GravityCore
TUTUION CLASS MANAGEMENT SYSTEM

GROUP 11

Master of Information Technology (MIT)
Department of Industrial Management - Faculty of Science
University of Kelaniya
September 2025

TEAM MEMBERS

Name	Registration Number
H M D C Bandara	FGS/MIT/2024/056
K H A Ayesha	FGS/MIT/2024/057
A E Jayakodi	FGS/MIT/2024/014
Wasana PREC	FGS/MIT/2024/052

GitHub Repository: https://github.com/dhanushkacb/OOP_Gravity

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

GravityEdu is a leading private educational center providing tuition classes for Advanced Level (A/L) students. The institute, known for its focus on student success, has traditionally managed key operations, student data, teacher management, class scheduling, and payments using manual spreadsheets and paper-based processes.

These manual workflows have resulted in several critical challenges:

- Errors in data entry.
- Difficulty in accurately tracking outstanding payments.
- Inefficient generation of reports.
- Time-consuming and high-volume administrative work.

To address these issues and streamline operations, GravityCore, a Python-based Tuition Class Management System, has been developed.

1.2 PROBLEM STATEMENT

The educational center is currently struggling with disconnected and manual processes. Using spreadsheets for managing student enrollments, payments, and class schedules is time-consuming and error-prone. This lack of integration makes it difficult to gain a clear, real-time view of finances and overall operational status, which significantly slows down effective decision-making. Furthermore, the increasing administrative workload overwhelms staff with routine tasks, distracting them from their primary goal: supporting students and enhancing the learning experience.

1.3 OBJECTIVES

The primary objectives of the GravityCore project are:

- To design and develop a centralized, user-friendly management system that automates key processes, creating a single, integrated platform for all tuition center operations.

- To provide comprehensive features including student management, teacher management, class scheduling, payment tracking, and robust reporting capabilities.
- To reduce manual work and significantly improve data accuracy by automating repetitive processes and ensuring consistent, up-to-date information across all modules.

1.4 PROJECT SCOPE

The GravityCore system provides the following core functionalities:

User Authentication:

The system provides a secure login mechanism with role based access control to ensure that only authorized users can access the application. Different roles (Admin, Staff/User) are supported, giving each role access only to the features they need. For example, an Admin can manage all data and generate reports, while a regular staff member is not able to manage payment or generate reports. This role-based control keeps data secure and prevents unauthorized changes.

Teacher Management:

Allows administrators to create, view, edit, and maintain teacher profiles, including contact details and subject expertise ensuring records stay accurate and up to date.

Student Management

The student management module handles the entire student lifecycle. Administrators can register new students with their personal details, update their information when required, and track attendance to monitor participation. It also allows staff to manage tute (study material) distribution so that each student receives the right resources at the right time.

Class Management

Class management makes scheduling easy and organized. Classes can be categorized by type (Group or Hall) and purpose (Theory or Revision). Teachers, subjects, and time slots can be assigned, and changes to the timetable can be updated in real-time, reducing confusion and last-minute miscommunication.

Payment Management

This module keeps track of all student payments. Administrators can record monthly tuition fees, apply discounts or exemptions where applicable, and instantly see which students have unpaid fees. It also generates detailed outstanding payment reports, helping the institute follow up on pending payments efficiently.

Batch Processing

This saves time by allowing administrators to upload large volumes of student data or payment records using CSV files. Instead of entering each record one by one, staff can import data in one go, minimizing manual effort and reducing the chance of typing errors.

Reports

The reporting feature provides a clear overview of the center's operations and finances. Reports can show total income for a selected period, outstanding balances from students who haven't paid, and statistics on student registrations. These insights help management make informed decisions and plan for future growth.

