Project Title: Tuition Management System

Supervised By

MS. Raiyan Janik Monir
Lecturer
Department of Software Engineering
Daffodil International University

Submitted By

MD. Shahoriar Hasan
ID: 192-35-2839

Department of Software Engineering
Daffodil International University

This project report has been submitted in fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering.

APPROVAL

This thesis titled on "**Tuition Management System**", submitted by **MD. Shahoriar Hasan (ID: 192-35-2839)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Chairman

Dr. Imran Mahmud
Professor & Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 1

Md Shohel Arman
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Md. Rajib Mia
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Md Habibur Rahman
Associate Professor
Department of Computer Science and Engineering
Islamic University, Bangladesh

Declaration

I hereby declare that I have done this project under the supervision of MS. Raiyan Janik Monir, Lecturer, Department of Software Engineering, Daffodil International University. I also declare that this project is my original work for the degree of B.Sc. in Software Engineering and neither the whole work nor any part of this project has been submitted for another degree in this or any other university.

Md. Shahoriar Hasan
ID: 192-35-2839
Department of Software Engineering,
Daffodil International University

Certified by:

MS. Raiyan Janik Monir
Lecturer
Department of Software Engineering,
Daffodil International University

Acknowledgment

In today's competitive environment, success requires bridging the gap between theoretical knowledge and practical application. It is my sincere aspiration to participate in this important project. First and foremost, I express my deepest gratitude to Almighty Allah for guiding me throughout this journey. Without His grace, this accomplishment would not have been possible. I remain profoundly thankful to my parents for their unwavering love, confidence, and support, which have shaped my path. I am honored to study at Daffodil International University, and I extend my sincere appreciation to Prof. Dr. Imran Mahmud, Head of the Department of Software Engineering. I also wish to thank all the respected faculty members whose engaging and clear instruction has enriched my academic journey. Special thanks go to my supervisor, MS. Raiyan Janik Monir, for her continuous guidance, valuable information, and encouragement in completing this project. Lastly, I wish to express my heartfelt appreciation to my batch-mates at DIU for their cooperation and compassion, which have greatly supported me in completing this task.Their camaraderie and teamwork have made this experience not only productive but also enjoyable. I look forward to carrying the lessons learned and the memories shared with all of them into the future.

Abstract

The Tuition Management System is a web-based platform designed to connect students with tutors through a streamlined, user-friendly interface. This system aims to modernize and simplify the private tutoring process by enabling students to search for tutors based on subjects, experience, and ratings, as well as book sessions and make payments securely online. The platform integrates essential features such as session scheduling, video conferencing, tutor reviews, and history tracking. Additionally, it includes administrative tools for managing users, sessions, and payments. By addressing common challenges like limited accessibility, poor session tracking, and inefficient payment handling, the Tuition Management System enhances the overall tutoring experience for students and tutors alike. This system is scalable, secure, and adaptable to various educational levels, making it a valuable tool in today's increasingly digital learning environment.

Table of Content

Chapter 1: Introduction

1.1 Project Planning and Initiation

The **Tuition Management System** is a web-based application developed to efficiently manage the operations of a tuition center or independent tutoring service. It serves as a complete academic and administrative management solution that supports:

- Student enrollment
- Session scheduling
- Billing and payment tracking
- Academic progress reporting
- Third-party tool integration (Ziteboard for virtual teaching, Google for document handling)

The system is structured with a responsive dashboard, role-based navigation, and full CRUD operations on all major entities like students, reports, and sessions. It simplifies the tutor's workload while ensuring accountability and transparency for parents and students.

1.1.1 Feasibility Study (Step-by-Step)

Phase 1: Preliminary Analysis & Project Scope Definition

Project Goals:

- Automate the entire tuition management process including students, classes, payments, and progress monitoring.
- Support daily and weekly schedule views for tutors to manage time efficiently.
- Record both progress reports (monthly) and session-wise reports (per class).
- Provide options to add new bills and track payment history.
- Integrate Ziteboard (for online whiteboard sessions) and Google services (like email, calendar, or storage).
- Generate real-time dashboards showing key stats like income, students, unpaid bills, and today's sessions.

Scope Overview:

- **Front-End**:
    - Dashboard with modules: Home, Students, Reports, Schedule, Billing, Ziteboard, Google
    - User forms for adding students, reports, and schedules

- **Back-End**:
    - MySQL database with tables like: std_info, schedule, txn, bill, report
    - Auto-incremented IDs and foreign key references
- **Output**:
    - Real-time schedule view (daily/weekly)
    - Reports by date and month
    - Billing interface to create and manage financial entries

Phase 2: Market Feasibility Analysis (Market Research)

A qualitative market study revealed the following insights:

- Private tutors and small institutions lack integrated platforms.
- Manual record-keeping (e.g., Excel or paper) leads to errors and inefficiency.
- There is strong interest in platforms offering:
    - Student profiles with timezone and contact info
    - Billing automation
    - Attendance & performance tracking
    - Integration with virtual whiteboards (like Ziteboard)

Target Audience:

- Freelance tutors managing international students
- Coaching centers with small to mid-size operations
- Online educators using platforms like Skype, Zoom, Ziteboard

Phase 3: Technical Feasibility Analysis

Technologies Used:

- **Front-End:** HTML, CSS, JavaScript
- **Back-End:** PHP
- **Database:** MySQL (accessed via phpMyAdmin)

Major Tables:

- std_info – stores all student-related data
- schedule – logs class times, dates, and status
- txn and bill – for financial transactions and billing
- report – detailed academic reporting

Functional Features:

Module	Key Features
Dashboard	Real-time stats (students, income, unpaid bills, sessions)
Students	Add/view students with timezone, course, fee
Schedule	View and assign daily or weekly class schedules
Reports	- Progress Reports: Monthly academic summary - Session Reports: Per-session insights with topics, marks, homework
Billing	- Add New Bill: Select student, enter amount, due date - Billing History: Track past invoices and payments
Ziteboard	External integration for live class whiteboarding
Google	Placeholder for integration with services like Calendar, Drive

Phase 4: Financial Feasibility Analysis

Given the self-hosted nature of the platform and reliance on open-source technologies, financial feasibility is strong.

Cost Estimation:

Item	Description	Estimated Cost
Development Time	3–4 weeks (solo/full-stack)	In-house
Hosting	Shared server / Localhost	$5–10/month
Software Tools	phpMyAdmin, MySQL, PHP stack	Free
Maintenance	Low (minimal updates required)	$0 if in-house
Ziteboard Plan	Optional (freemium available)	$0–$9/month

1.2 Target User Profile and Tentative Elicitation Process

Target User Profile

The system is primarily built for the following user groups:

Private Tutors

- Typically manage multiple students individually
- Need support for managing time zones, payments, and academic progress
- Conduct online sessions via platforms like Ziteboard, Skype, or Zoom

Small Coaching Institutes

- Handle several students and classes
- Require features like weekly scheduling, billing history, and academic reports
- Often include an admin who manages student records and payments

Parents and Guardians

- Indirect users who receive progress reports, session summaries, and billing information
- Require transparency and regular academic feedback

Students

- Participate in scheduled classes
- Receive feedback through session or progress reports
- Can be tracked for performance over time

Tentative Elicitation Process

To define the system requirements and understand user expectations, the following requirement elicitation steps were conducted:

Interviews

- One-on-one discussions with 2 private tutors and 1 small coaching center operator
- Identified key pain points in managing schedules, student data, and payments

Brainstorming Sessions

- Generated the core feature list:
    - Daily/Weekly class scheduling
    - Monthly progress reports
    - Session-wise class reports
    - Billing (Add new + History)
    - Integration with Ziteboard and Google tools

Prototype Testing

- Shared UI mockups and early dashboard layouts
- Collected tutor feedback on:
    - Schedule form usability
    - Report format clarity
    - Billing structure and flow

These steps helped ensure that the system was truly aligned with real-world needs and offered practical functionality.

1.3 Project Scheduling

Effective project scheduling ensures timely delivery and helps identify potential bottlenecks in the development process. The Tuition Management System followed a structured timeline, divided into well-defined phases with clear objectives and deliverables.

a. Time Frame

The estimated duration of the project was 5 weeks, with key phases and tasks distributed as follows:

Phase	Duration	Description
Requirement Gathering	Week 1-3	Interviews, observation, and documentation of user needs
System Design	Week 4-5	ER diagrams, data flow diagrams, user interface sketching
Development Phase I	Week 6-8	Backend setup, database schema (phpMyAdmin), and CRUD operations
Development Phase II	Week 7-9	Dashboard implementation, reports, billing, and integration features
Final Review & Deployment	Week 10	Final code review, deployment on local/hosted server, documentation
Testing & Debugging	End of Week 10	System testing, bug fixing, optimization, user feedback

b. Gantt Chart (Textual Representation)

Here is a text-based Gantt chart representing each task across the 5-week development cycle:

Tasks	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
Requirement Gathering										
System Design (ERD, UI, DFD)										
Database Setup										
Backend Development										
Frontend UI Implementation										
Integration (Ziteboard/Google)										
Testing & Debugging										
Deployment & Documentation										

c. Risk Management Plan

Identifying potential risks early helps in creating mitigation strategies that ensure smoother project delivery. Below is a list of anticipated risks and mitigation approaches.

Risk	Probability	Impact	Mitigation Strategy
Scope Creep	Medium	High	Lock feature list during planning; avoid mid-project changes
Data Loss / Corruption	Low	High	Regular backups of the database and codebase
Integration Issues (Ziteboard, Google)	Medium	Medium	Use modular coding; test APIs separately before merging
UI/UX Feedback Delays	Medium	Medium	Schedule early prototype reviews with tutors
Deployment Environment Conflicts	Low	High	Use standardized setup (XAMPP/WAMP); test in multiple environments
Unfamiliar Bugs / Debugging Delays	High	Medium	Allocate buffer time in Week 10 for in-depth testing

Chapter 2: Design and Implementation

2.1 Functional Requirements

Functional requirements define the specific behavior and functionalities of the Tuition Management System. These requirements were identified through user research, observation, and early prototype testing, ensuring the system meets the operational needs of tutors and educational institutes.

The functional requirements of "Tuition Management System" are:

ID	Module	Functionalities
FR-01	Student Management	Add/edit/delete student info including timezone, contact, course, fee
FR-02	Scheduling	Assign sessions daily or weekly; view upcoming schedules; manage session status

FR-03	Reports	- Generate Progress Reports (monthly academic summary) - Generate Session Reports (per-class record including topics, test scores, and comments)
FR-04	Billing	- Add New Bills with fee, due date, and notes - View full Billing History
FR-05	Transactions	Record payment amount, method (cash, online), and transaction ID
FR-06	Dashboard	Show real-time summaries: total students, today's sessions, unpaid bills, total income
FR-07	Ziteboard Integration	Launch whiteboard sessions for online classes
FR-08	Google Integration	Placeholder for future extension with Google Calendar, Drive, or Gmail

2.2 Non-Functional Requirements

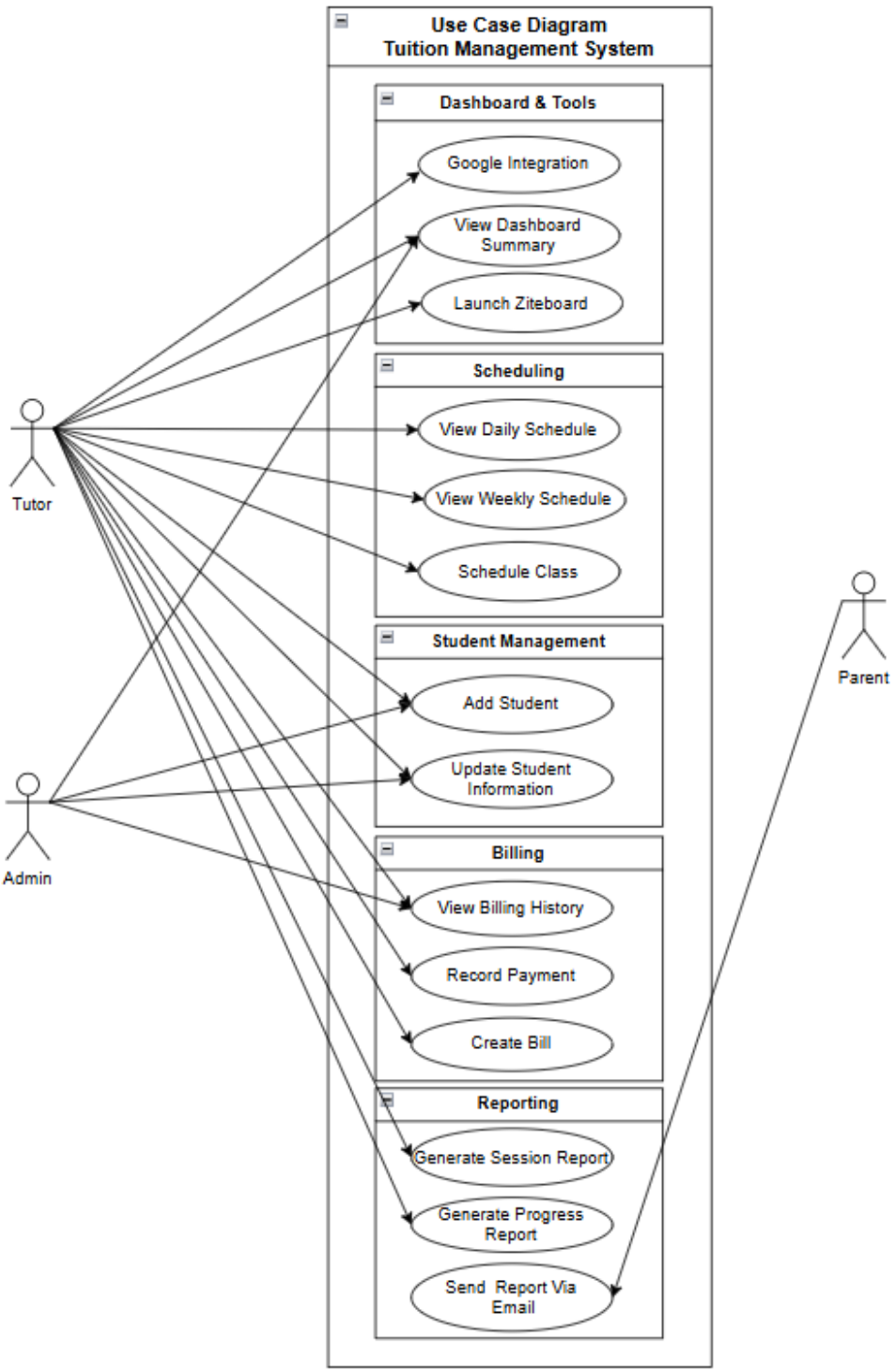
Non-functional requirements define the system's quality attributes, such as performance, usability, and security. These characteristics ensure that the system is not only functional but also stable, reliable, and user-friendly.

