Design and Analysis of Software Systems
Assignment - 3
IMS Design

Abhiram Tilak - 2022113011
Moida Praneeth Jain - 2022101093

March 27, 2024

1 Software Used

We have used StarUML to design the class and state diagrams, and LaTeXfor the report.

The figure 1 is a high resolution image of the class diagram. In case the image is blurry, use the following drive link to get access to the files in pdf, jpeg and staruml (mdj) formats.

Drive Link: TODO

2 Class Diagram

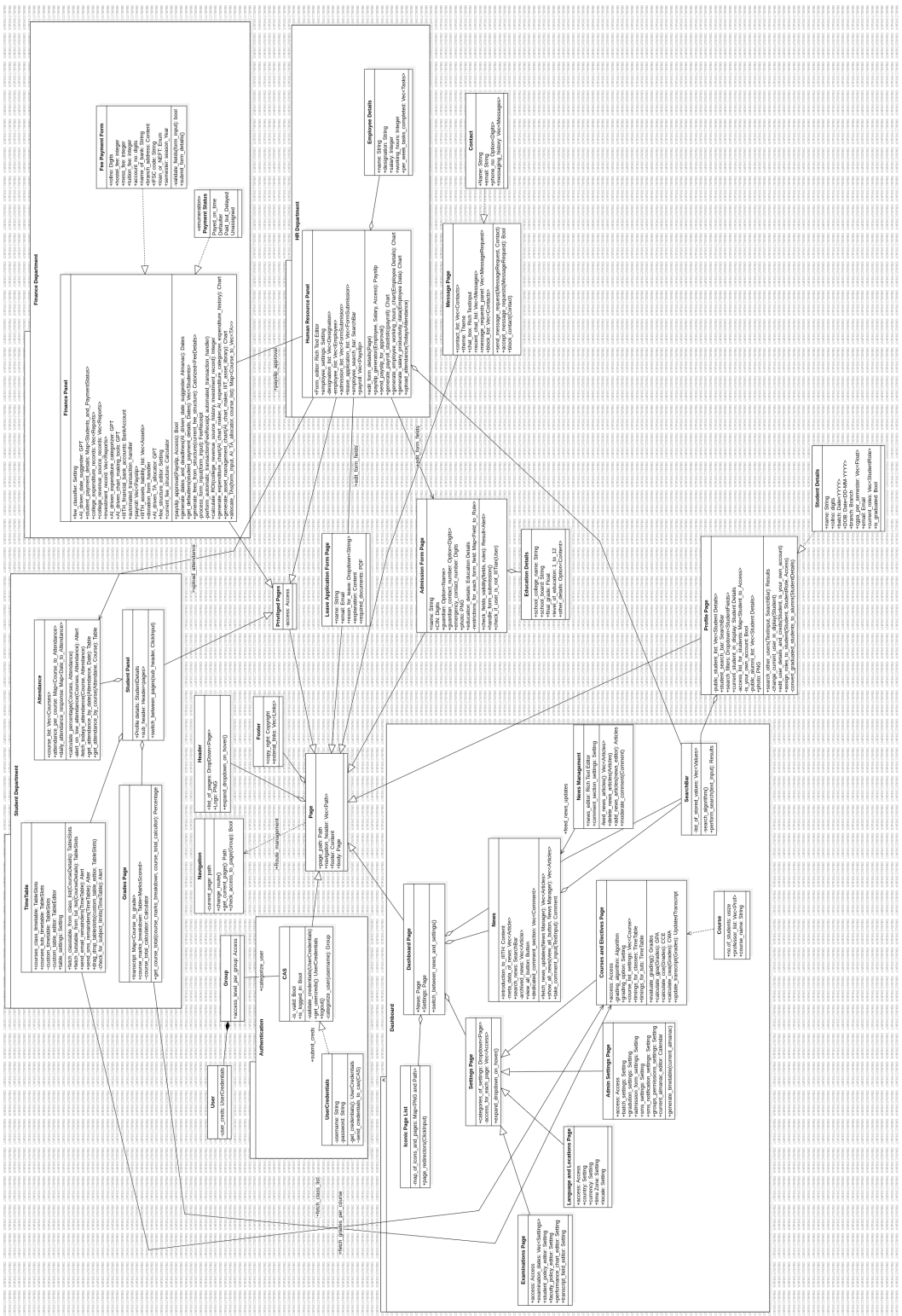


Figure 1: Class Diagram for IMS

3 Class Summaries

User	**Class State:** Is a basic unit of a user of IMS, containing a unique username(email) and password. • User Credentials

Group	**Class State:** Group is a basic categorization of Users to have special access and privileges • Access Level

UserCredentials	**Class State:** Contents of form input after the person enters details. • Is_Valid_User • Is_Logged_In
	• get_credentials(): Takes form input and converts it into credentials to be passed to CAS. • send_credentials_to_cas(): Sends credentials to external CAS in json format.

CAS	**Class State:** Central Authentication system, used to validate IIIT credentials. • Is_Valid_User • Is_Logged_In
	• Validate User Credentials(): Used to validate user credentials given as input • Get User Credentials(): Used to get the from from the submission page. • Categorize User(): After authentication tries to classify user and give sufficient Access.

Page	**Class State:** Contains basic skeleton of each page of the website or the portal • Page Path • Navigation Header • Footer • Body

Footer	**Class State:** Footer located at the bottom of the webpage • Copyright • External Links

Header	**Class State:** Header at the top of the page used for navigation • Dropdown of pages • IIIT Logo

Navigation	**Class State:** Handles backend navigation and routing between pages visited • Current Page
	Class Behavior • change_route(): Used to handle changing the path of the page and redirect to the next one • get_current_page(): Get the path of the current page, used for displaying and redirection • check_access_to_page(): Get the access levels of each page and show a trespassing alert otherwise

Dashboard	**Class State:** The main and the opening page of the portal which has most features common but some items can be not displayed or displayed based on Access. • News • Page
