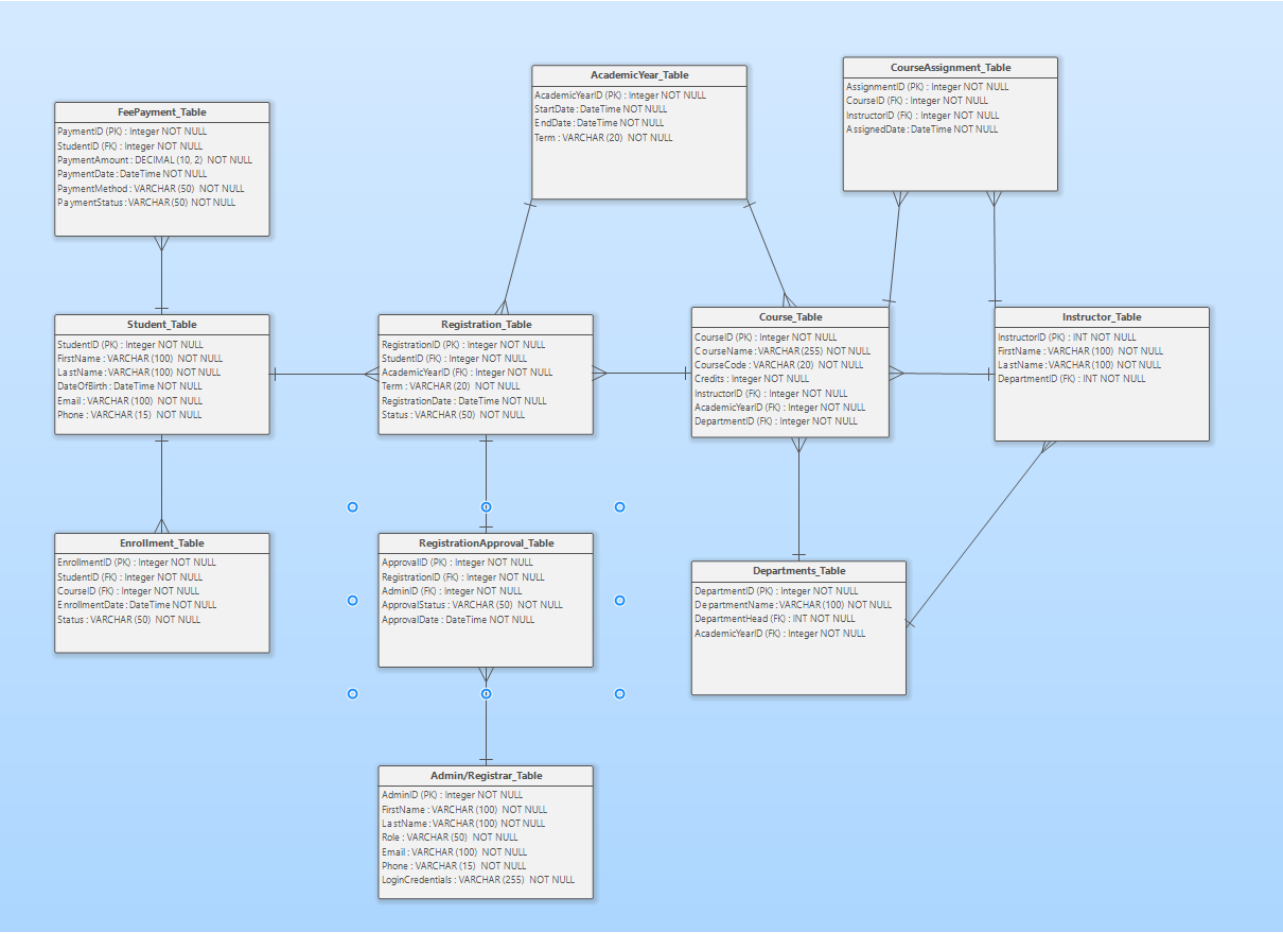
Exceptions:

- The system may display error messages if the admin attempts to add a course with duplicate titles or if there are issues during course management.

Class Diagram:



Physical Model ERD



Physical Model Explanation

12. **Student Table**:
    - **StudentID** is the Primary Key.
    - **Foreign Key**: The StudentID is referenced in the **Registration**, **Enrollment**, and **Fee Payment** tables.
    - **One-to-Many** relationship with **Registration** (one student can have many registrations).
    - **One-to-Many** relationship with **Enrollment** (one student can be enrolled in many courses).
    - **One-to-Many** relationship with **Fee Payment** (one student can make many payments).
13. **Registration Table**:

- **RegistrationID** is the Primary Key.
- **Foreign Keys**:
  - StudentID references Student.StudentID (relating registrations to specific students).
  - AcademicYearID references AcademicYear.AcademicYearID (connecting registrations to a specific academic year).
- **One-to-Many** relationship with **Student** (one student can have many registrations).
- **Many-to-One** relationship with **AcademicYear** (many registrations can belong to one academic year).

14. **Course Table**:
    - **CourseID** is the Primary Key.
    - **Foreign Keys**:
      - InstructorID references Instructor.InstructorID (associating each course with an instructor).
      - DepartmentID references Department.DepartmentID (linking each course to a department).
      - AcademicYearID references AcademicYear.AcademicYearID (linking courses to a specific academic year).
    - **One-to-Many** relationship with **Instructor** (one instructor can teach many courses).
    - **Many-to-One** relationship with **Department** (many courses can belong to one department).
    - **Many-to-One** relationship with **AcademicYear** (many courses can belong to one academic year).
15. **Enrollment Table**:
    - **EnrollmentID** is the Primary Key.
    - **Foreign Keys**:
      - StudentID references Student.StudentID (connecting the enrollment to a student).
      - CourseID references Course.CourseID (linking the enrollment to a specific course).
    - **One-to-Many** relationship with **Student** (one student can be enrolled in many courses).

- **One-to-Many** relationship with **Course** (one course can have many students enrolled).

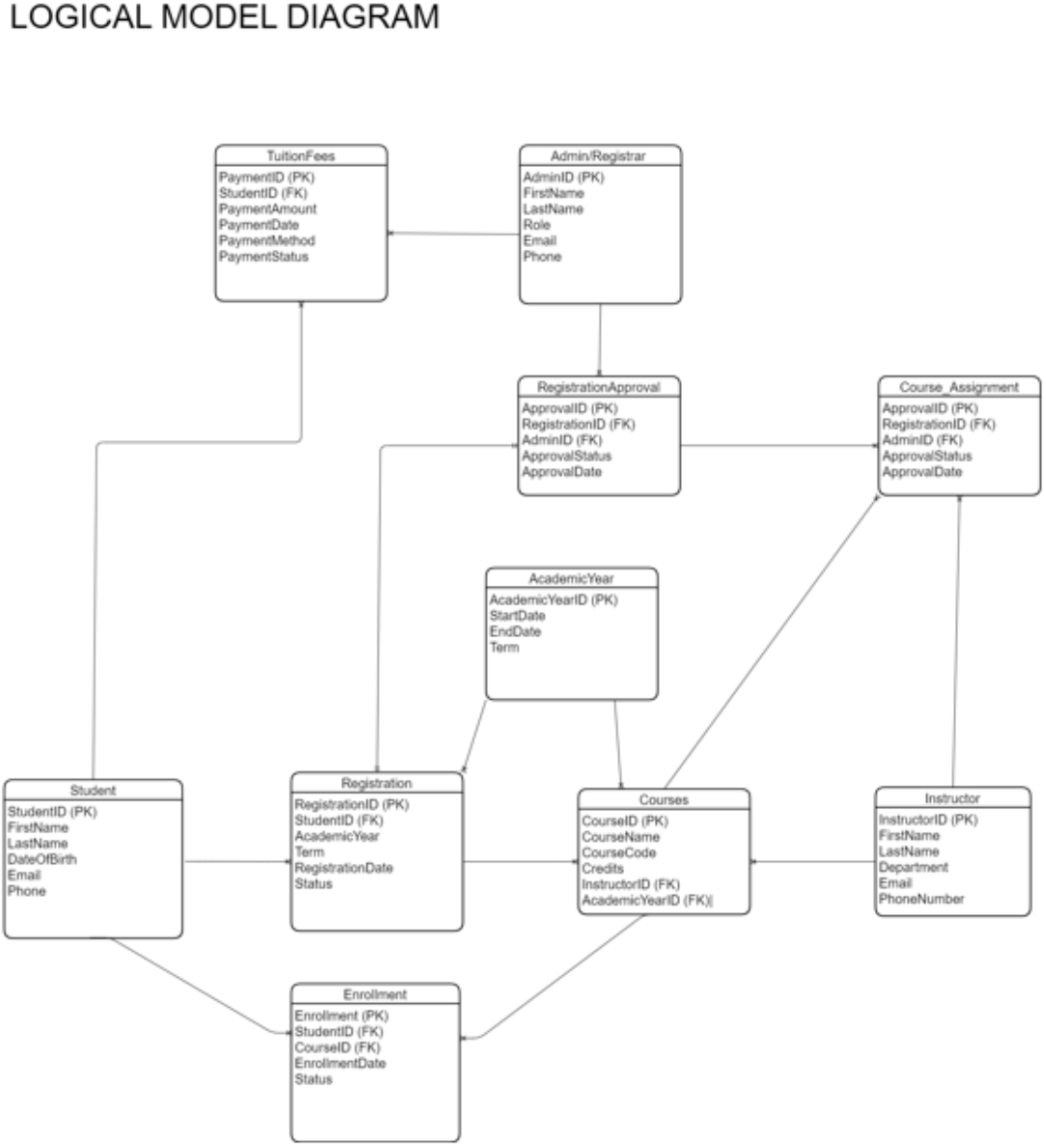
16. **Admin/Registrar Table**:
    - **AdminID** is the Primary Key.
    - **Foreign Key**: AdminID in **RegistrationApproval** table.
    - Admins/Registrars are responsible for managing registrations and approvals.
17. **Fee Payment Table**:
    - **PaymentID** is the Primary Key.
    - **Foreign Key**: StudentID references Student.StudentID (linking payments to a student).
    - **Many-to-One** relationship with **Student** (many payments can be made by one student).
18. **Academic Year Table**:
    - **AcademicYearID** is the Primary Key.
    - **One-to-Many** relationship with **Course** (one academic year can have many courses).
    - **One-to-Many** relationship with **Registration** (one academic year can have many registrations).
    - **One-to-Many** relationship with **Course Assignment** (one academic year can have many assignments).
19. **Registration Approval Table**:
    - **ApprovalID** is the Primary Key.
    - **Foreign Keys**:
        - RegistrationID references Registration.RegistrationID (connecting the approval to a registration).
        - AdminID references Admin.AdminID (associating the approval with the admin).
    - **One-to-One** relationship with **Registration** (a registration can only have one approval).
20. **Course Assignment Table**:
    - **AssignmentID** is the Primary Key.
    - **Foreign Keys**:

        - CourseID references Course.CourseID (linking the assignment to a course).
        - InstructorID references Instructor.InstructorID (associating the assignment with an instructor).
    - **One-to-Many** relationship with **Course** (one course can have many assignments).
    - **One-to-Many** relationship with **Instructor** (one instructor can have many assignments).

21. **Department Table**:
    - **DepartmentID** is the Primary Key.
    - **DepartmentName** is the name of the department (e.g., Computer Science, Mathematics).
    - **One-to-Many** relationship with **Course** (one department can offer many courses).
    - **Many-to-One** relationship with **Instructor** (many instructors can belong to one department).

22. **Instructor Table**:
    - **InstructorID** is the Primary Key.
    - **Foreign Key**: DepartmentID references Department.DepartmentID (associating an instructor with a specific department).
    - **One-to-Many** relationship with **Course** (one instructor can teach many courses).
    - **Many-to-One** relationship with **Department** (many instructors can belong to one department).

Logical Model ERD



Logical Model Explanation

1. Student to Registration: One-to-many relationship (one student can have many registrations).

2. Course to Registration: One-to-many relationship (one course can have many registrations).

3. Student to Enrollment: One-to-many relationship (one student can be enrolled in many courses).

4. Course to Enrollment: One-to-many relationship (one course can have many enrolled students).

5. Admin/Registrar to Registration Approval: One-to-many relationship (an admin can approve many registrations).

6. Registration to Registration Approval: One-to-one relationship (a registration is approved only once).

7. Course to Course Assignment: One-to-many relationship (one course can have many assignments).

8. Instructor to Course Assignment: One-to-many relationship (one instructor can be assigned to many courses).

9. Fee Payment is linked to Student: Many-to-one relationship (many payments can be made by one student).

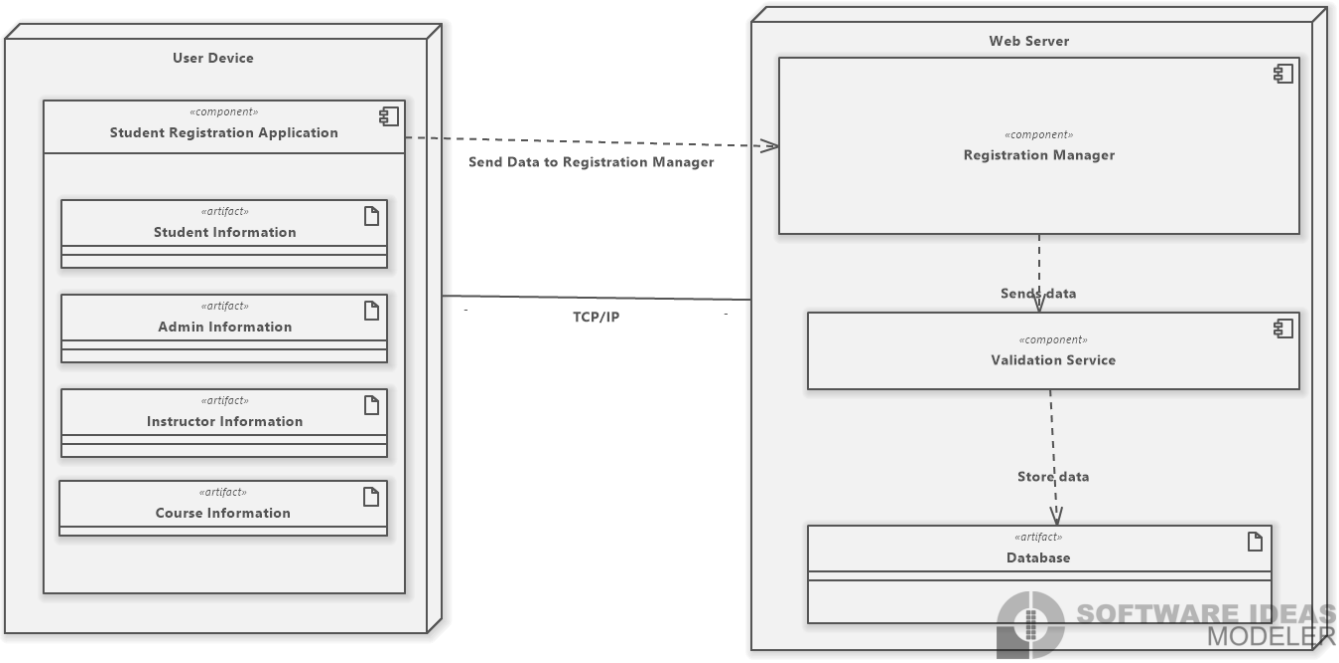
10. Course to Academic Year: Many-to-One

A Course belongs to one Academic Year (i.e., a course is taught in a specific academic year).

11. Registration to Academic Year: Many-to-One

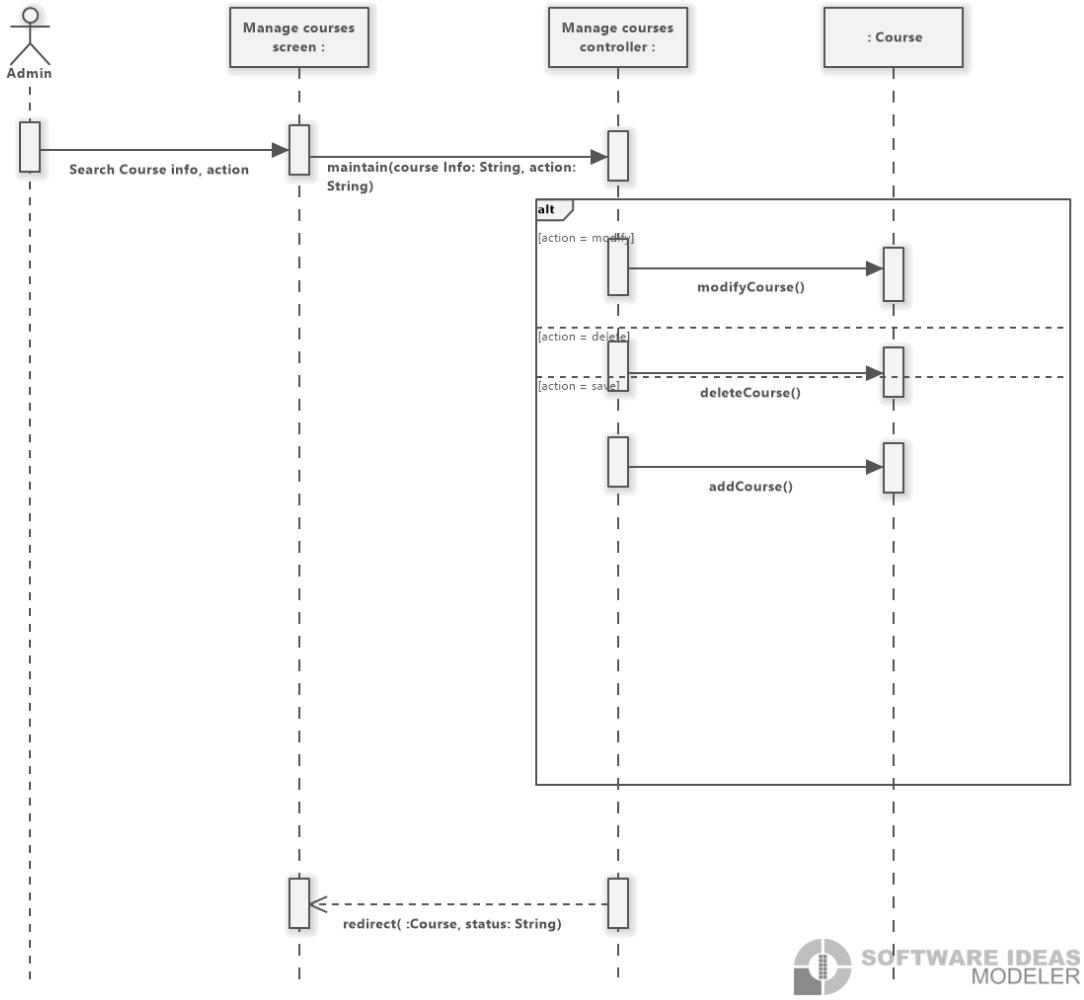
Each Registration is tied to one Academic Year (indicating which academic year the registration pertains to).