CHAPTER 2: PROJECT MANAGEMENT & REQUIREMENTS

2.1 PROJECT MANAGEMENT

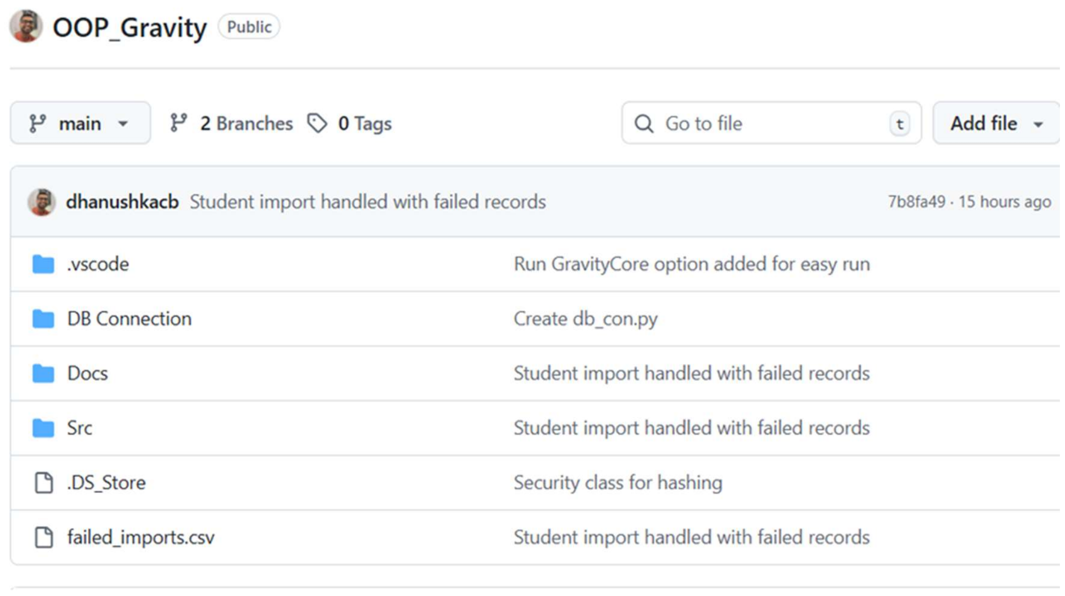
Effective project management was crucial to the success of the GravityCore system. Key aspects of the project management contribution are outlined below.

To ensure a transparent and efficient workflow, the project utilized **GitHub's project management features.**

- Tasks were organized into project boards representing specific development phases.
- Each individual task is assigned to team members, and tracked using GitHub's status indicators.

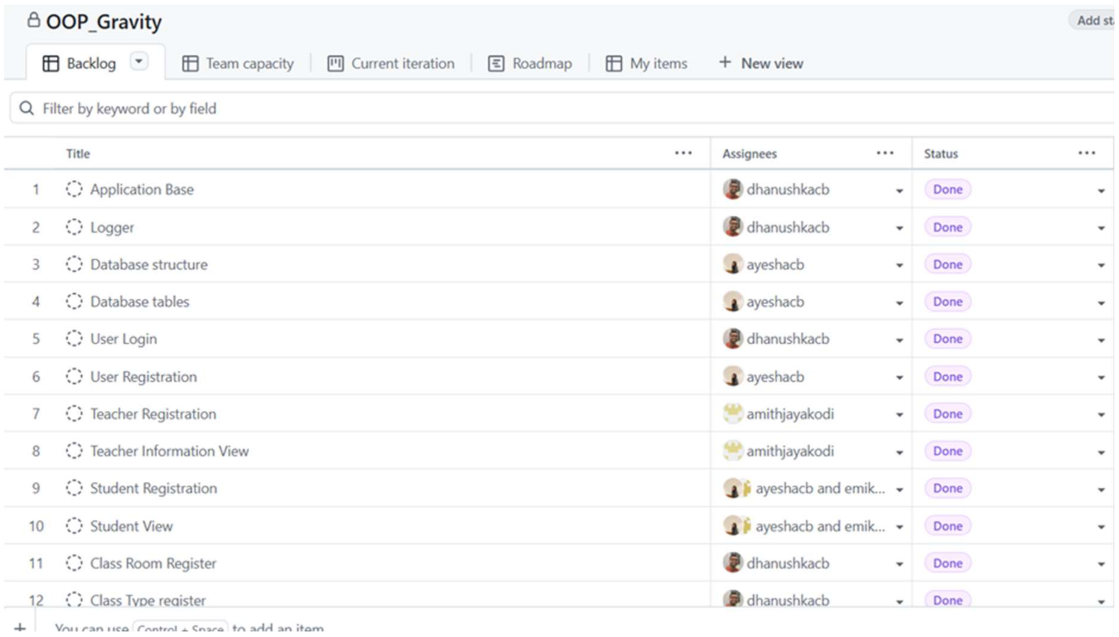
This approach ensured clear accountability and streamlined progress monitoring, allowing the team to address blockers quickly and stay up to date.