	Class Behavior • switch_between_news_and_settings(): By default only one of News or settings page is shown to view.

SearchBar	**Class State:** Template search bar. • List of stored values
	Class Behavior • search_algorithm(): This defines the algorithm used to prioritize search results, leniency in matching results, etc. • perform_search(): Handles search when text input is specified and displays results.

Page List	**Class State:** List of clickable icons used for user friendly navigation. • Map of icons and page
	Class Behavior • page_re-directors(): Used to handle redirection to respective pages when icons are clicked on

Settings Page	**Class State:** Sample Settings page containing different types of settings with different level of access • Categories of settings • Access for each Page
	Class Behavior • expand_dropdown_on_hover(): Each settings page contains various settings which will be divided into various categories.

Examination Page	**Class State:** Page used by Exam Cell for changing settings related examination, dates, policies etc. • Access • Examination Dates • Student Policy Editor • Faculty Policy Editor • Performance Chart Editor • Transcript field editor

Language and Locations	**Class State:** Page with settings related to language and location settings, accessible to all users. • Access • Country • Currency • Time Zones • Locale

Admin Settings Page	**Class State:** Contains Administrator Settings related to the whole IMS portal like SMS settings. Also contains settings related to graduation and admission. • Access • Batch Settings • Graduation Settings • Admission Form Settings • SMS Settings • SMS Notification Settings • Group Permissions Settings • Almanac / Calendar Settings
	Class Behavior • Generate Time Table(): Handles generate time tables from the current Settings and configuration

Course and Electives Page	**Class State:** Settings page usually used by faculty to change things like grading, and also set classes and tutorials. • Access • Grading Algorithm • Grading Option • Course List Setting • Timings for Classes • Timings for Tutorials
	Class Behavior • evaluate_grade(): Use the grading algorithm to find the grades. • calculate_gpa(): Used to calculate GPA from grades • calculate_cce(): Used to calculate CCE from grades • calculate_cwa(): Used to calculate CWA from grades • update_transcript(): Updates the transcript from the grades calculated from each course.