ID	Category	Description
NFR-01	Performance	Responsive interface with <2 seconds load time
NFR-02	Usability	Clean, tutor-friendly dashboard with form-based inputs
NFR-03	Security	Sensitive data (e.g., billing info) stored securely
NFR-04	Maintainability	Modular code structure for easy updates and scaling
NFR-05	Scalability	Capable of handling 100+ students and 1000+ sessions
NFR-06	Portability	Runs on any modern browser; deployable via XAMPP/WAMP or live server
NFR-07	Availability	Localhost or hosted environment; ready for online deployment

2.3 Object-Oriented System Design Using UML

Object-Oriented System Design (OOSD) is a design methodology that organizes a system around objects—entities that combine data and behavior. For the Tuition Management System, OOSD was applied using UML (Unified Modeling Language), a standardized way to visualize and document the system's architecture.

A. Use Case Diagram for Tuition Management System



B. Use Case Description:

Use Case 1: Add Student

Use Case Name	Add Student
Actor	Tutor, Admin
Description	Allows the tutor or admin to add a new student to the system.
Precondition	Tutor/Admin is logged in.
Postcondition	New student record is saved in the std_info table.
Normal Flow	1. Tutor selects "Add Student" 2. Enters student details 3. Clicks "Submit"
Alternate Flow	1. Required fields missing → error message shown 2. Duplicate entry → prompt warning

Use Case 2: Schedule Class

Use Case Name	Schedule Class
Actor	Tutor
Description	Allows tutor to schedule a class for a student on a specific date and time.
Precondition	Student must already exist in the system.
Postcondition	Schedule is saved in the schedule table.
Normal Flow	1. Tutor selects stud ent 2. Chooses date and time 3. Clicks "Schedule"
Alternate Flow	1. If time slot overlaps with another session → warning shown

Use Case 3: Generate Session Report

Use Case Name	Generate Session Report
Actor	Tutor
Description	Records academic details of a session including topics and comments.
Precondition	The class session must be marked as completed.
Postcondition	Session report is saved in the report table.
Normal Flow	1. Tutor selects session 2. Enters report fields 3. Clicks "Submit"
Alternate Flow	1. Session is not completed → report button is disabled

Use Case 4: Generate Progress Report

Use Case Name	Generate Progress Report
Actor	Tutor
Description	Summarizes a student's monthly academic performance.
Precondition	Student must have completed sessions.
Postcondition	Progress report is compiled and saved/exported.
Normal Flow	1. Select student and month 2. Compile data 3. Generate and export report
Alternate Flow	1. No sessions found → "No data to generate report" message shown

Use Case 5: Create Bill

Use Case Name	Create Bill
Actor	Tutor, Admin
Description	Creates a billing entry linked to student sessions.
Precondition	Student and session data must exist.
Postcondition	Bill is saved in the bill table.
Normal Flow	1. Select student and amount 2. Set due date and notes 3. Click "Create Bill"
Alternate Flow	1. Duplicate bill detected → system prompts a warning

Use Case 6: Record Payment

Use Case Name	Record Payment
Actor	Tutor, Admin
Description	Records the payment transaction for a generated bill.
Precondition	The bill must exist and be unpaid.
Postcondition	Payment is saved in the txn table; bill is marked as paid.
Normal Flow	1. Select bill 2. Enter amount, transaction ID, payment method 3. Submit
Alternate Flow	1. Missing transaction ID → error prompt

Use Case 7: View Dashboard Summary

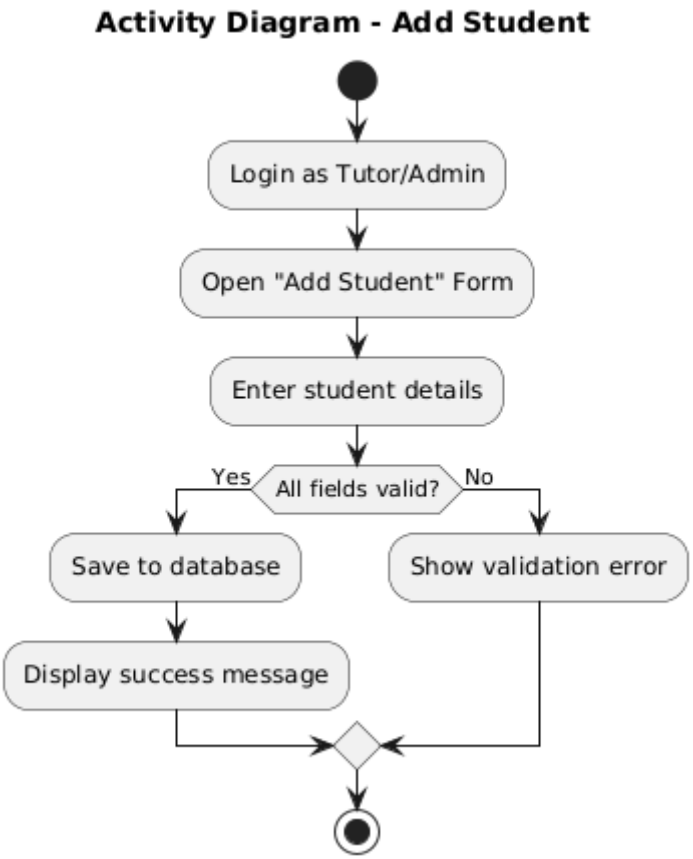
Use Case Name	View Dashboard Summary
Actor	Tutor, Admin
Description	Shows real-time overview of students, income, sessions, and billing status.
Precondition	System has existing records.
Postcondition	Dashboard widgets display live data.
Normal Flow	1. User logs in 2. Dashboard loads with real-time summaries
Alternate Flow	1. No records exist → show default "0" values

Use Case 8: Launch Ziteboard

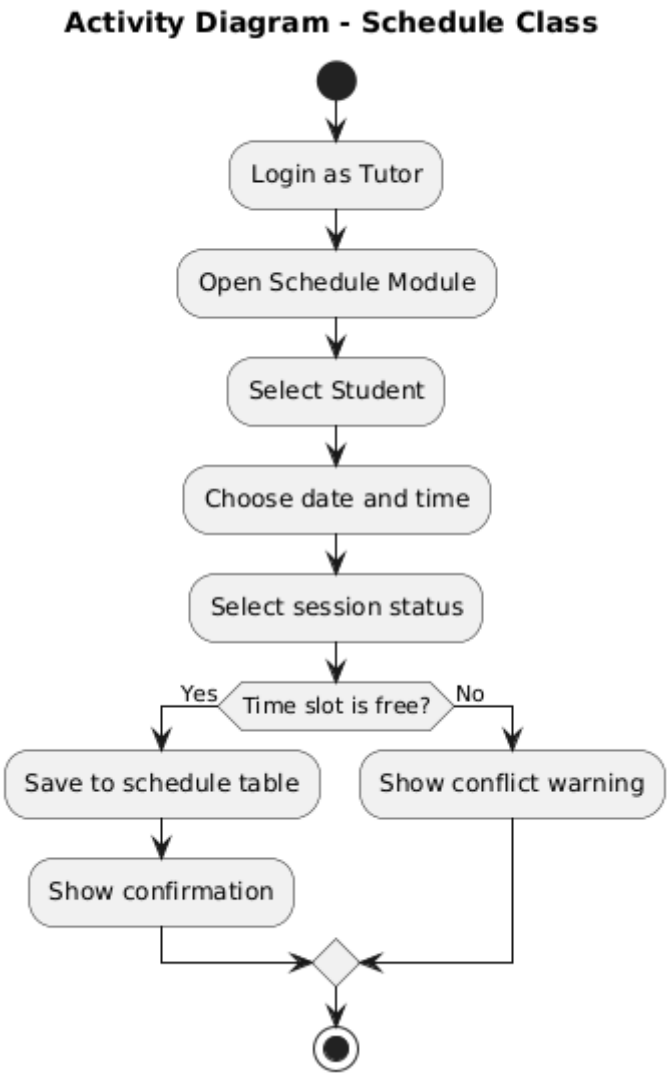
Use Case Name	Launch Ziteboard
Actor	Tutor
Description	Opens Ziteboard for real-time whiteboard teaching.
Precondition	Tutor is in an active session.
Postcondition	Ziteboard is opened in a new browser tab.
Normal Flow	1. Click "Ziteboard" button 2. Redirect to Ziteboard session
Alternate Flow	1. Internet not available → prompt warning