Deployment Diagram:

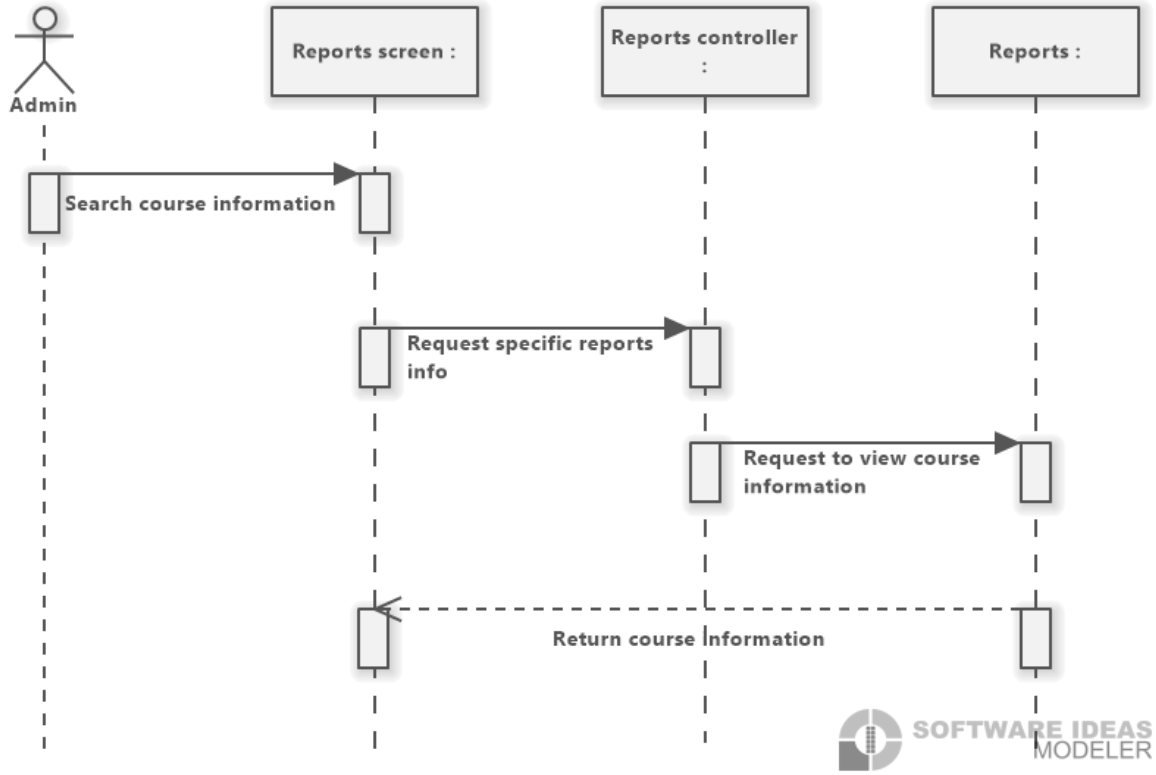

User Device
«component»
Student Registration Application
«artifact»
Student Information
«artifact»
Admin Information
«artifact»
Instructor Information
«artifact»
Course Information
Send Data to Registration Manager
TCP/IP
Web Server
«component»
Registration Manager
Send data
«component»
Validation Service
Store data
«artifact»
Database
SOFTWARE IDEAS
MODELER

Sequence Diagrams:

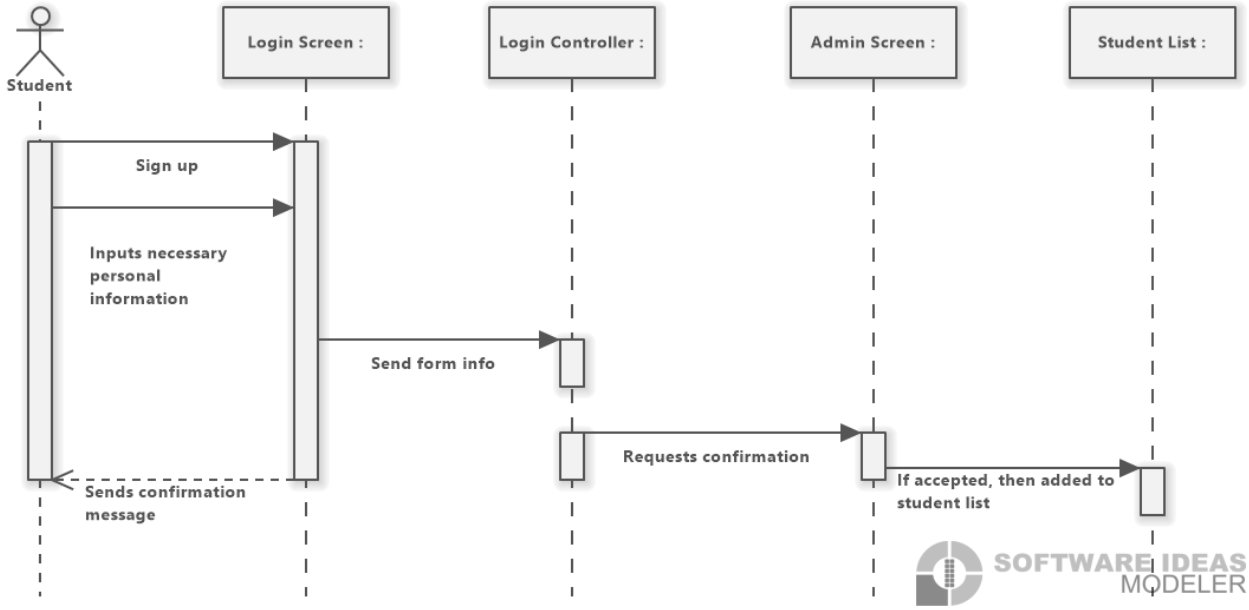
Manage Courses



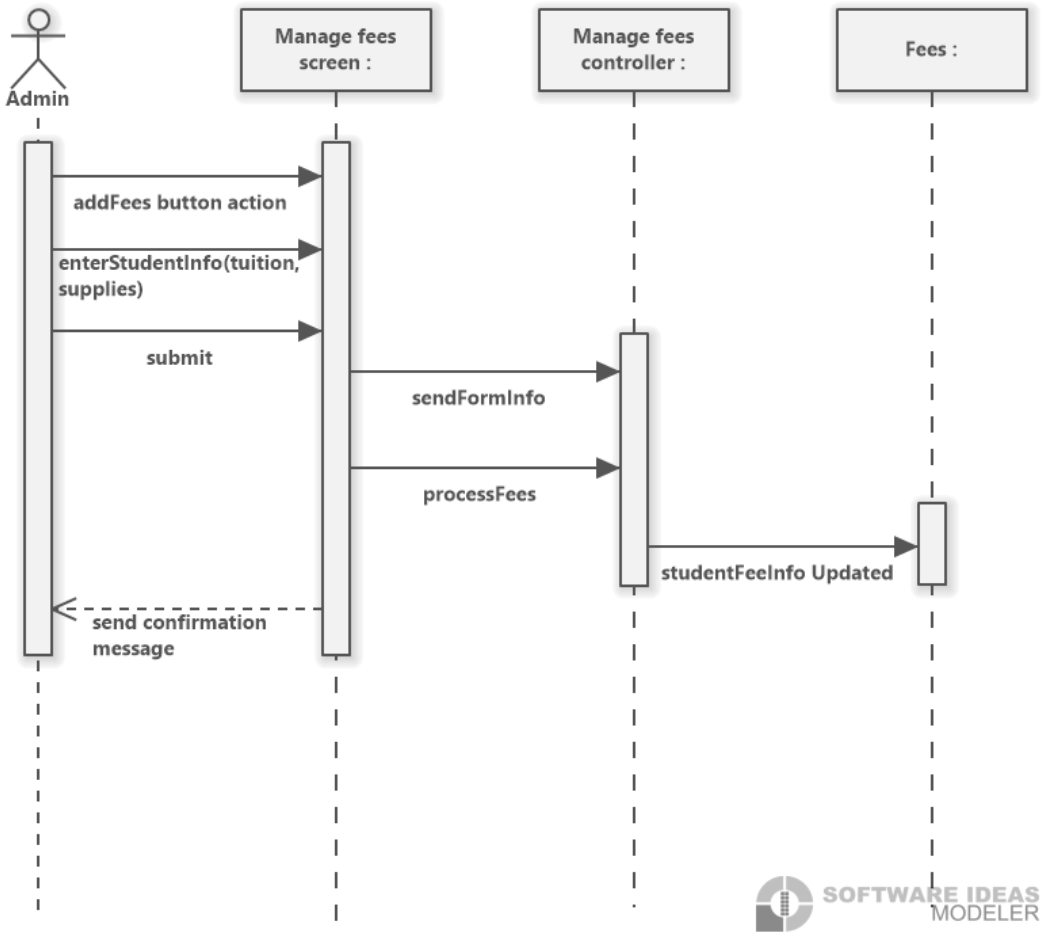
View Reports



Student Login

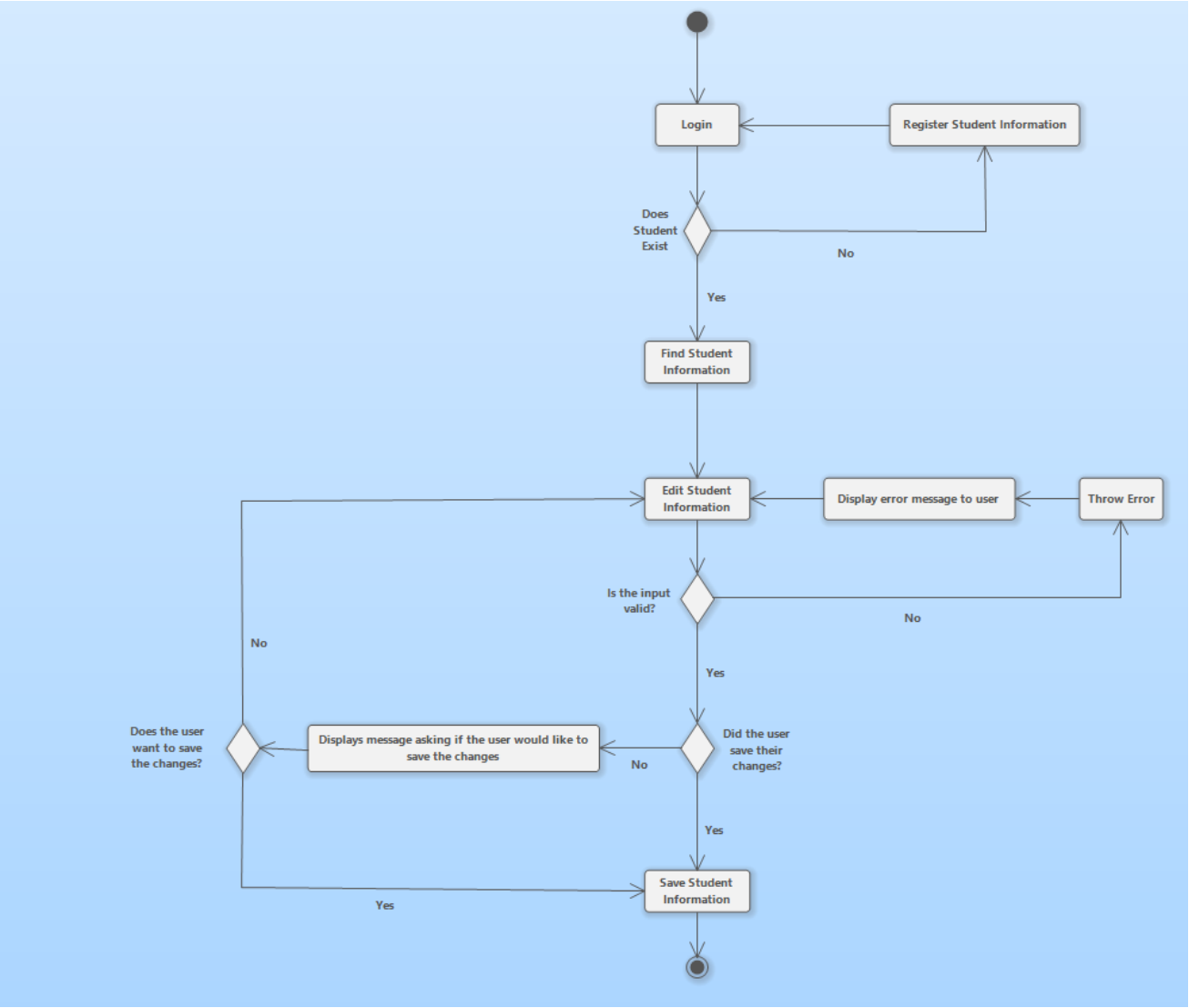


Add Fees

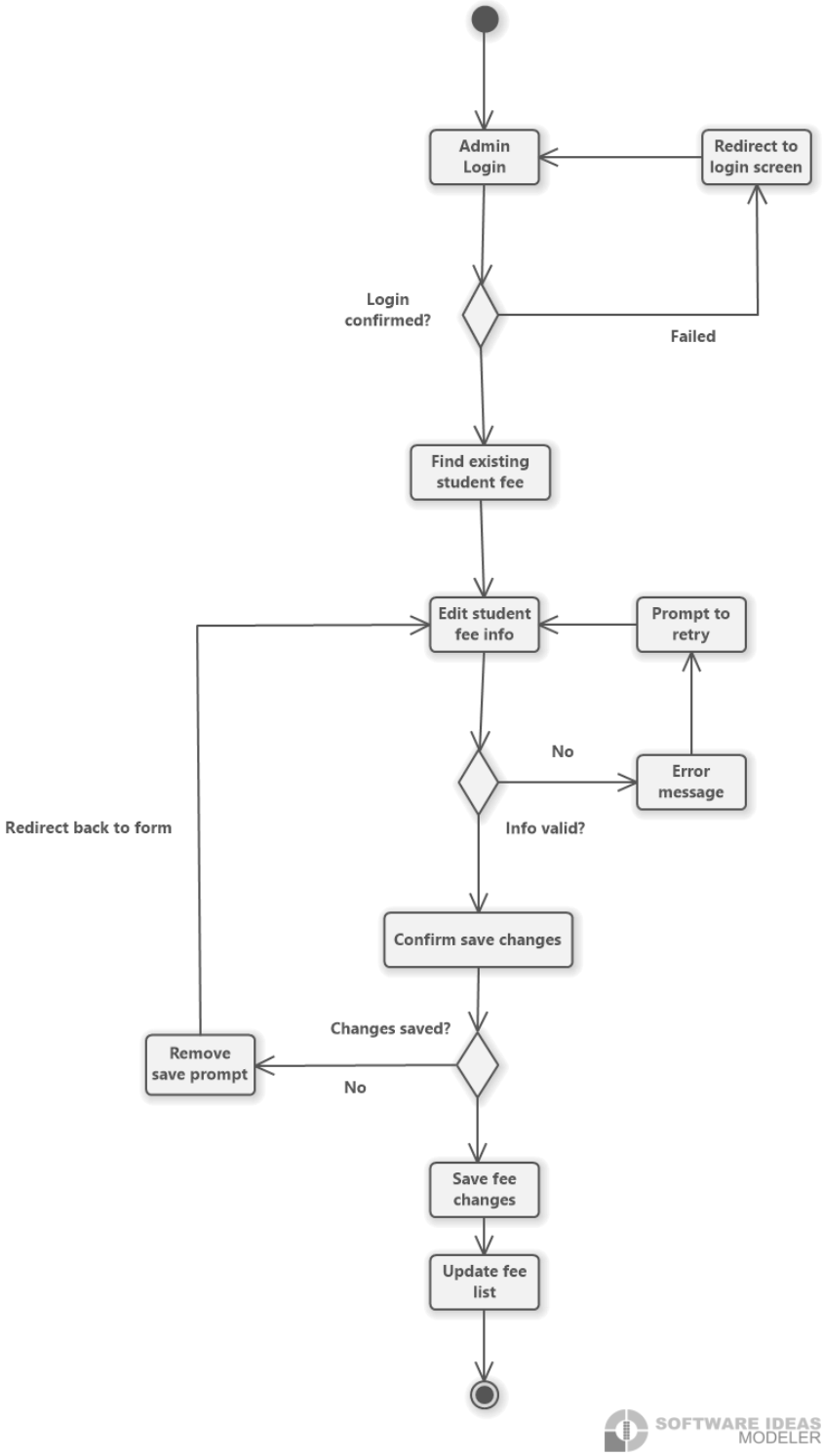

Admin
Manage fees screen :
Manage fees controller :
Fees :
addFees button action
enterStudentInfo(tuition, supplies)
submit
sendFormInfo
processFees
studentFeeInfo Updated
send confirmation message
SOFTWARE IDEAS MODELER