News	**Class State:** This is the default body of the dashboard page. contains news articles which has a separate management class to change the content. • Introduction to IIITH • Meta Data of News • Search News • Archived News • ”View all” button • Dedicated Comment Section

Continued on next page

Table 0: (Continued)

	Class Behavior • fetch_news_updates(): Used to fetch news updates from the editable news update management panel • show_all_news(): When the "View All" buttons is pressed, it expands to show all news up for display • take_comment_input(): Handle posting comments in the comment section if there is access.

News Management	**Class State:** Management component used for generating content and moderating the comments of the News articles. This can be accessed by News Moderators. • News Editor • Comment Section Settings
	Class Behavior • feed_news_articles(): Used to feed the dashboard page with latest news whenever fetched for updates. • delete_news_articles(): Used to delete older news articles • add_news_articles(): Used to add new articles from the Rich text editor • moderate_comments(): Use the delete, flag, report comments by administrator.

Profile Page	**Class State:** Page visible to every user dedicated to showing user details and also supports finding other users, students, alumni and find their level of access to IMS. • Public Student List • Student Search Bar • Search Filters • Current Student in Display • Access List for students • is_your_account • Public Alumni List

Continued on next page

Table 0: (Continued)

	Class Behavior • search_other_students(): Use the search bar to search other students and alumni and display the results • change_current_user_in_display(): Used to handle the current user details that are shown in the placeholder. • edit_user_details_and_creds(): Gives the ability to edit some of the profile details depending on if the signed-in user is trying to edit his own account. • assign_roles_to_students(): Users with enough privileges can assign roles to users. This roles have no purpose in IMS and are different groups. • convert_graduated_students_to_alumni(): Automated function that converts graduated students to alumni

Student Details	**Class State:** This is a structure containing all student details at the present State. • Name • Roll No • Batch • Date of Birth • Branch • CGPA • email • current_roles • is_graduated

Admission Form Page	**Class State:** Admission form which is available if the user is not recognized as a IIITian • Name • CIN • Guardian Name • Guardian Contact Number • Emergency Contact Number • Photo • Education Details • Restrictions for each form field

Continued on next page

Table 0: (Continued)

	Class Behavior: • check_fields_validity(): Checks if all fields are entered properly with constraints • handle_form_submissions(): Record all the form details and convert into JSON format • check_if_user_is_not_IIITian(): This check is to control page visibility

Human Resource Panel	**Class State:** Privileged page for Human Resource Employees used to perform various actions. • Admission form editor • Employee Settings • Designation List • Employee List • Admission List • Leave Application List • Employee Search Bar • Payroll
	Class Behavior: • edit_form_details(): Used to edit forms like Leave application, Admission form. • payslip_generator(): Have the ability to examine employee details and generate a payslip. • send_payslip_for_approval(): Send payslip for approval to the finance department • generate_payroll_statistics(): Use payroll data to generate a chart • generate_employee_working_hour_chart(): Take in data about employees and their working hours, and generate performance charts. • generate_salary_productivity_data(): Use the tasks completed by employees and performance data to get the per capita productivity chart. • Upload attendance: Upload the recorded attendance for the day and upload it. Shows up in the attendance page.

Message Page	**Class State:** A page dedicated to have messaging services among users of IMS • Contact List • Theme • Chat Box • Recent Chat List • Message Requests Panel • Block List
	Class Behavior: • send_message_request(): Used to send a curated request to the other IMS user contact, asking permissions to send messages • accept_message_requests: Used to accept message requests from a contact • block_contact(): Used to block a contact and send it to Block list, messages from this user will not be displayed.