C. Activity Diagram

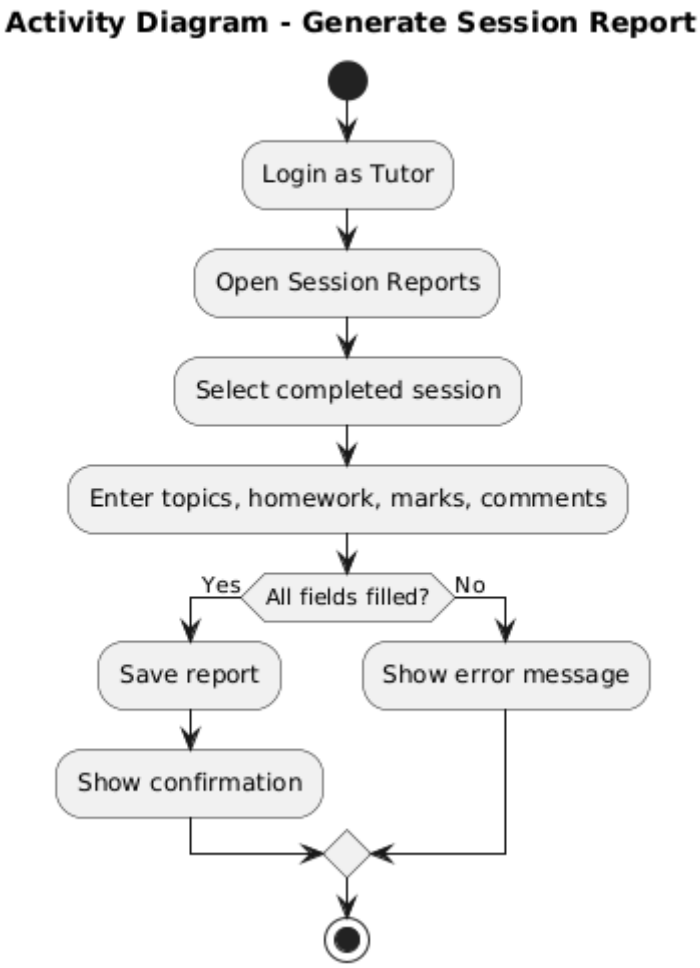
1. Add Student



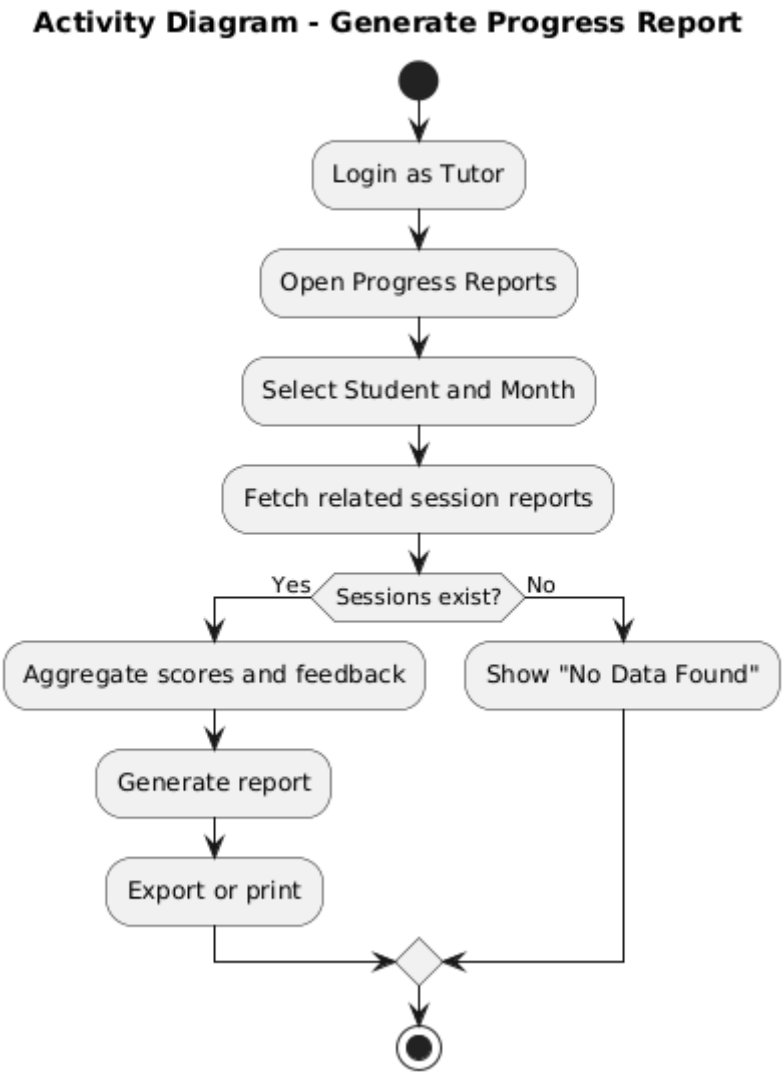
2. Schedule Class



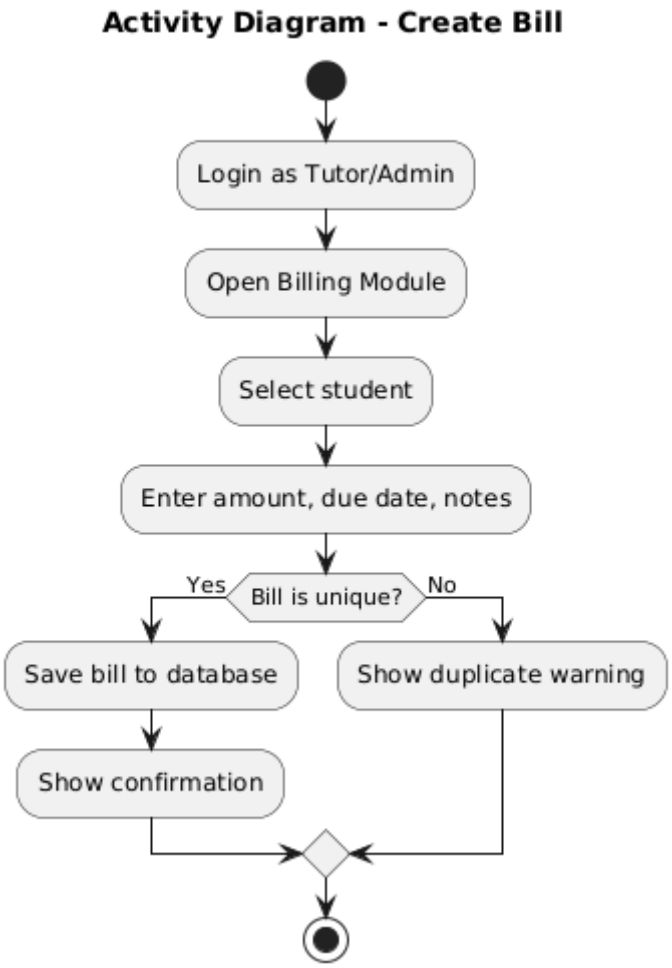
3. Session



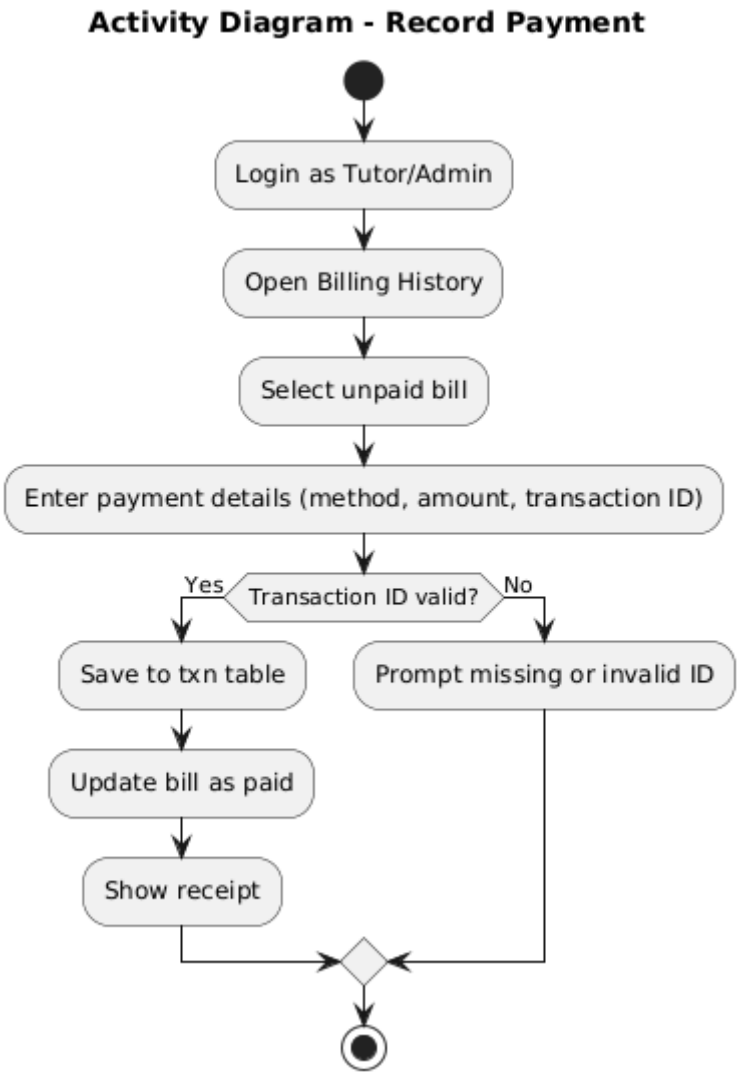
4. Report



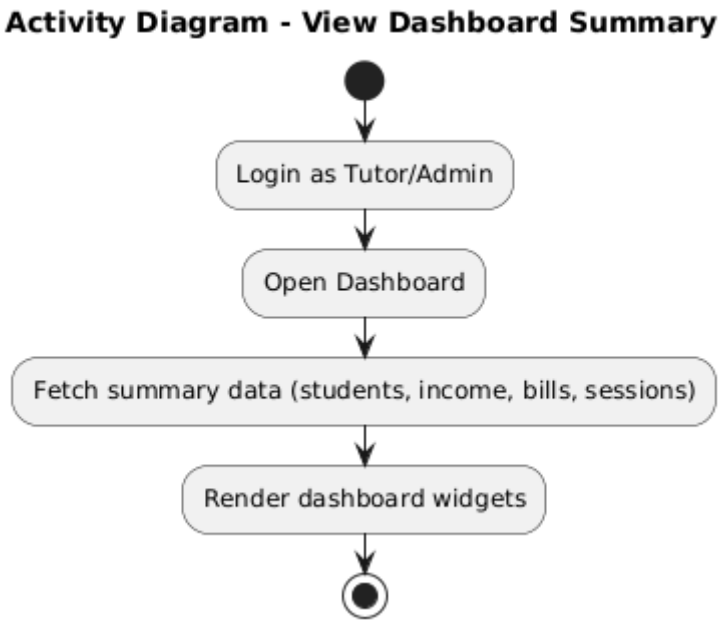
5. Create Bill



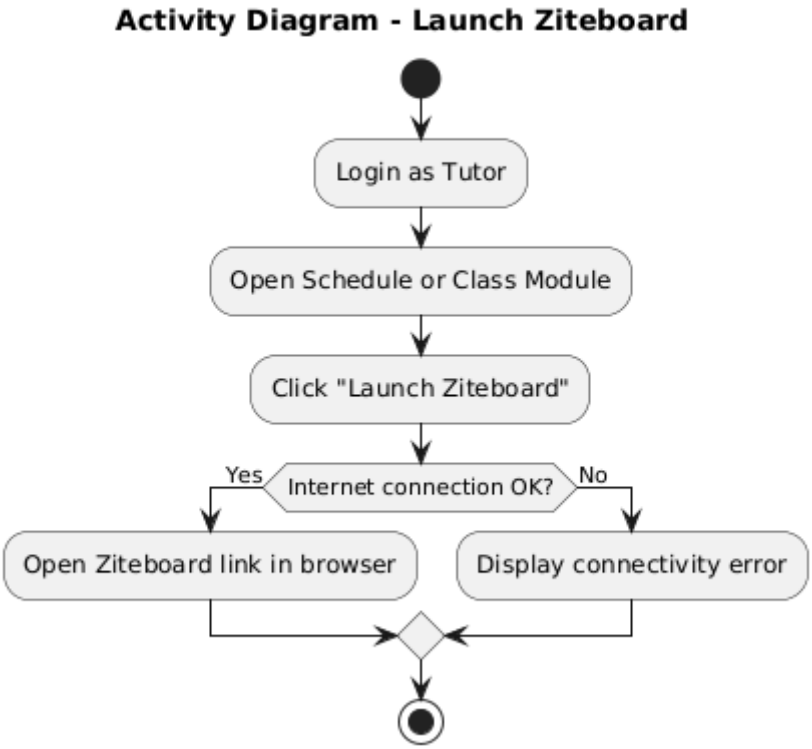
6. Payment



7. View Dashboard