Activity Diagram:

Edit Student Information

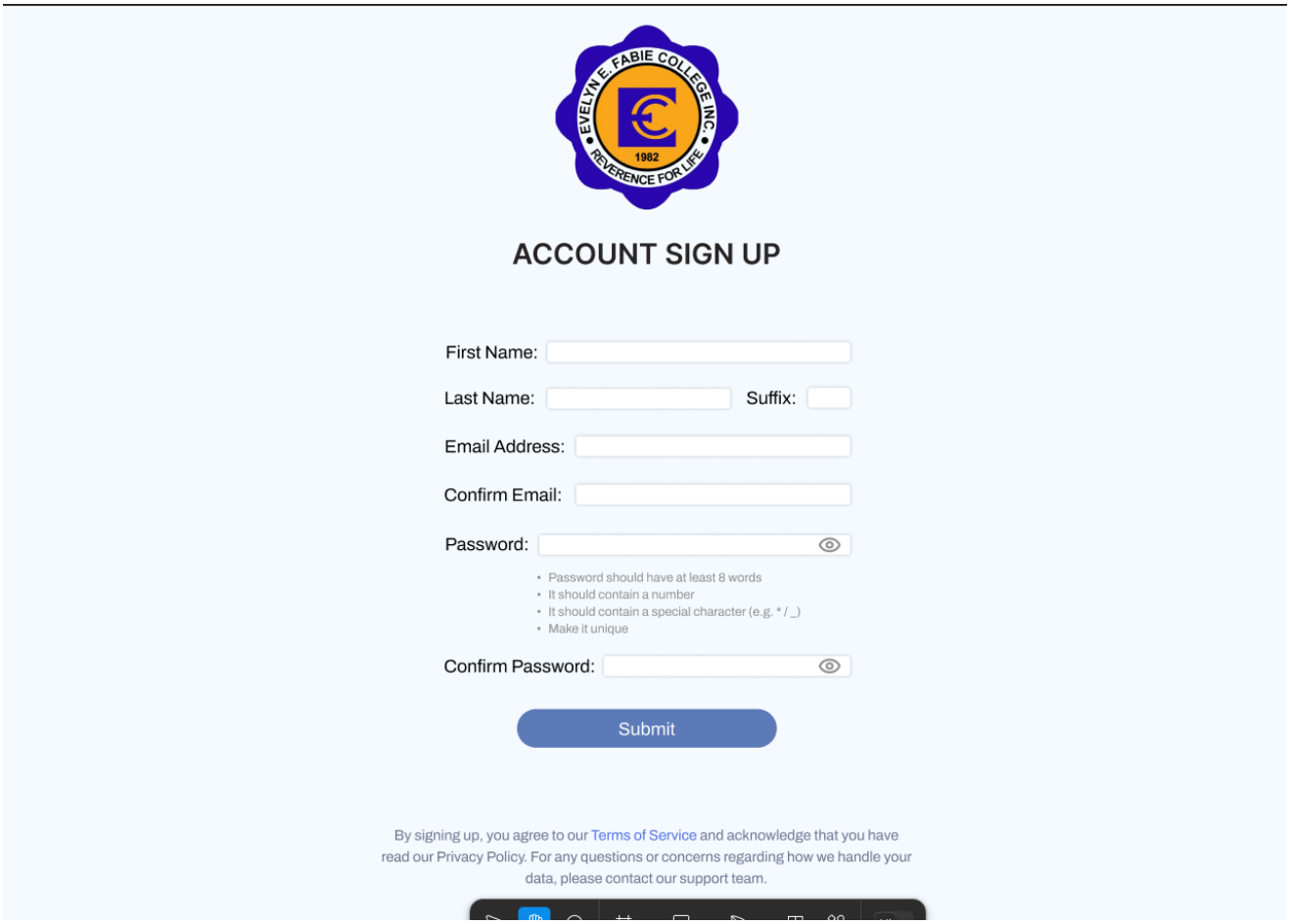


Modify Fees

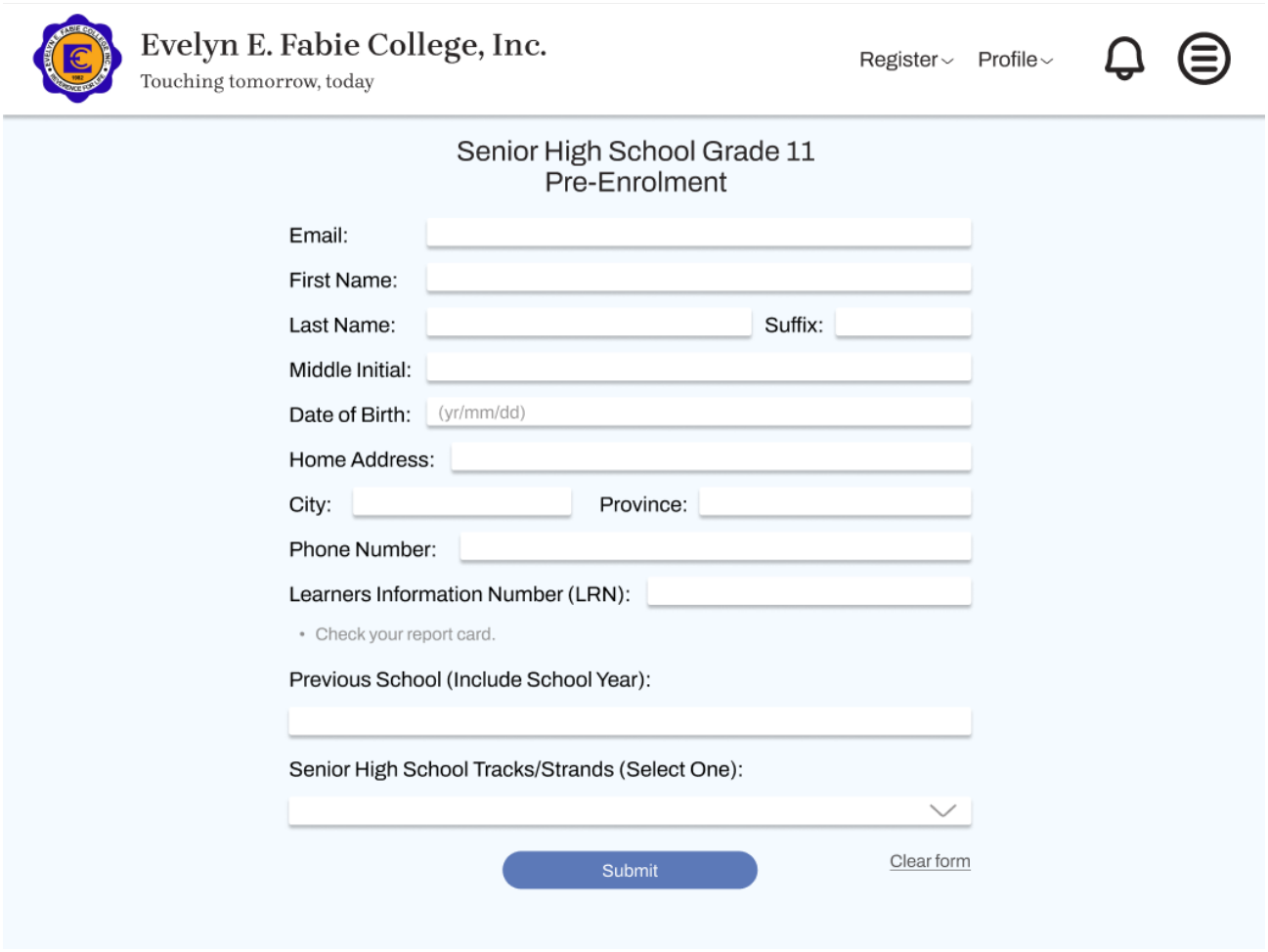

Admin Login
Redirect to login screen
Login confirmed?
Failed
Find existing student fee
Edit student fee info
Prompt to retry
No
Error message
Redirect back to form
Info valid?
Confirm save changes
Changes saved?
Remove save prompt
No
Save fee changes
Update fee list
SOFTWARE IDEAS MODELER