Contact	**Class State:** This is an abstraction of a user of IMS, contains information that other user can see • Name • Email • Phone No • Messaging History

Fee Payment Form	**Class State:** This is the form that students fill to perform online fee payment through IMS. • Roll No • Hostel Fees • Mess Fees • Tuition Fees • Account No • Name of the Bank • Branch Address • IFSC Code • [Loan or NEFT] (Enum) • Semester
	Class Behavior: • validate_fields(): Used to ensure if all the fields are filled accordingly, and the form is valid • submit_form_details(): Used to submit form details, to be process by finance department

Finance Panel	**Class State:** Privileged page for finance departments, contains tools (AI driven), Settings, Databases, Form editors, Access to edit critical components like Fee Structure • Fee Classifier • AI Driven Date/Deadline predictor • Student Payment Details • college expenditure records • college revenue source records • investment records • AI Driven Expenditure Classifier • AI Driven chart making tools • IIITH Financial bank accounts • automated transaction handler • payroll • IIITH asset liability list • donation form handler • fee structure editor • current fee strucutre
	Class Behavior: • payslip_approval(): Take payslip from HR team and approve it after manual inspection. • generate_dates_and_deadlines(): Use the AI tools and manual intervention to generate deadlines for fee submission and other financial processes. • get_defaulters(): Use the deadline/dates info and the student fee details to find out fee payment defaulters • generate_fees_from_structure(): Use the current implementation of fee structure as input to calculate fees required to pay for each student • process_form_input(): Take the form input and make readable fields accepted by automatic transactor • perform_automatic_transaction(): Take fee information and perform automatic online bank transfers • calculate_roi(): Use the revenue, and investment and donations to calculate ROI of college • generate_expenditure_chart(): Use the expenditure records to generate a classification chart with detailed breakdown • generate_asset_management_chart(): Use the asset records to generate charts on current status on assets and their value.

Student Panel	**Class State:** This panel is accessible to students of IIIT and contain various subpages dedicated to them • Profile Details • Sub Header
	Class Behavior: • switch_between_pages(): Used to switch between subpages in the student panel

Grades Page	**Class State:** A page for information of grades and marks of each course, also contains transcript • Transcript • Course Marks • Course Total Calculator
	Class Behavior: • get_course_total(): Used to get course total of the subject depending on the course and marks breakdown and the grading scheme defined in the course settings.

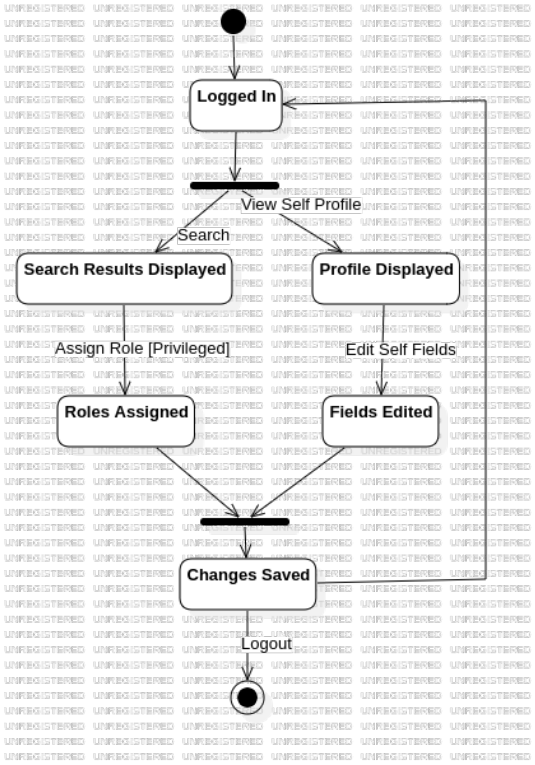
Attendance	**Class State:** Page dedicated to display attendance information of the student • course list • daily attendance response • attendance per course
	Class Behavior: • calculate_percentage(): Used to calculate percentage attendance of a particular course • alert_on_low_attendance(): automated system that uses sms alerts on low attendance • fetch_todays_attendance(): automated system that fetches attendance updates every once in a while • get_attendance_by_date(): used to make detailed table with attendance breakdown by date • get_attendance_by_course(): used to make table with attendance per course.