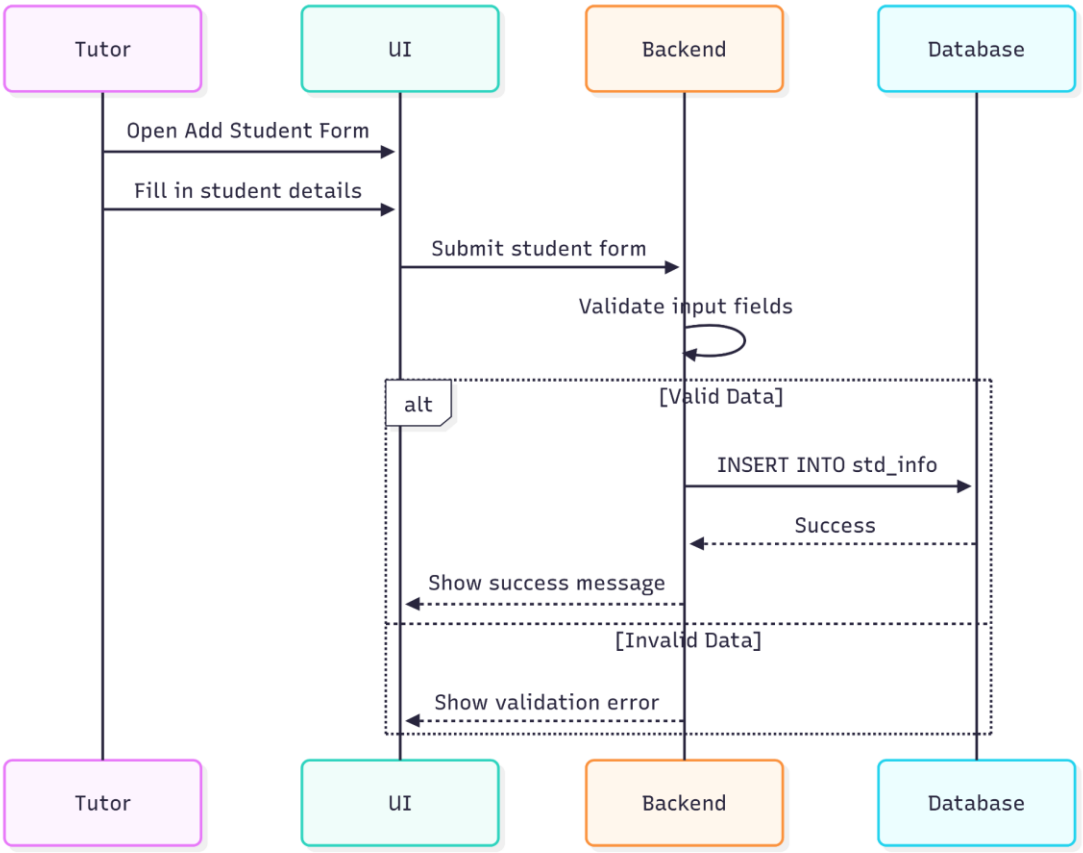


8. Ziteboard

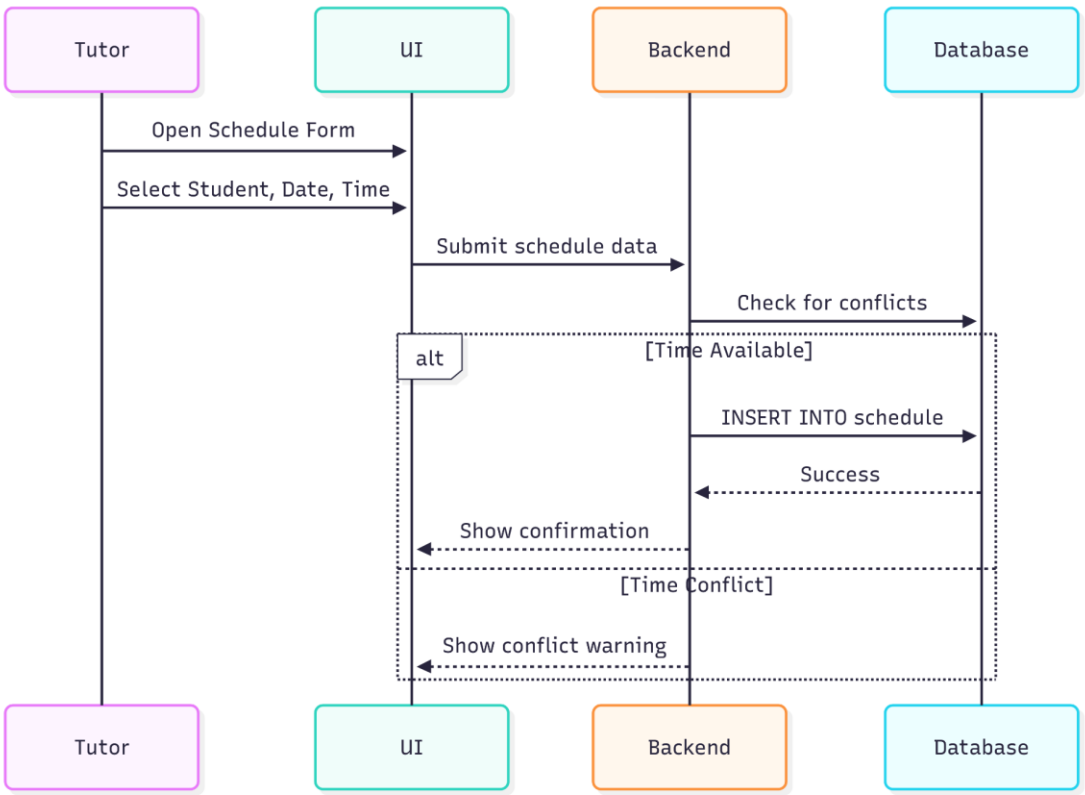


D. Sequence Diagram

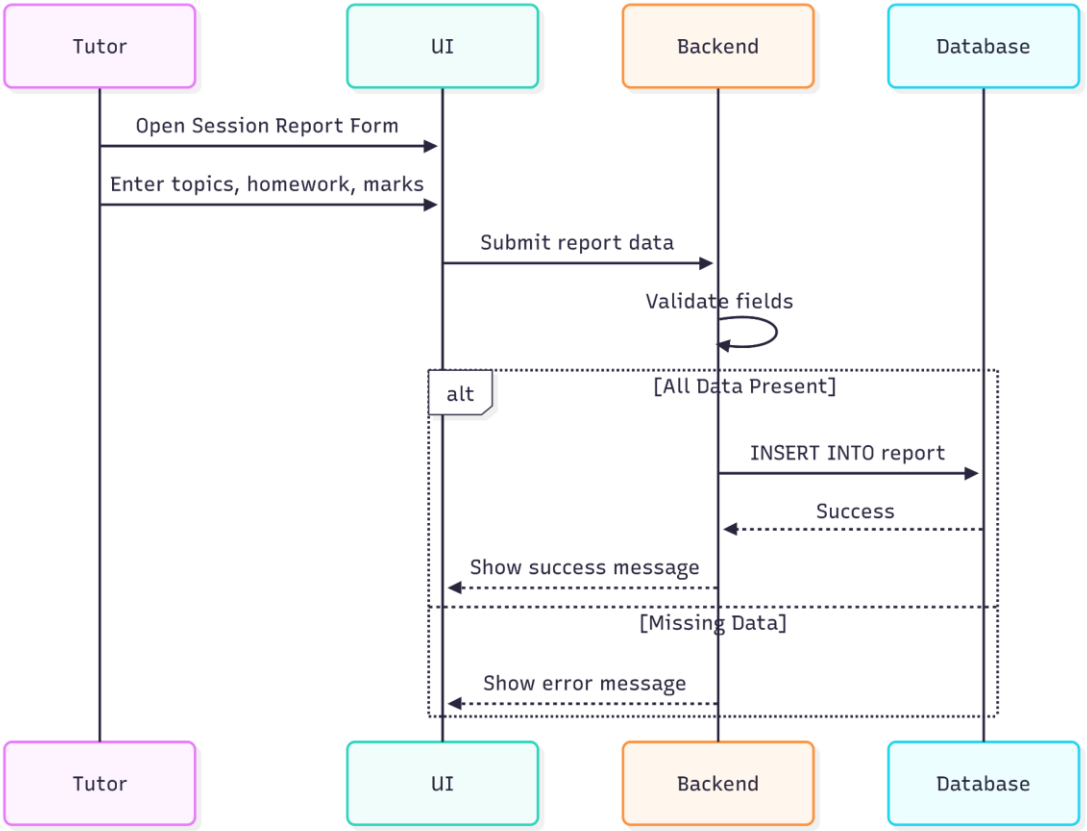
1. Add Student:



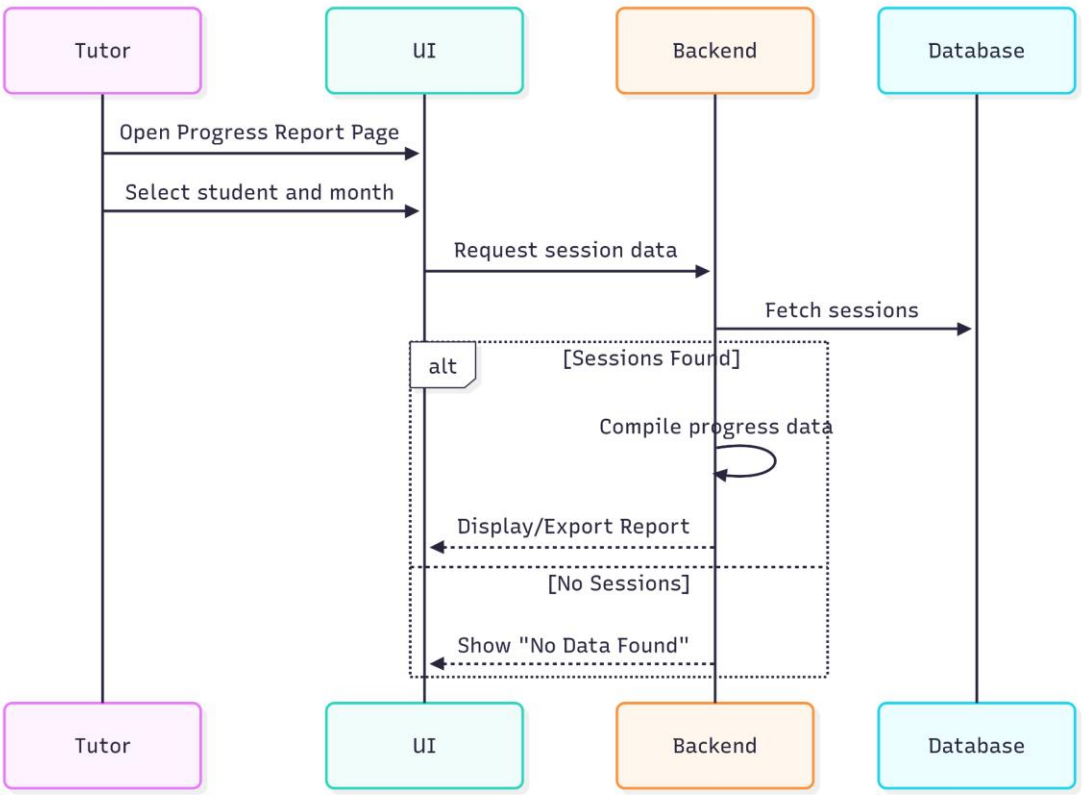
2. Schedule Class:



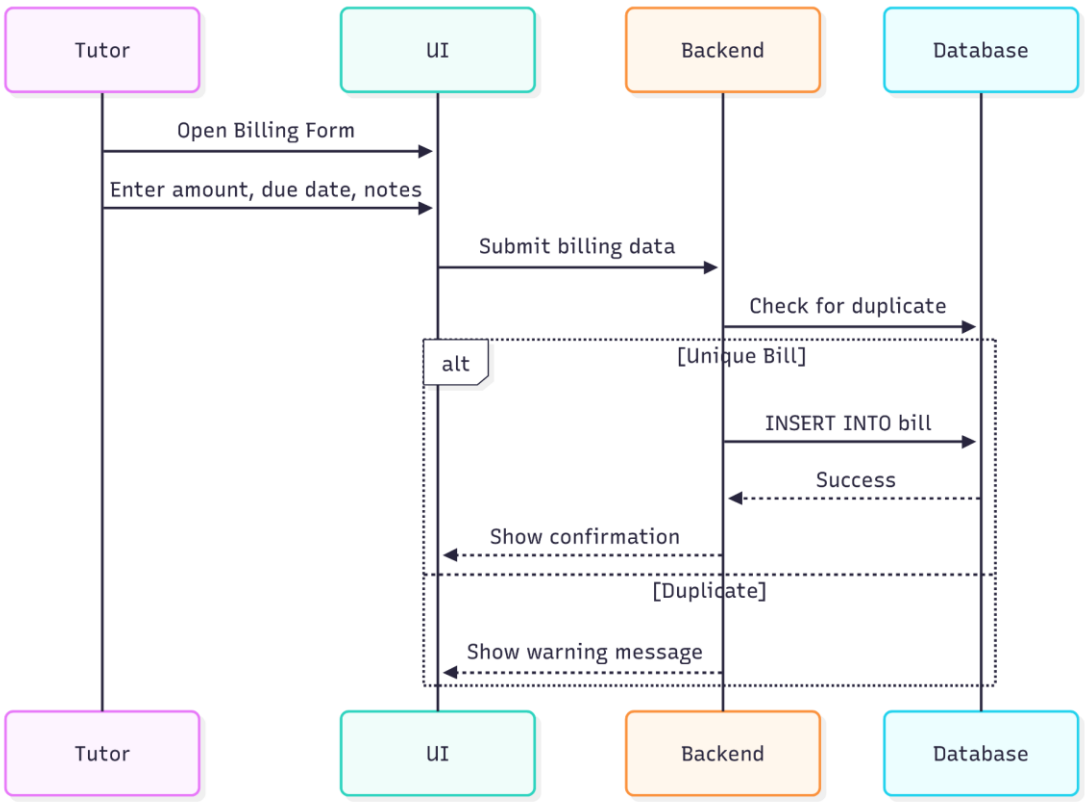
3. Generate Session Report:



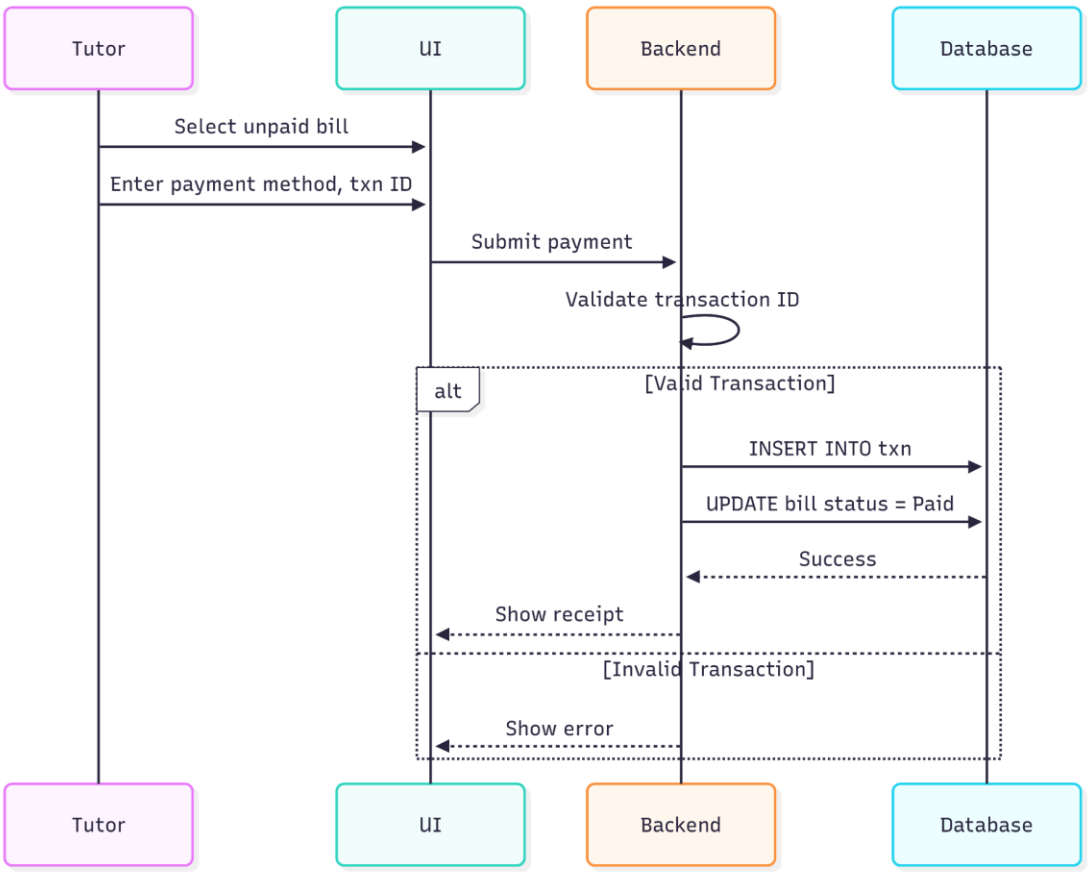
4. Generate Progress Report:



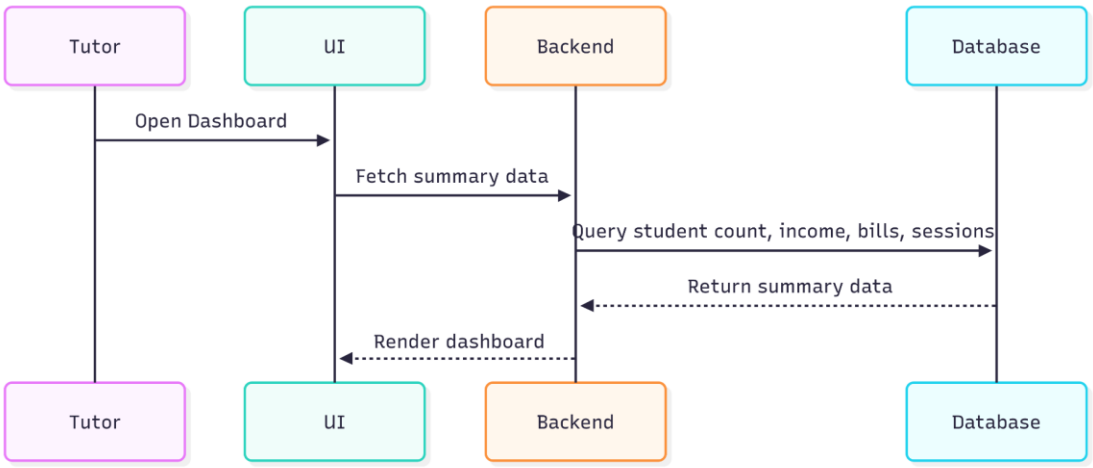
5. Create Bill:



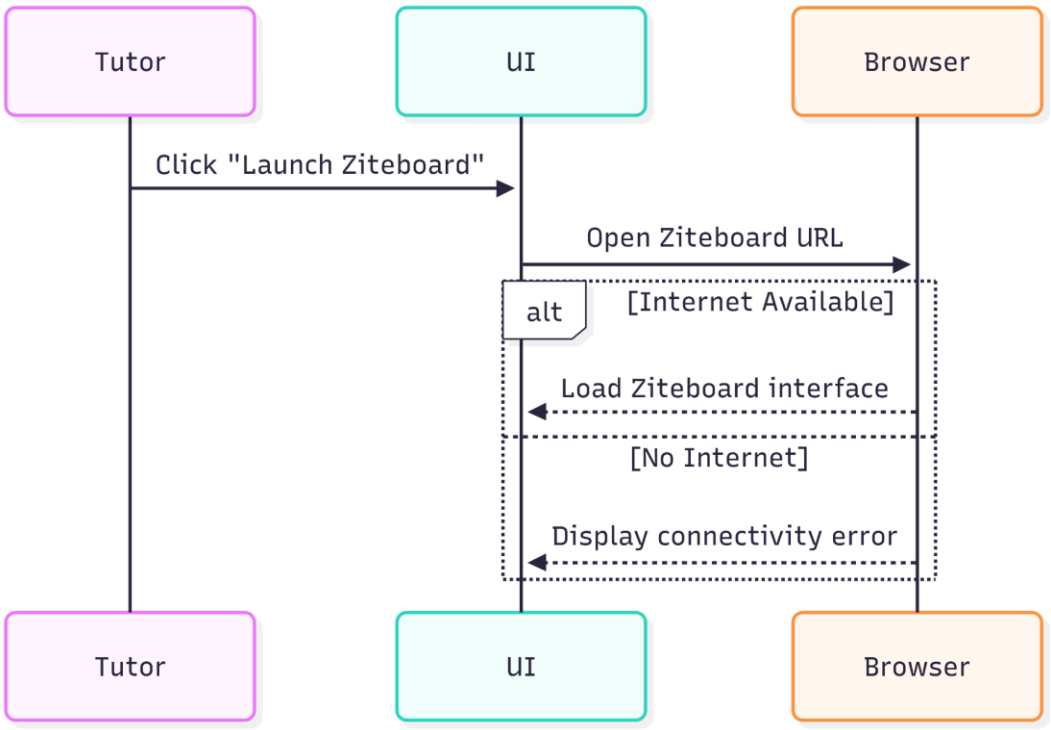
6. Record Payment:



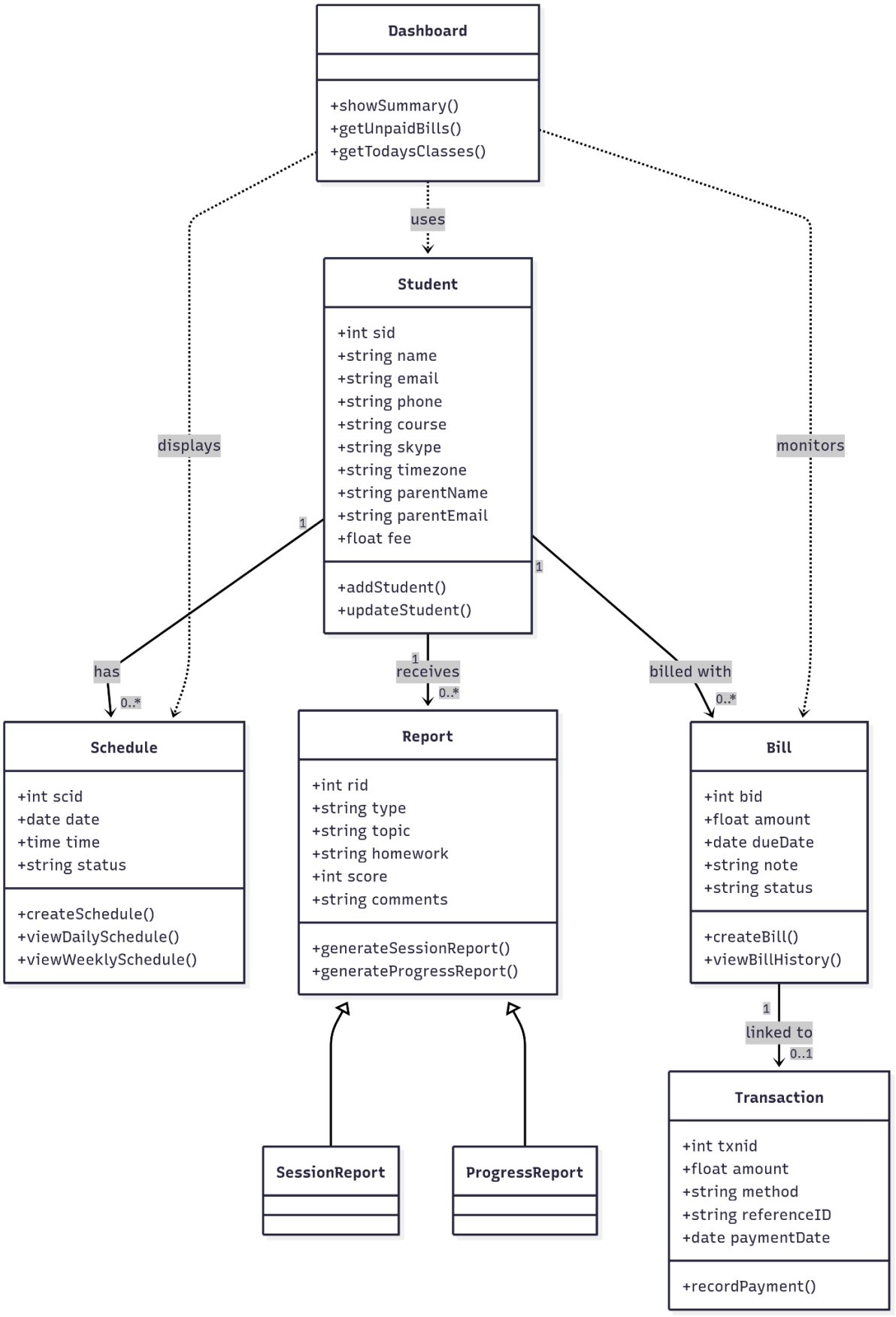
7. View Dashboard Summary:



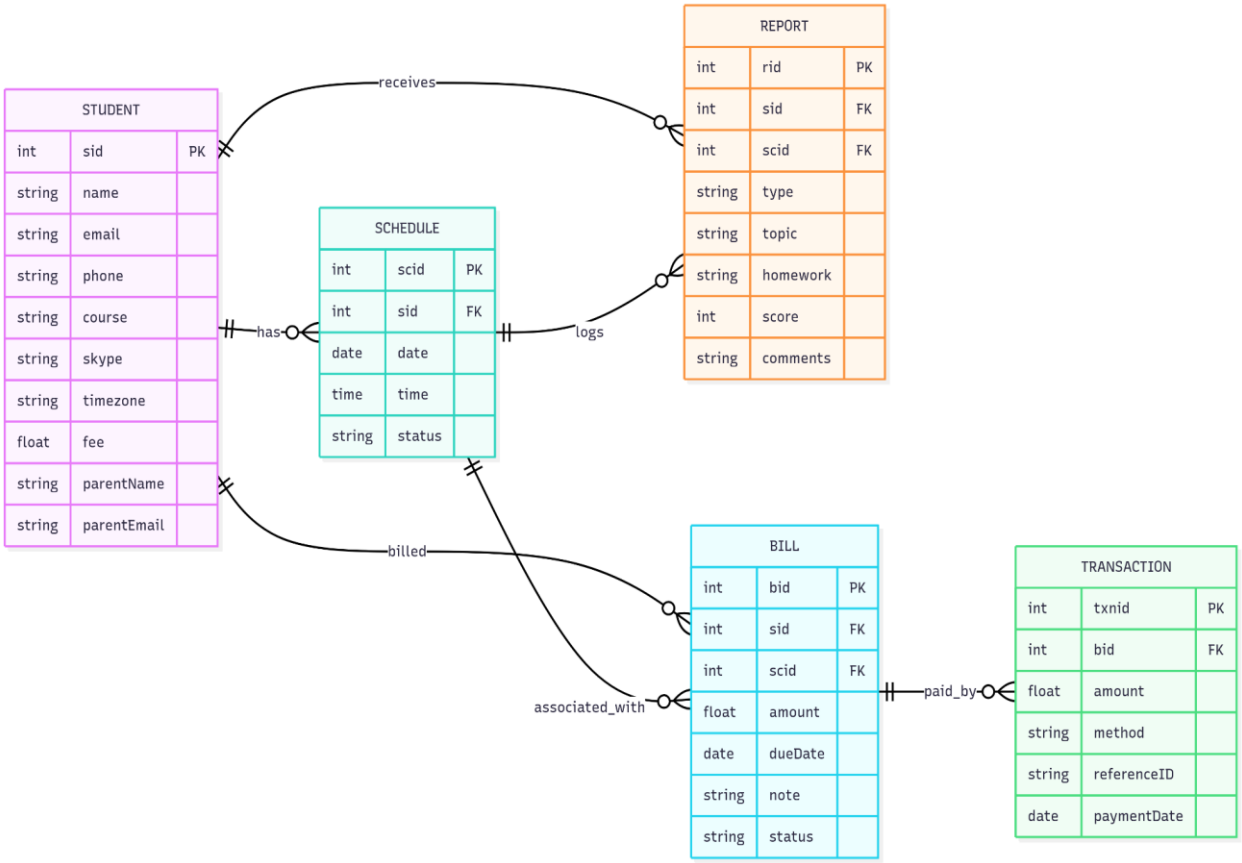
8. Launch Ziteboard:



E. Class Diagram



F. ER Diagram



2.4 Architectural Model

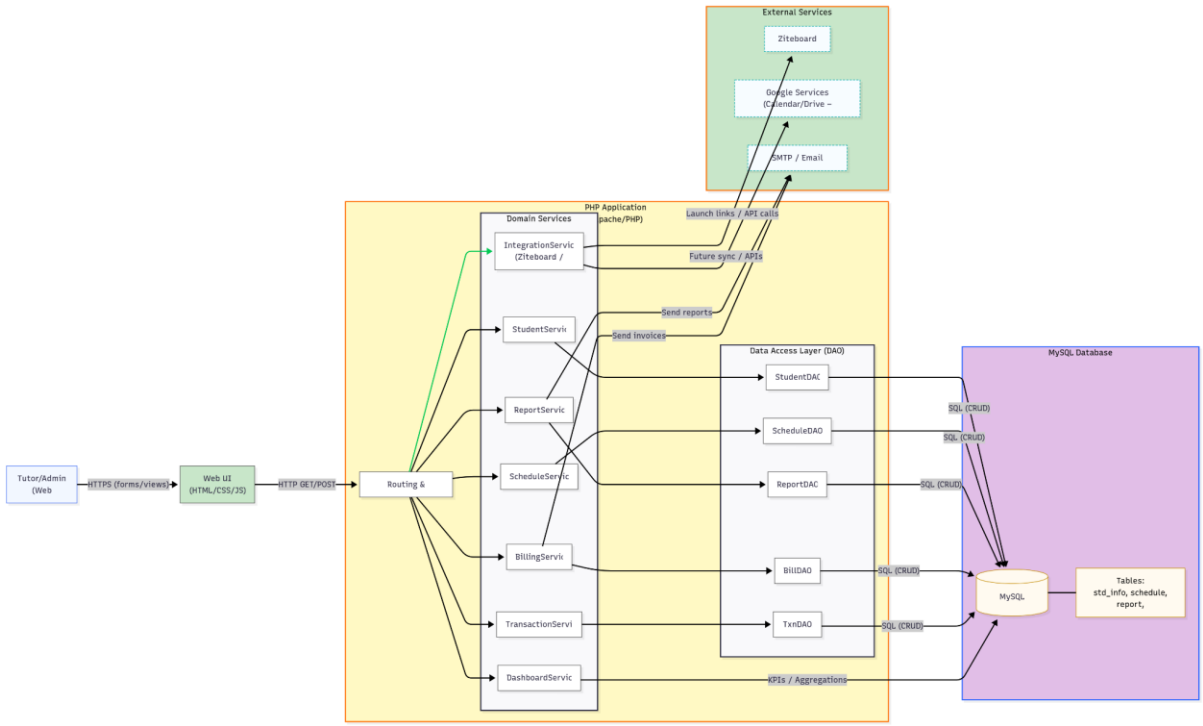
The architectural model of your Tuition Management System represents how the system is logically structured and how its different components interact.

It follows a multi-tier architecture.

2.4.1 Component Architecture (Logical Model)

The Component Architecture illustrates the logical breakdown of the Tuition Management System into key modules and how these modules interact with one another. It focuses on the structural organization of the software, showing the relationships between user-facing components, business logic, and data handling layers without describing physical deployment.

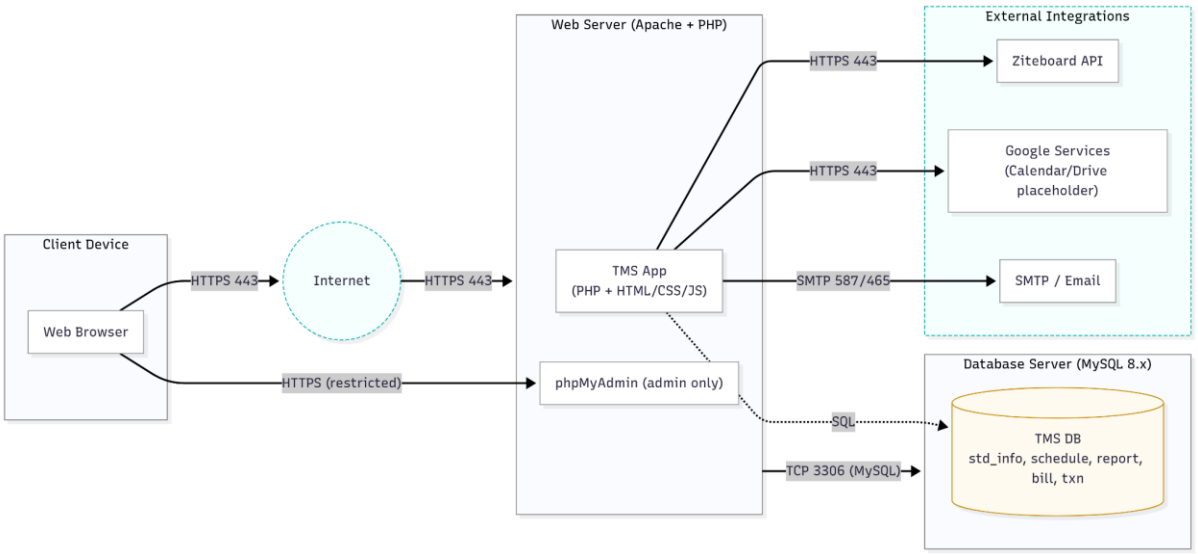
a. Logical Model:



2.4.2 Deployment Architecture (Physical Model):

The Deployment Architecture describes how the Tuition Management System's software components are mapped onto the physical hardware and network infrastructure. It shows the real-world arrangement of the system, including clients, servers, databases, and external services, and how they communicate.

b. Physical Model



Chapter 3: Software Testing

Software testing is an essential process to ensure that the system meets its functional and non-functional requirements and is free of defects. Testing was performed at various stages of development to identify errors, ensure data accuracy, and confirm that user interactions behave as expected.

3.1 Testing Features

The key testing areas of the Tuition Management System included:

Feature	Description
Form Validation	Ensures that forms like Add Student, Schedule Class, and Create Bill cannot be submitted without required fields.
Database Connectivity	Validates successful insertion, updating, and retrieval of data from MySQL tables (std_info, schedule, bill, report, txn).
Field-Specific Checks	Validates email format, fee as a numeric field, unique student names, and date selections for schedule and billing.
Session Linking	Confirms that session reports and billing entries are properly linked to a valid schedule ID and student ID.
Dashboard Metrics	Verifies that summaries for income, today's sessions, and unpaid bills reflect real-time data.
Module Navigation	Ensures proper redirection between dashboard modules without broken links or missing data.
External Integration	Validates that the Ziteboard link launches in a new tab and Google integration placeholder doesn't break the UI.

3.2 Testing Strategies

The testing process followed these common strategies:

1. Unit Testing

- Each PHP module and backend function was tested independently (e.g., addStudent(), createBill()).
- Focused on function output, database response, and error messages.

2. Integration Testing

- Ensured smooth data flow between modules.
- Example: Student data added → Schedule created → Report submitted → Bill generated → Payment recorded.

3. Validation Testing

- Tested all form inputs for empty fields, invalid formats, and edge cases.
- Included checks for date/time conflicts, numeric-only input, and unique constraints.

4. User Interface Testing

- Checked button clicks, dropdowns, calendars, and labels.
- Ensured intuitive design, responsive layout, and usability.

5. Regression Testing

- After any code update (e.g., adding report filters), earlier functionalities were re-tested to prevent new bugs.

6. System Testing

- Complete end-to-end testing of the full application in a simulated live environment.
- Includes test case execution (see Section 3.3)

3.3 System Testing

System testing was performed across all modules of the Tuition Management System. Below is a sample set of test cases with execution results

Test Case Table:

Test Case ID	Module	Test Description	Input	Expected Result	Actual Result	Status
TC_01	Add Student	Submit form with all valid data	Valid name, email, fee, course	Student added to DB, success message shown	As Expected	✅ Passed
TC_02	Add Student	Submit form with empty fields	Empty name, course	Validation error displayed	As Expected	✅ Passed
TC_03	Schedule Class	Create schedule with overlapping time	Same student, same date/time	Conflict warning displayed	As Expected	✅ Passed

TC_04	Schedule Class	Valid schedule creation	Student ID: 12, Date: 2025-08-01 6PM	Entry saved in schedule table	As Expected	✅ Passed
TC_05	Session Report	Enter incomplete report fields	Missing topic or comments	Form validation triggered	As Expected	✅ Passed
TC_06	Progress Report	Generate report with no sessions	Student ID: 15, Month: August	"No data found" message shown	As Expected	✅ Passed
TC_07	Billing	Create bill with valid info	Fee: 1200, Due: 2025-08-10	Entry saved in bill table	As Expected	✅ Passed
TC_08	Record Payment	Record payment without txn ID	Missing ref ID	Validation error shown	As Expected	✅ Passed
TC_09	Dashboard	Load dashboard with no data	N/A	All metrics show "0" or empty state	As Expected	✅ Passed
TC_10	Ziteboard Launch	Open Ziteboard link	Click "Launch Ziteboard"	New tab opens with ziteboard.com	As Expected	✅ Passed

Chapter 4: Deployment and Maintenance

4.1 Overview

Deployment and maintenance are critical stages in the software development process, ensuring that the system is not only released smoothly but also remains stable, scalable, and adaptable to user needs after launch. The Tuition Management System followed an Agile-based development model and adhered to the standard Software Release Life Cycle (SRLC) to guide its iterative releases, user feedback loops, and post-deployment updates.

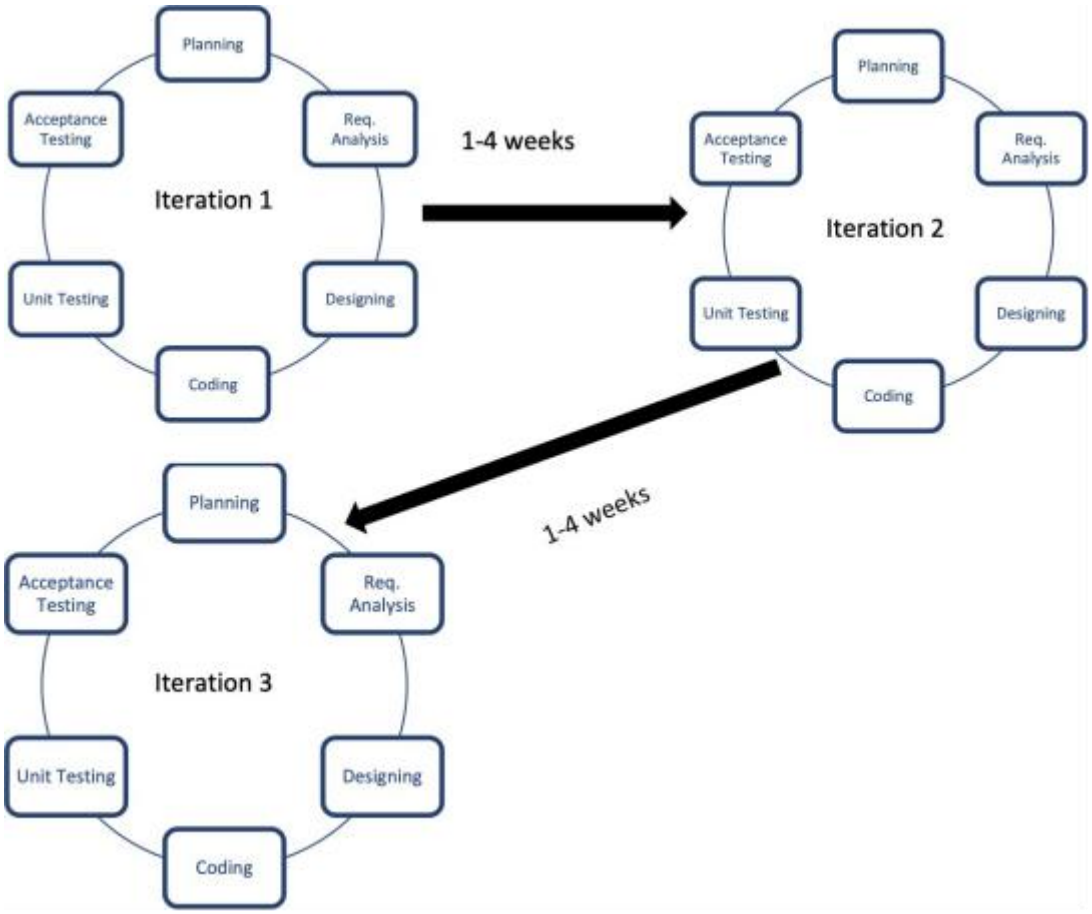
4.2 Agile-Based Deployment Strategy

From the very beginning, the development of this system adopted Agile methodologies, focusing on:

- Incremental development (modules built one at a time: student → schedule → billing → reporting)
- Continuous testing and integration
- Early prototype delivery for feedback
- Backlog-driven iteration planning

- Short sprint cycles (1/4 week per Iteration)

Each module was developed and reviewed independently, allowing for faster validation, feedback, and refinements before final deployment.