Preliminary User Interface Design:

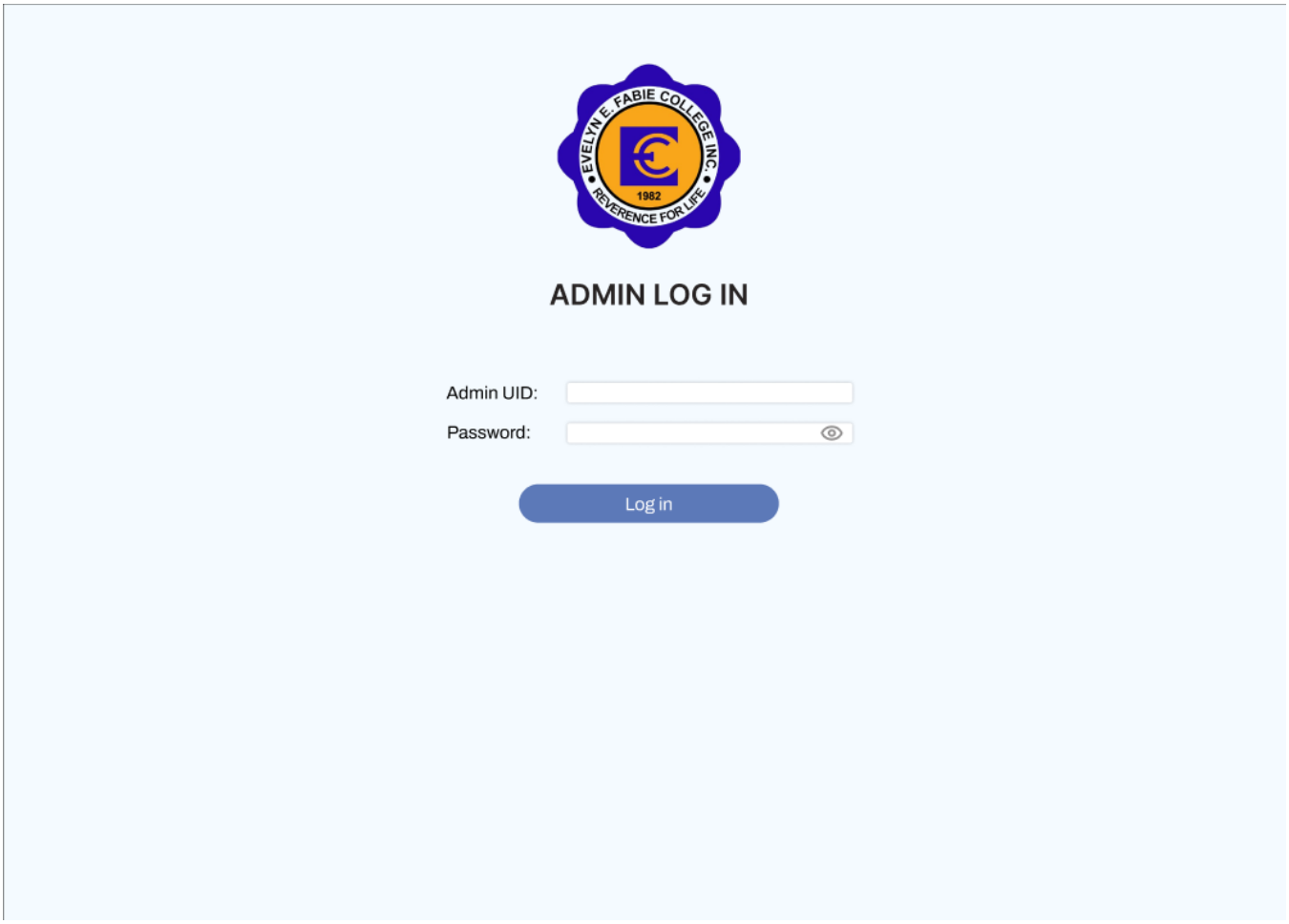
Student Sign-Up Page



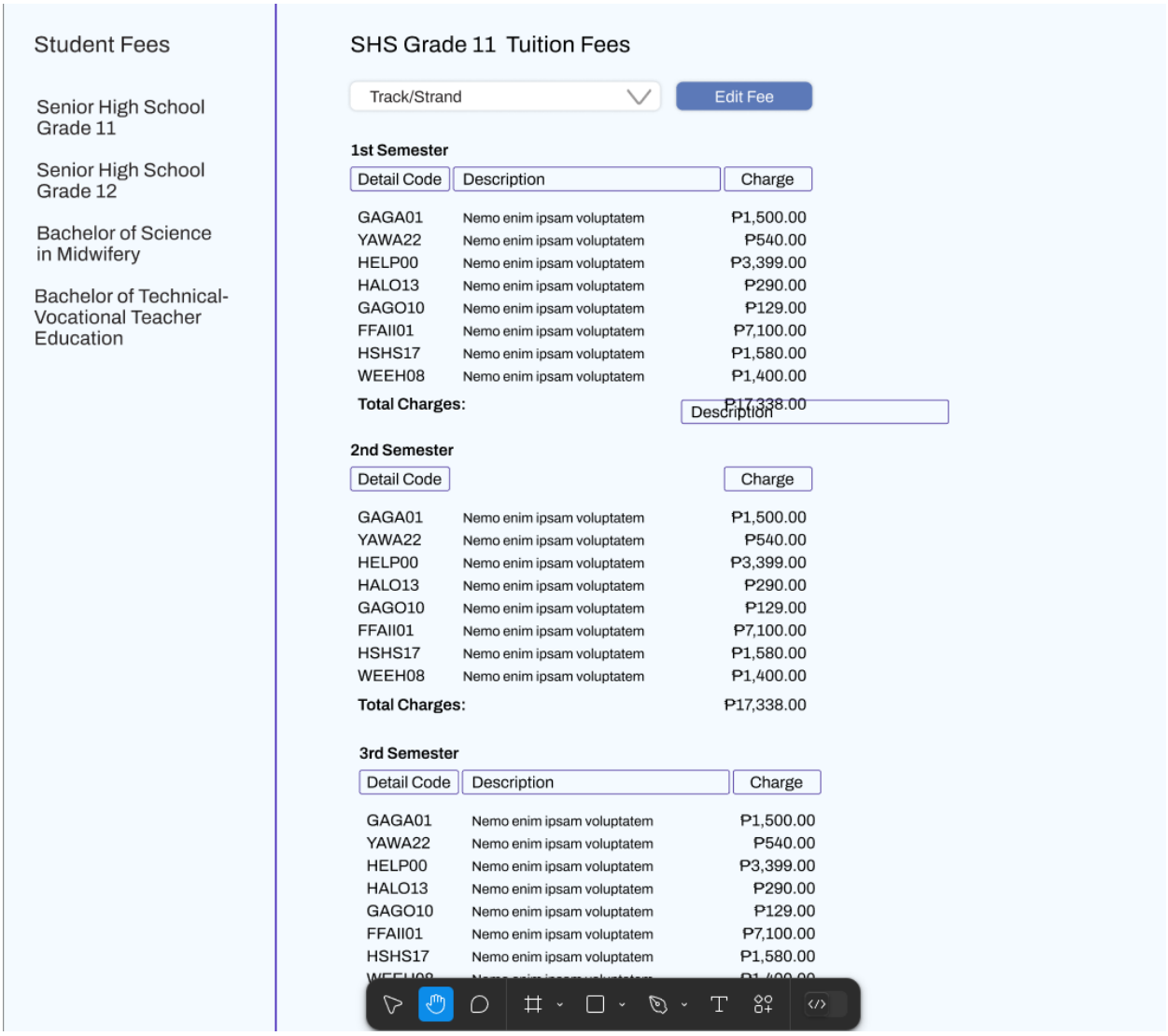
Pre-enrollment Page



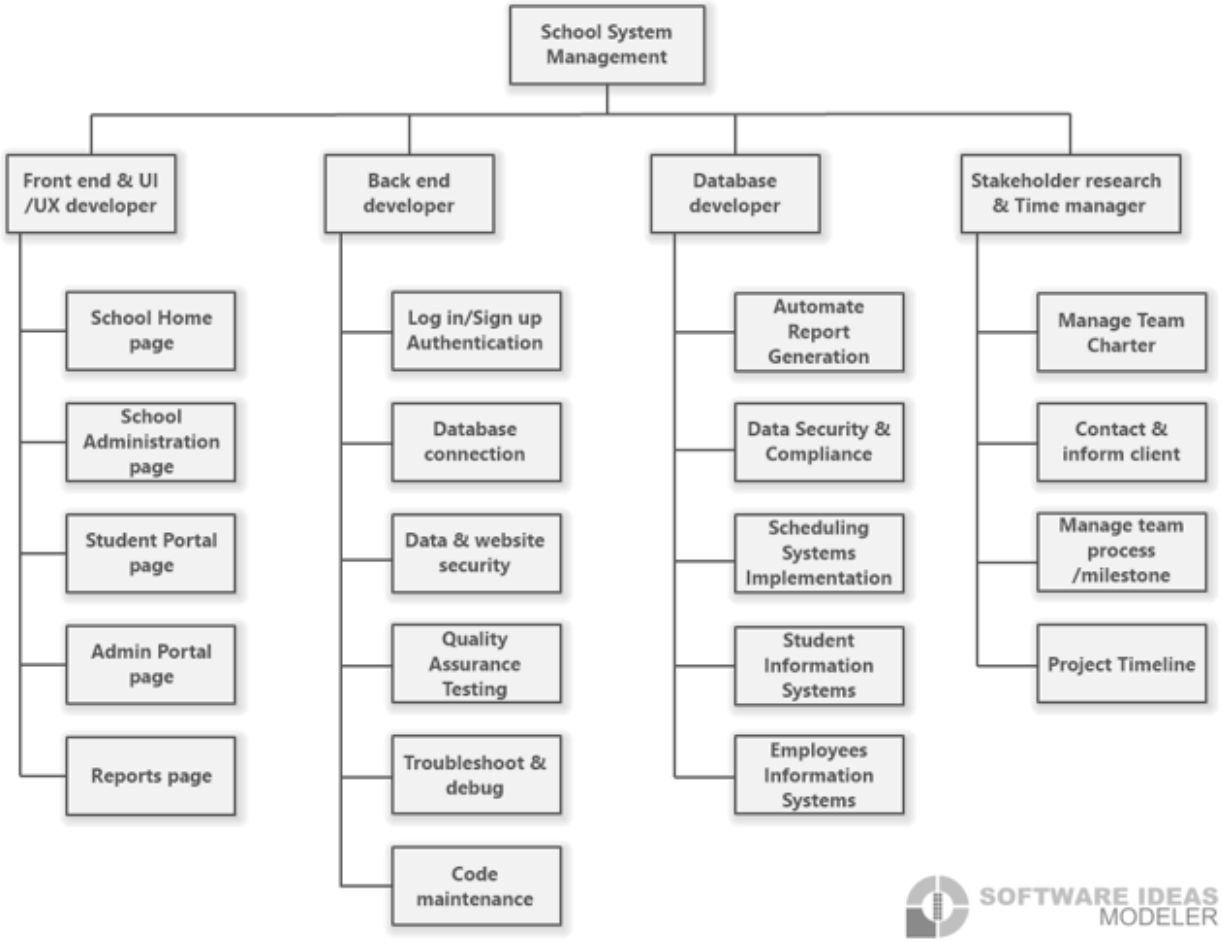
Admin Log In


EVELYN E. FABIE COLLEGE INC.