Time Table	**Class State:** A dedicated time table for students, with support for custom entries • course class timetable • course tutorial timetable • custom timetable • custom table editor • table settings
	Class Behavior: • fetch_classtable_from_class_list(): automated fetching function for information and times of classes from course page • fetch_tutstable_from_tuts_list(): automated fetching function for information and times of tutorials from course page • send_email_remainders(): Automated system that sends email remainders, as configured • send_sms_remainders(): Automated system that sends sms remainders, as configured • drag_drop_tableslots(): Customizable editor to drag and drop events, supports Name and Time of the event • check_for_subject_limits(): Used to check for clashes in the timetable and alerts

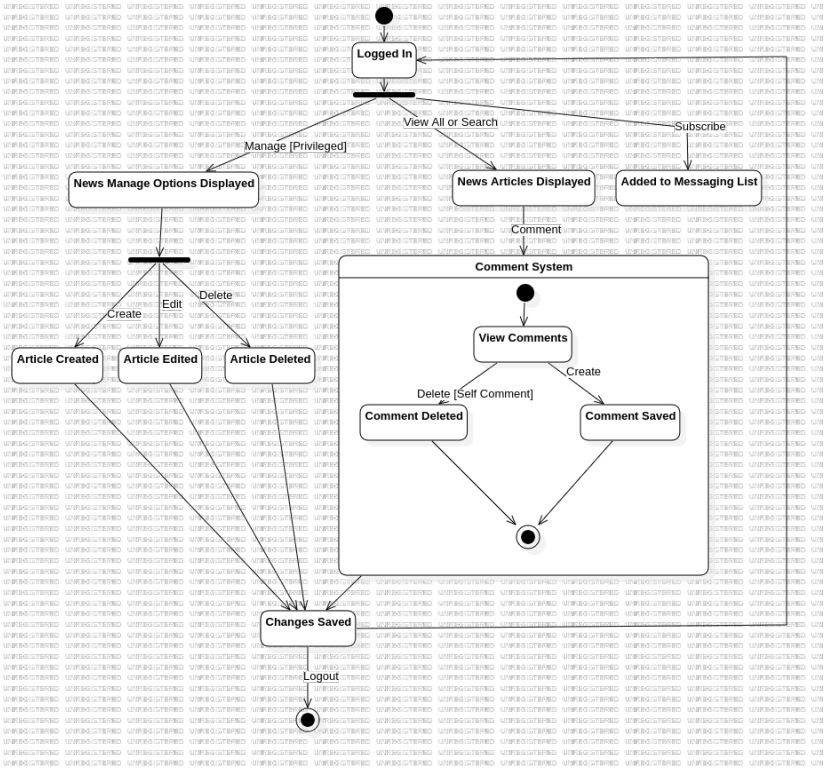
4 State Diagram

State diagrams for the most frequent and important use cases have been presented below.