4.3 Software Release Life Cycle (SRLC)

The **SRLC** phases were followed to structure the release process of the Tuition Management System. Below is a description of each phase with how it was applied:

1. Pre-Alpha Stage

- **Goal:** Basic internal development with no complete functionality
- **Activities:** (`std_info`, `schedule`, `bill`, `report`, `txn`)
    - Database schema designed
    - Initial codebase setup (backend in PHP, frontend with HTML/CSS)
    - Internal feature planning and documentation
- **Tools Used:** phpMyAdmin, VS Code, Trello (task tracking)

2. Alpha Release

- **Goal:** First functional version with core modules (not fully stable)
- **Delivered Modules:**
    - Student Management
    - Scheduling (daily/weekly)
    - Billing (Add + History)
- **Testing:** Conducted by developer (unit and manual integration testing)
- **Limitations:** No report generation or Ziteboard integration at this stage

3. Beta Release

- **Goal:** Feature-complete version made available for external review
- **New Features Added:**
    - Session and Progress Reporting
    - Dashboard summary cards
    - Ziteboard and Google tool integration
- **Testers:** 2 local tutors tested the system for real use cases
- **Feedback Loop:** Adjusted form labels, corrected timezone handling, improved validation

4. Release Candidate (RC)

- **Goal:** Ready-for-deployment version after bug fixes and polishing
- **Actions Taken:**
    - Security validation on inputs
    - Session-level filtering for report generation
    - Final bug fixing and interface cleanup
- **Documentation Completed:** User Manual, Deployment Guide, Technical Report

5. Production (Stable Release)

- **Goal:** Final deployment on localhost or live server
- **Deployment Environment:**
    - **Localhost** via XAMPP
    - **Live option:** Compatible with CPanel or shared PHP hosting
- **Version Tag:** v1.0 Stable

6. Post-Release Maintenance

- Activities:
    - Regular backups of *.sql and *.php files
    - Version tracking using Git (local)
    - Bug reports logged and patched incrementally
    -
- Planned Enhancements:

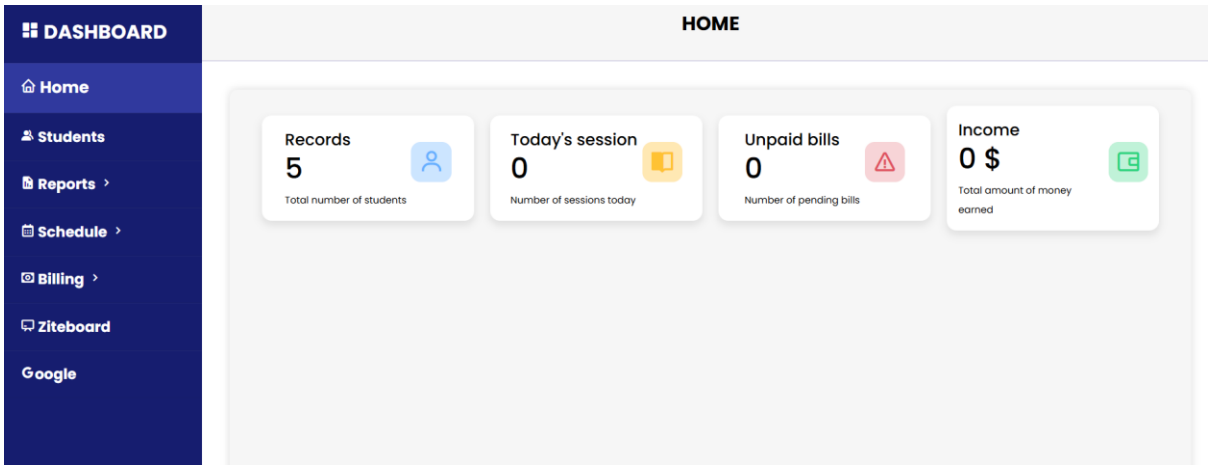
    ➔ Multi-user login system (Admin, Tutor roles)
    ➔ Email notifications for bills and reports
    ➔ Data visualization for progress analytics

4.4 Deployment Workflow (Summary)

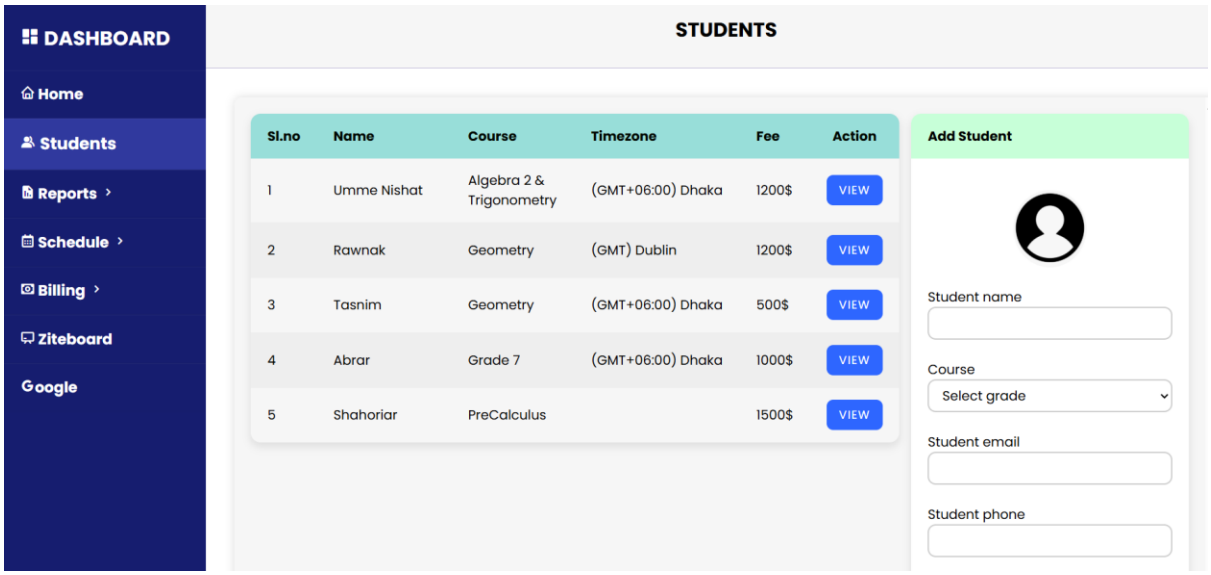
Step	Description
Development Environment	VS Code + XAMPP + MySQL
Testing Strategy	Manual testing after each sprint/module
Deployment Mode	Localhost (initial) → Optional CPanel hosting
Backup Approach	Manual backup of DB and codebase weekly
Logging & Error Handling	PHP error logs, manual tracking for major bugs
User Feedback Loop	Included post-beta with real tutors