1982
REVERENCE FOR LIFE
ADMIN LOG IN
Admin UID:
Password:
Log in

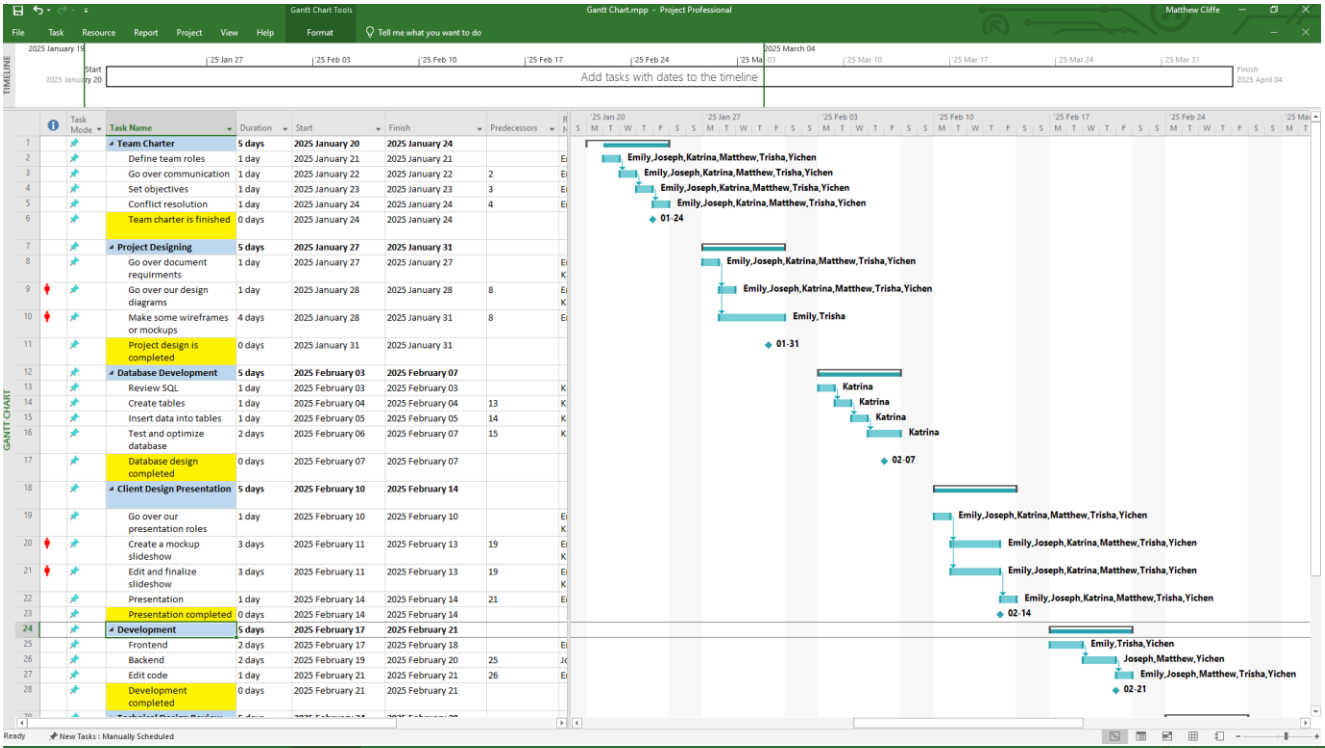
Tuition Fee Page

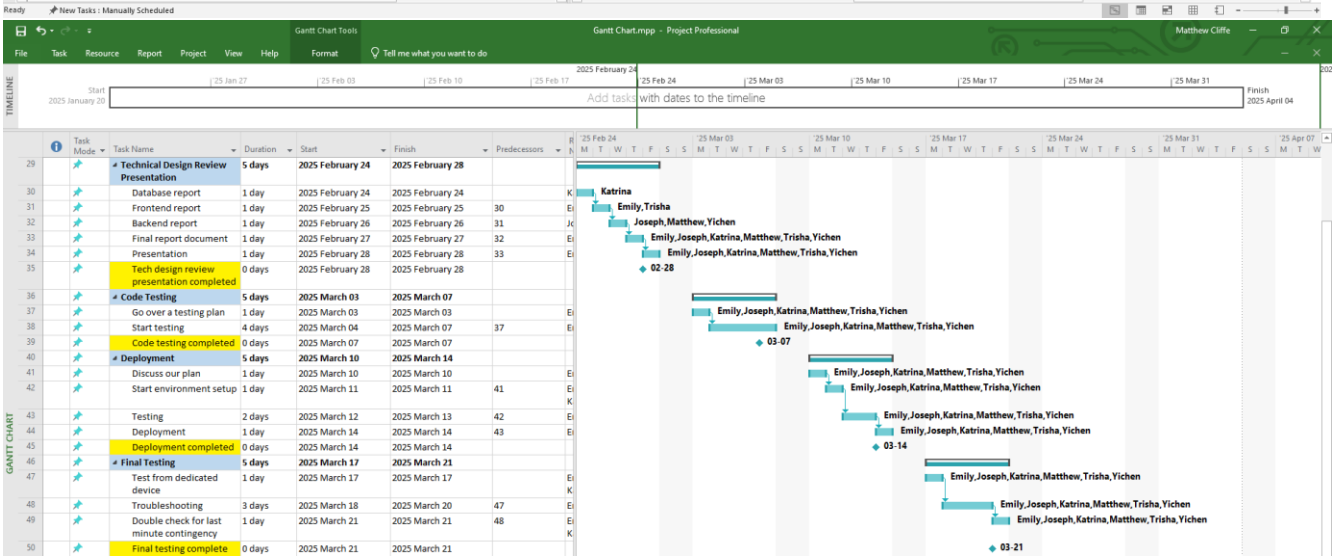


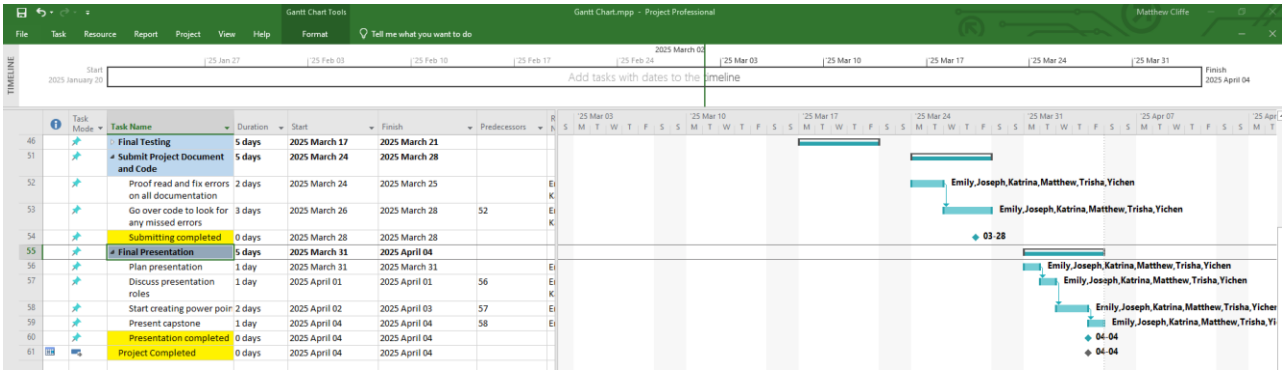
Work Breakdown Structure:



Gantt Chart:

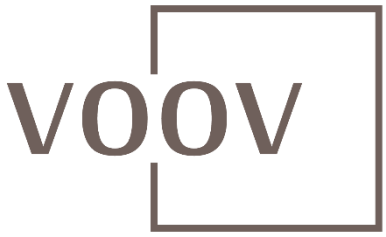

Gantt Chart.mpp - Project Professional
Add tasks with dates to the timeline
GANTT CHART
Ready


Gantt Chart.mpp - Project Professional
Add tasks with dates to the timeline
GANTT CHART


Gantt Chart.mpp - Project Professional
Add tasks with dates to the timeline

Team Contract

Team Name: Voov
Team Logo:



Purpose:
This agreement serves as a road map for our trip. The team contract serves the function of outlining each member's obligations and setting limits based on conversations and expectations.
Mission:
Our goal is to utilize our expertise and professional abilities to achieve a successful outcome.
Vision:
We aim to create a sustainable and impactful App/website/Application software that supports local businesses.
Team Goals

- To practice confidence and professionalism during the whole course of the project.
- Meet the course study deadlines with an outstanding grade not lower than 95%.
- Expand on our knowledge and skills through group problem solving

Skills we aim to gain through the process:

- Effective communication
- Cooperation and respect for each other
- Time management
- Being a team player and collaborator
- Flexibility in workspace

Team Member Contact Information

Team Member	Contact Info	Constraints
Mathew Cliffe	403.909.5846 matthew.cliffe@edu.sait.ca	N/A

Trisha Aquino	825.733.4339 / trishayvonne.aquino@edu.sait.ca	None – use either Discord or Teams
Joseph Bui	825.437.5527 / joseph.bui@edu.sait.ca	N/A
Katrina Keen Buenavista	403.827.7912 / katrinakeen.buenavista@edu.sait.ca	None – use either Discord, WhatsApp or Teams
Yichen Hao	587.429.5134 / yichen.hao1@edu.sait.ca	Cannot be reached Friday evenings
Emily Thieu	403-383-8739/ emily.thieu@edu.sait.ca	None – use either Discord or teams

Team Members with Professional Biography

Katrina Keen Buenavista

Educational Background	• Bachelor's in industrial technology major in Computer Technology
Employment History	• Coop/On-the-job-Training – IT helpdesk
Summary of Skills	• Programming Languages: • Front-end: React.js, next.js • Back-end: C++, C#, Python, HTML, CSS • Database: MySQL, non-SQL

Intellectual Property Statement

- Unauthorized use or disclosure of any intellectual property is strictly prohibited.
- **Voov** is committed to protecting its intellectual property rights and ensuring that they are used solely for the company's and its stakeholders' benefit.
- Team members should not share work and related intellectual property that belongs to the client without the client's prior written consent.

Team Member Role Assignment

Phase 1:

Name	Role	Description
Mathew Cliffe	Project Manager/Team Leader	• Project coordination and scheduling • Communication with team members and stakeholders • Managing deadlines and resources Organizes and manages team members
Emily Thieu, Trisha Aquino	Front-end developer and UI/UX designer	• Creating the structure and layout of the entire web application • Design and implement the user interface
Katrina Keen Buenavista	Database Designer	• Defining the detailed database design, including tables and other database-specific constructs

		• Connecting the database to the entire application
Yichen Hao, Joseph Bui	Back-End Developer	• Develop API for bridging Back-End & Front-End

Phase 2:

Name	Role	Description
Mathew Cliffe	Project Manager/Team Leader	• Project coordination and scheduling • Communication with team members and stakeholders • Managing deadlines and resources Organizes and manages team members
Emily Thieu, Trisha Aquino	Front-end developer and UI/UX designer	• Creating the structure and layout of the entire web application • Design and implement the user interface
Katrina Keen Buenavista, Trisha Aquino	Database Designer	• Defining the detailed database design, including tables and other database-specific constructs Connecting the database to the entire application
Yichen Hao	UI/UX designer, Back-End Developer	• Developed base use case and planning for UI
Matthew Cliffe, Joseph Bui	Back-End Developer	• Develop API for bridging Back-End & Front-End

Phase 3:

Name	Role	Description
Mathew Cliffe	Project Manager/Team Leader	• Project coordination and scheduling • Communication with team members and stakeholders • Managing deadlines and resources Organizes and manages team members
Emily Thieu, Trisha Aquino	Front-end developer and UI/UX designer	• Creating the structure and layout of the entire web application • Design and implement the user interface
Katrina Keen Buenavista, Trisha Aquino	Database Designer	• Defining the detailed database design, including tables and other database-specific constructs Connecting the database to the entire application
Yichen Hao	UI/UX designer, Back-End Developer	• Developed base use case and planning for UI
Matthew Cliffe, Joseph Bui	Back-End Developer	• Develop API for bridging Back-End & Front-End

Established Norms and Expectations
Conduct

It is required of every member to abide by the following rules. Penalties for breaking the written regulations may vary depending on the offense committed:

- If someone misses a meeting or class without warning, they will have 24 hours to explain. If the group determines that the absence is warranted, the offender will get a verbal warning.
- If a group member reaches a deadlock in decision-making and won't give up, a game of Uno (with stacking) will be used to decide the winner.
- It is required of members to attend meetings at the appointed time and day. If a member anticipates being late for the meeting, they should let everyone know. "On time" is defined as starting the meeting within ten to fifteen minutes of the stated start time.
- If a member will not be present at the planned meeting, they should let the other members know.
- Throughout the project, it is anticipated that each participant will treat the other with appropriate decorum and respect; if not, they will get a verbal warning.
- If any modifications should be made to the group members' tasks, they should always be informed. When an issue arises, don't be afraid to tell everyone right away so that a solution may be found.
- It is required of every member to finish allocated work on schedule. If necessary, extensions can be worked out.

Communication

- In-College / Video conference (primarily through MS Teams)
- Primary mode of communication: Teams
- Secondary mode of communication: Discord
- Normal working hours: 12:00 PM to 12:00 AM
- Required time to reply: Within 5-hour period

Meeting Times

- Regular Meeting: Teams, Saturday @ 1-2:50 PM
- Optional Meeting #2: Teams, Saturday after class for 1-2 hours to deliver updates on the work progress
- Optional Meeting #3: Discord, Mondays and/or Thursdays at 8 PM

Decision-Making Process

- Major decisions will be made through consensus among team members.
- In disagreements, the project manager will facilitate discussions to resolve the issues.

Conflict Handling

Here are the lists of conflict-handling strategies with potential resolutions during the project:

- Establish regular check-ins and make sure that lines of communication are open and transparent to avoid misunderstandings and to keep everyone updated.

- After acknowledging that everyone has different preferences, decide on a cooperative workflow that combines everyone's skills while guaranteeing flexibility and compromise.
- Define fair job distribution and clear expectations. Make sure that the leader implements progress checks to ensure that all team members are held accountable.
- To establish an agreement, promote candid conversations and brainstorming sessions where all ideas are considered. Keep the project's goals front and center.

Expelling Plans

If a group member does not adhere to the rules stated on **Conduct**, they will be addressed according to their offense:

First Offense

The person responsible is given a verbal warning. He or she is required to provide each group member with their preferred beverage.

Second Offense

A formal apology letter is written and sent to every member of the group, along with a 20% deduction from the final project grade as a second warning. The offense, an admission of guilt for their behavior, and an apology should all be included in the letter.

Third Offense

Expulsion from the group.

Agreement

We all agree to follow this team contract throughout the CPSY-301 project and are all responsible for enforcing the rules outlined above.

Mathew Cliffe

Trisha Aquino

Yichen Hao

Emily Thieu

Joseph Bui

Katrina Keen Buenavista