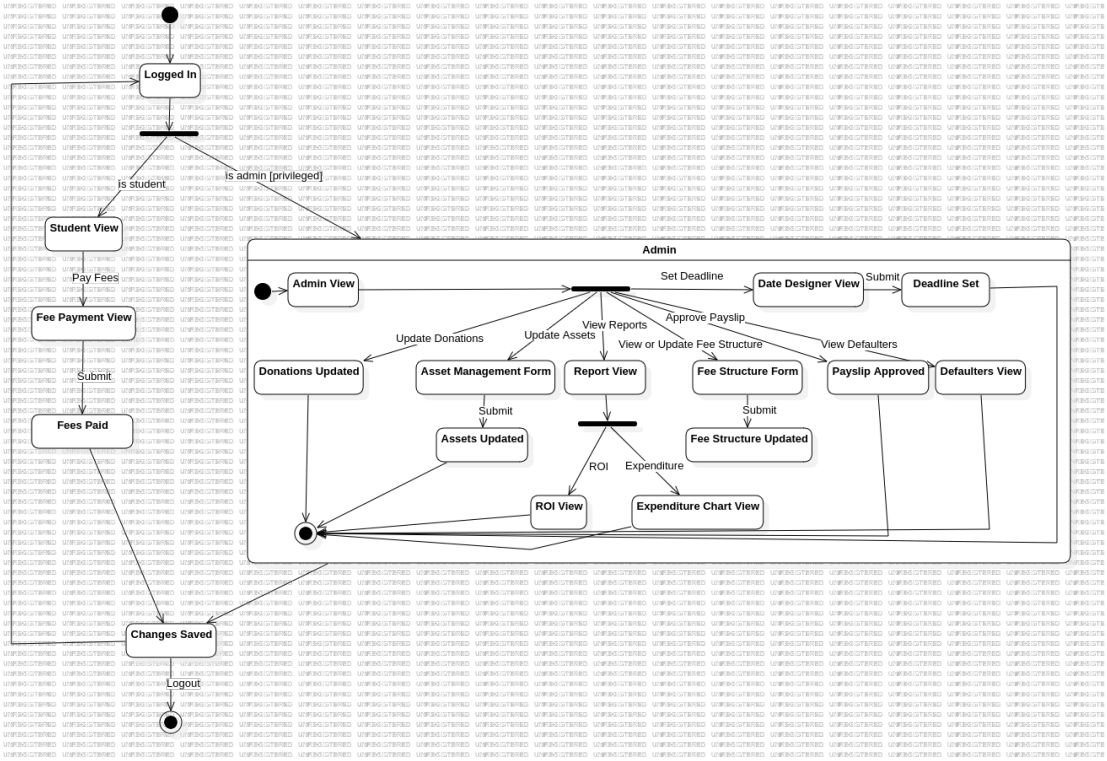
4.1 User Profile



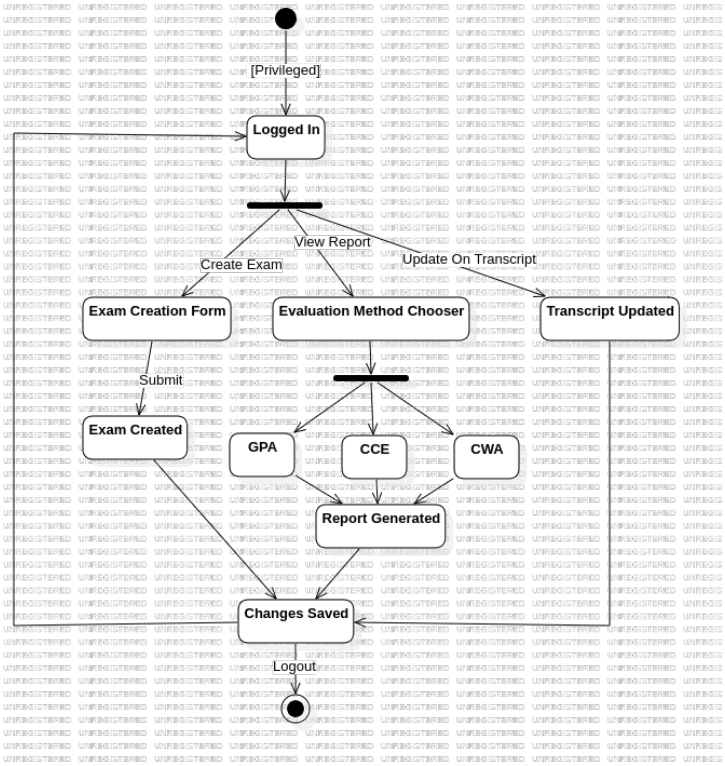
4.2 News



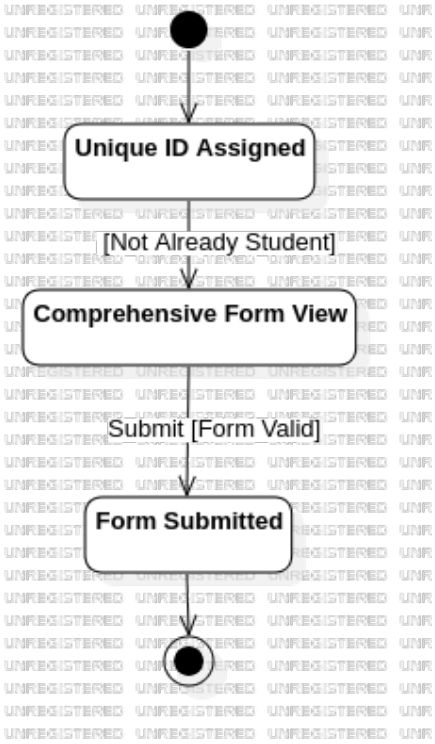
4.3 Finance



4.4 Examination



4.5 Admission



5 Design Outline and Features

5.1 Low Coupling

The design follows the principle of low coupling by separating concerns into distinct modules, such as User Management, Student Details, and Finance Management. Each module has a well-defined set of responsibilities and minimizes dependencies on other modules, allowing for easier maintenance and modification.

5.2 High Cohesion

The design promotes high cohesion by grouping related functionalities and data within the same module. For example, the Course and Batch Management module handles all aspects related to course definition, batch creation, student and faculty assignment, and class scheduling, ensuring that related operations are cohesively organized.

5.3 Separation of Concerns

The design separates different concerns into distinct modules, following the principle of separation of concerns. For instance, the User Management module handles user-related operations, while the Human Resource Management module focuses on employee-specific tasks, such as hiring, attendance tracking, and payroll processing.

5.4 Modularity

The design is structured as a modular system, with each module representing a distinct functional area. This modularity allows for better organization, maintainability, and scalability, as individual modules can be developed, tested, and updated independently.

5.5 Information Hiding

The design adheres to the principle of information hiding by encapsulating module-specific data and functionality within the respective modules. For example, the details of fee structure and invoicing are hidden within the Finance Management module, while the User Management module handles user-related information and operations.

5.6 Extensibility

The modular design promotes extensibility by allowing new functionalities or modules to be added without significantly impacting existing components. For instance, if a new requirement arises for managing hostels, a new module can be added to the system without disrupting other modules.