Chapter 5: User Manual

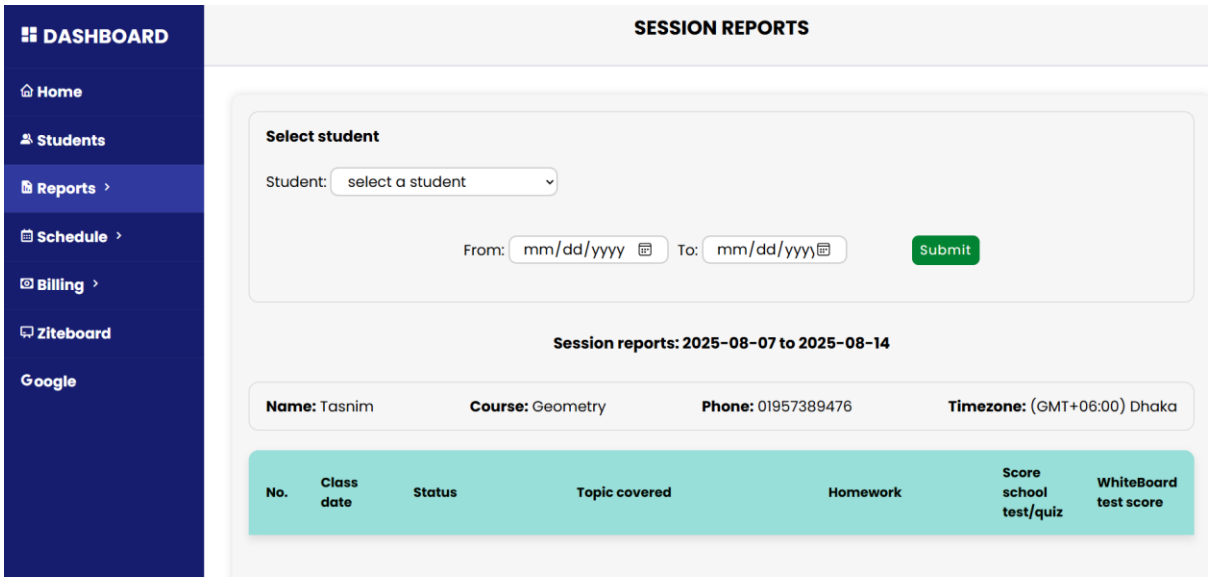
5.1 Dashboard



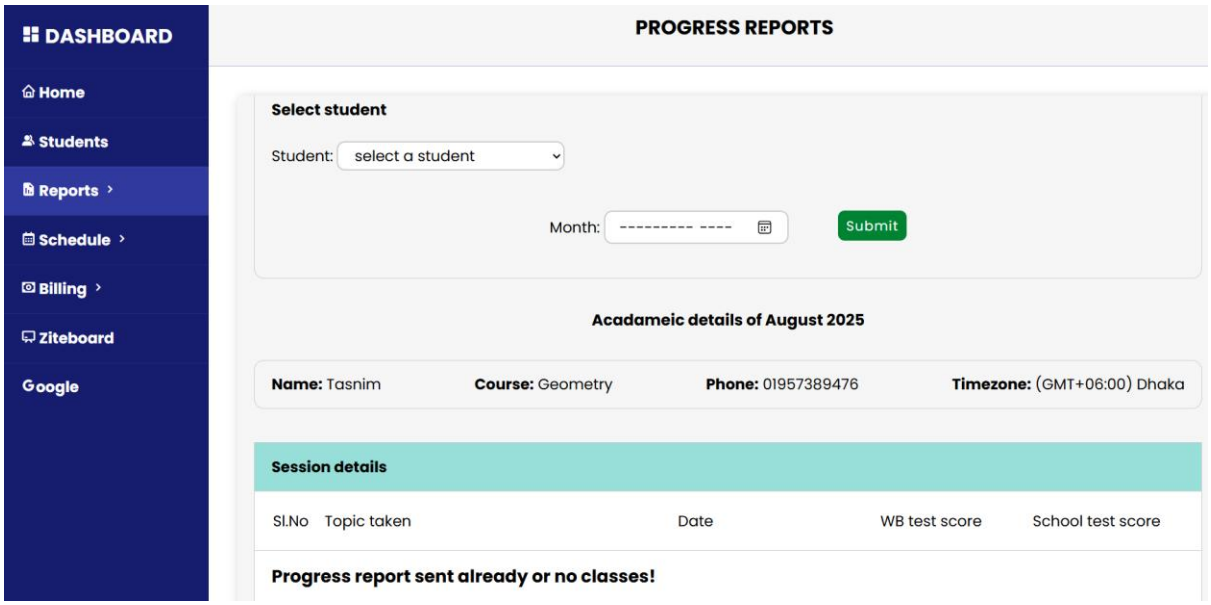
5.2 Add Student



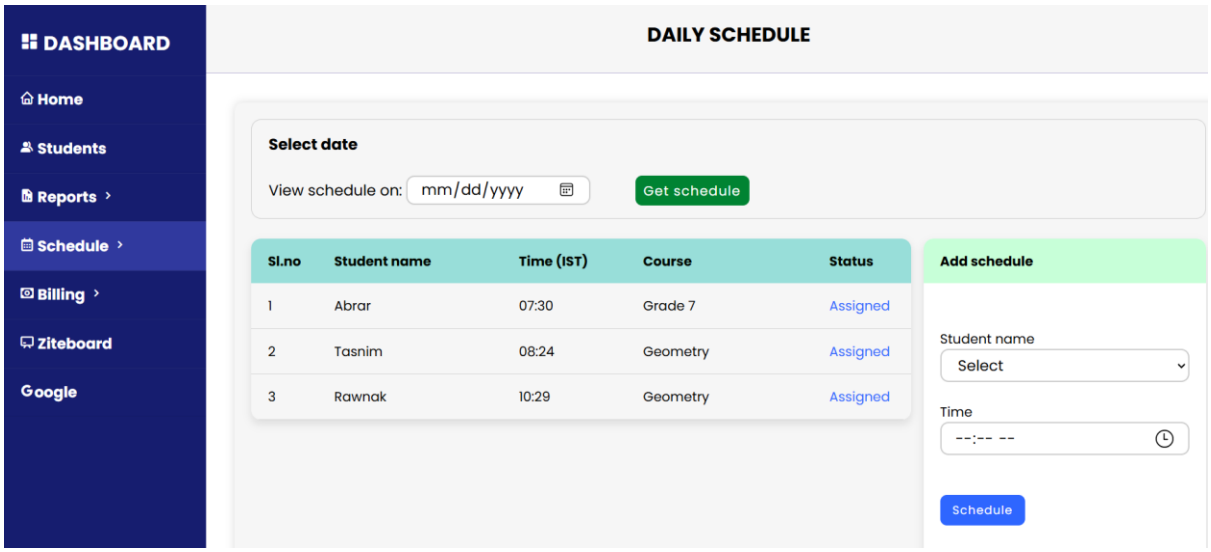
5.3 Session Reports



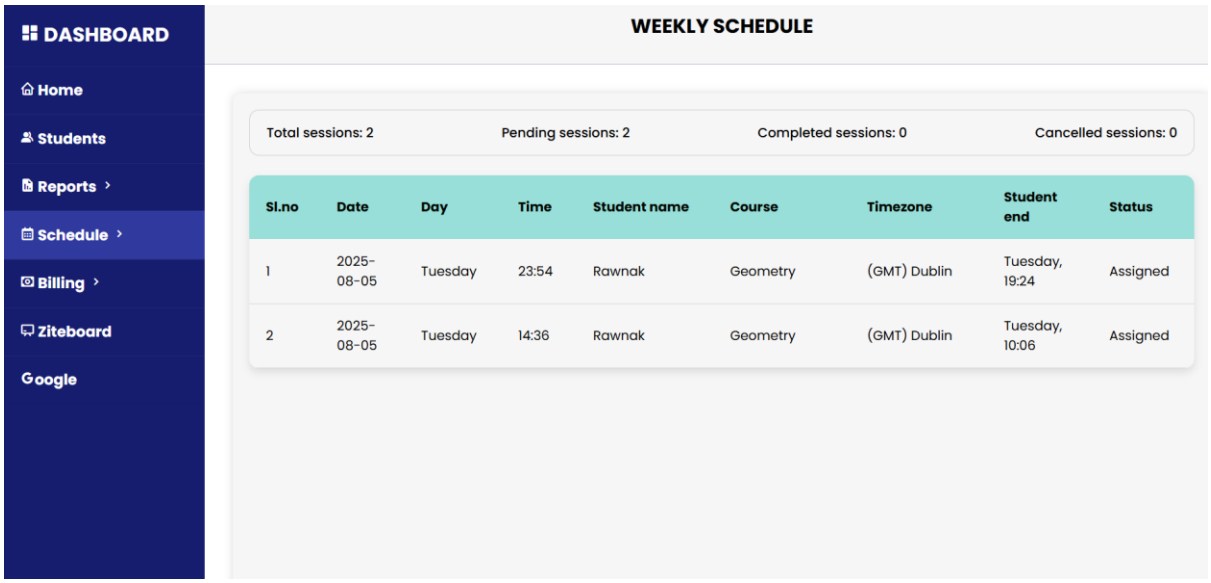
5.4 Progress Report



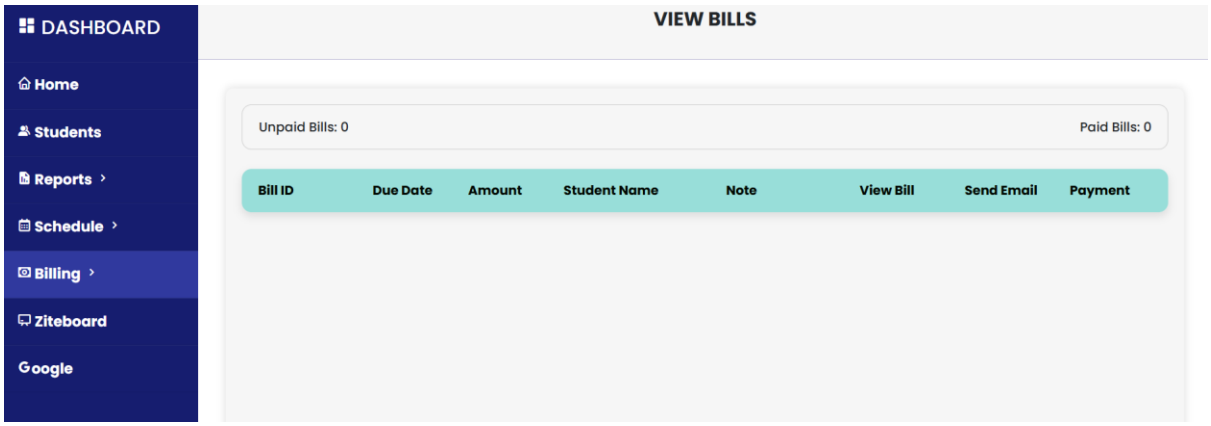
5.5 Daily Schedule



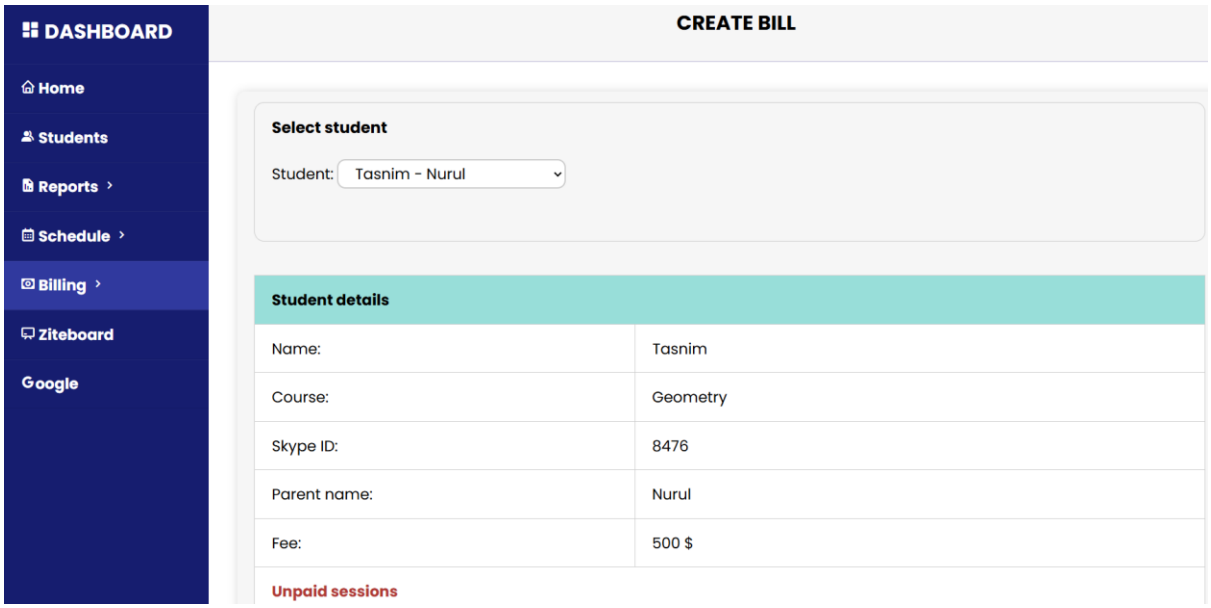
5.6 Weekly Schedule



5.7 Billing



5.8 Create Bill



Chapter 6: Project Summary

6.1 Introduction

This project aimed to develop a Tuition Management System — a centralized, user-friendly web application that simplifies and automates the administrative tasks of private tutors and small educational centers. The system covers student enrollment, class scheduling, billing, report generation, and tool integrations to improve productivity and reduce manual workload.

6.2 Objectives Achieved

The project successfully achieved the following core objectives:

- Built a structured and normalized MySQL database to manage student, schedule, transaction, report, and billing data.
- Developed an intuitive dashboard interface with real-time stats (students, income, sessions, billing).
- Implemented student management, daily/weekly scheduling, and billing with both "add new" and "history" features.
- Included academic tracking via session reports and monthly progress reports.
- Integrated third-party tools like Ziteboard for online whiteboarding and placed support for Google services.
- Applied Agile methodology and followed the Software Release Life Cycle (SRLC) through stages: Pre-Alpha → Alpha → Beta → Release Candidate → Production → Maintenance.

6.3 Technologies Used

- Front-end: HTML, CSS, JavaScript
- Back-end: PHP (Procedural)
- Database: MySQL (phpMyAdmin)
- Server Environment: XAMPP (localhost)
  Tools: Ziteboard (external), Google integration placeholders

6.4 Development Methodology

The system was developed using the Agile model, which allowed for:

- Iterative development in focused sprint cycles
- Regular testing and feedback from target users
- Continuous refinement of features based on usability

Each module (Student, Schedule, Billing, Reports) was completed as an independent sprint, then integrated and tested collectively.

6.5 Challenges Faced

Challenge	Resolution
Database normalization and linking multiple entities	Applied foreign keys and proper indexing
Avoiding duplicate billing or session entries	Implemented validation checks and constraints
Timezone handling for international students	Added timezone field in std_info table
User feedback on interface complexity	Simplified forms and navigation after beta feedback

6.6 Outcomes

The system proved to be:

- Practical – directly solves the real-life problems of tutors
- Flexible – scalable for future features (multi-user login, payment gateway)
- Lightweight – can run on localhost or be deployed on shared hosting
- Reliable – includes validations, backups, and clean structure

Tutors testing the system found it significantly improved their workflow — especially with billing and academic reporting automation.

6.7 Future Scope

While the Tuition Management System is complete and functional, the following improvements are planned:

- Role-based login system (Admin, Tutor, Parent)
- Email notifications for bills and performance reports
- Analytics dashboards for visual performance tracking
- Online payment gateway integration (e.g., Razorpay, PayPal)
- Export and import data modules (CSV, Excel)

6.8 Conclusion

The Tuition Management System is a complete and scalable solution tailored for the needs of independent tutors and small institutes. From ideation to deployment, the system was designed to be simple, efficient, and impactful. By automating daily operations and providing centralized control, it helps tutors focus more on teaching and less on administration.

6.9 References

[1]. Sommerville, I. (2016). Software Engineering (10th ed.). Pearson Education. (Guidelines on software lifecycle models, Agile practices, and requirements engineering.)
[2]. Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill Education. (Comprehensive coverage of SRLC phases, project planning, and deployment strategies.)

[3]. Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd ed.). Addison-Wesley. (Reference for UML activity diagrams, use case diagrams, and object-oriented design.)

[4]. phpMyAdmin. (n.d.). phpMyAdmin Documentation. Retrieved from: https://docs.phpmyadmin.net/ (Used for MySQL database management during project development.)

[5]. W3Schools. (n.d.). PHP Tutorial. Retrieved from: https://www.w3schools.com/php/ (Reference for implementing backend functionalities in PHP.)

[6]. MySQL. (n.d.). MySQL 8.0 Reference Manual. Oracle. Retrieved from: https://dev.mysql.com/doc/ (Documentation for SQL queries, database schema design, and normalization.)

[7]. Ziteboard. (n.d.). Ziteboard Online Whiteboard. Retrieved from: https://ziteboard.com/ (Tool integrated for online whiteboard teaching sessions.)

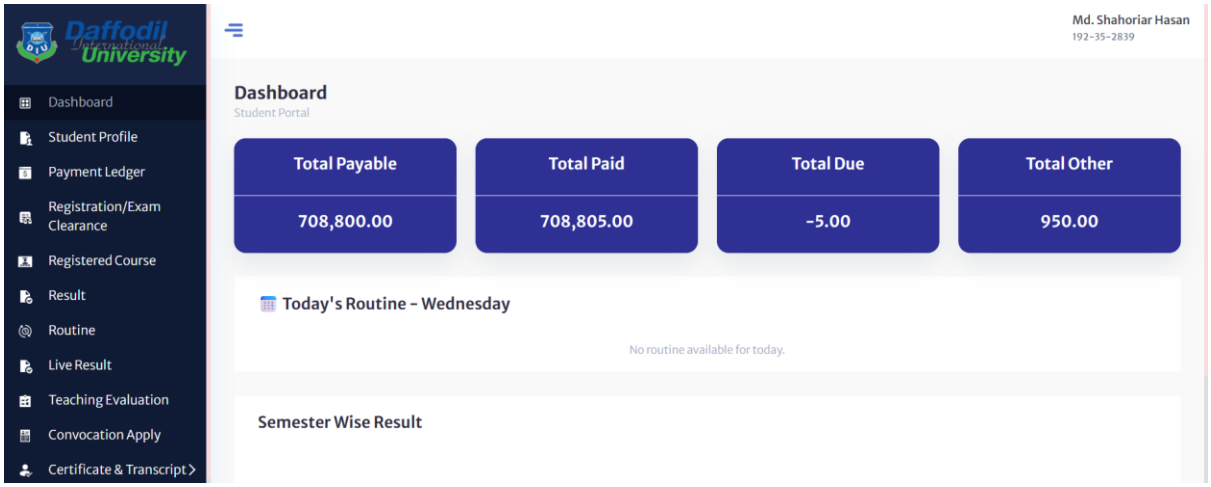
[8]. Agile Alliance. (2001). Manifesto for Agile Software Development. Retrieved from: https://agilemanifesto.org/ (Foundation for Agile methodology applied in the development process.)

[9]. OpenAI. (2025). ChatGPT. Retrieved from: https://chat.openai.com/ (Assisted in structuring project documentation and diagrams.)

APPENDIX A: Code Link

GitHub Link: https://github.com/Shahoriar-Rawnak/Tuition-Management-System

Account Clearance



Plagiarism Report

192-35-2839

ORIGINALITY REPORT

16%	13%	4%	13%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	6%
2	dspace.daffodilvarsity.edu.bd:8080 Internet Source	5%
3	www.coursehero.com Internet Source	1%
4	Submitted to American Intercontinental University Online Student Paper	1%
5	Submitted to NCC Education Student Paper	1%
6	www.postjobfree.com Internet Source	<1%
7	Submitted to University of Portsmouth Student Paper	<1%
8	Submitted to Coventry University Student Paper	<1%
9	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
10	www.arcjournals.org Internet Source	<1%
11	Submitted to Adventist University of Central Africa Student Paper	<1%

12	Submitted to University of Northumbria at Newcastle Student Paper	<1%
13	Submitted to Victoria University Student Paper	<1%
14	Submitted to University of Bedfordshire Student Paper	<1%
15	nainnovation.tdc.sap.com Internet Source	<1%
16	brainmass.com Internet Source	<1%
17	www.episup.com Internet Source	<1%