5.7 Reusability

The design encourages reusability by creating generic and reusable components or modules that can be shared across different parts of the system. For example, the Messaging System module can be reused for communication between different stakeholders, such as students, faculty, and administrators, reducing duplication of effort and promoting code reuse.

6 Reflection on our Design

These principles are integrated throughout the design to ensure a well-structured, maintainable, and scalable Institute Management System. The modular approach, clear separation of concerns, and adherence to principles like low coupling and high cohesion contribute to the overall quality and flexibility of the design.

6.1 Strongest Aspects

6.1.1 Modular Architecture

The design follows a modular approach, separating concerns into distinct modules such as User Management, Student Details, Finance Management, and Course and Batch Management. This modular architecture promotes maintainability, scalability, and extensibility, as individual modules can be developed, tested, and updated independently without affecting the entire system.

6.1.2 Adherence to Design Principles

The design adheres to well-established design principles like low coupling, high cohesion, separation of concerns, and information hiding. This adherence ensures that the system is organized, maintainable, and flexible, reducing the complexity and dependencies between different components.

6.2 Weakest Aspects

6.2.1 Potential Performance Bottlenecks

While the modular design promotes maintainability and extensibility, it may introduce performance bottlenecks or overhead due to inter-module communication and data transfer. Careful optimization and performance tuning may be required to ensure efficient system operation, especially in scenarios with high concurrency or large data sizes.

6.2.2 Complexity in Integration and Deployment

With multiple modules and potentially different technologies or frameworks involved, integrating and deploying the entire system can become complex. Although unit testing would be smooth due to the modular nature of the design, integration testing would be a tough due to the large number of modules themselves.