GitHub Repository: https://github.com/dhanushkacb/OOP_Gravity



dhanushkacb Student import handled with failed records 7b8fa49 · 15 hours ago

.vscode	Run GravityCore option added for easy run
DB Connection	Create db_con.py
Docs	Student import handled with failed records
Src	Student import handled with failed records
.DS_Store	Security class for hashing
failed_imports.csv	Student import handled with failed records



	Title	Assignees	Status
1	Application Base	dhanushkacb	Done
2	Logger	dhanushkacb	Done
3	Database structure	ayeshacb	Done
4	Database tables	ayeshacb	Done
5	User Login	dhanushkacb	Done
6	User Registration	ayeshacb	Done
7	Teacher Registration	amithjayakodi	Done
8	Teacher Information View	amithjayakodi	Done
9	Student Registration	ayeshacb and emik...	Done
10	Student View	ayeshacb and emik...	Done
11	Class Room Register	dhanushkacb	Done
12	Class Type register	dhanushkacb	Done

DEPLOYMENT STRATEGY

The system follows a structured deployment process:

- **Development Workflow:** Utilizes Version Control with feature branching, pull requests, and a code review process. The aim is to implement Continuous Integration (CI).
- **Deployment Environments:** Includes distinct Development and Production environments.
- **Database:** A plan is in place for database migrations and for Backup and Recovery to ensure data safety and availability.

2.2 FUNCTIONAL REQUIREMENTS

These define what the system must do to meet the user's needs:

1. **User/Admin Management:**
   - The system shall allow users (Admins and Staff) to log in with a username and password.
   - The system shall allow Admins to add, edit, and delete user accounts.
2. **Teacher Management:**
   - The system shall allow adding new teachers with necessary details.
   - The system shall allow viewing, editing, and deleting teacher records.
3. **Student Management:**
   - The system shall allow registering students with details (Name, Reg Year/Month, Contact, Email, Stream, Discount %).
   - The system shall allow updating or deleting student details.
   - The system shall allow bulk import of student details via CSV upload.
4. **Class & Attendance:**
   - The system shall allow scheduling classes with Subject, Teacher, Type (Group/Hall), Category (Theory/Revision), Time slot, and Classroom.
   - The system shall allow editing or deleting scheduled classes.
   - The system shall allow marking student attendance for each class session.
5. **Payment & Reporting:**
   - The system shall allow recording monthly tuition fee payments.
   - The system shall automatically apply discounts/exemptions when calculating payments.
   - The system shall provide functionality for generating reports (Total income, Outstanding balances, Student-wise payment history).

2.3 NON-FUNCTIONAL REQUIREMENTS

These define the quality attributes and constraints under which the system must operate:

- **Performance:** The system shall be capable of supporting concurrent users efficiently.
- **Scalability:** The system shall support multiple tier levels to accommodate growth and increased demand.
- **Security:** The system shall incorporate encrypted credentials (password hashing) and role-based access control (RBAC).
- **Usability:** The system shall feature a consistent and intuitive GUI using Tkinter and easy navigation.
- **Maintainability:** The system shall adopt a modular code design to facilitate smooth updates and enhancements.
- **Reliability:** The system shall aim for 99.9% uptime and perform daily backups to ensure data safety.

CHAPTER 3: SYSTEM DESIGN AND IMPLEMENTATION

3.1 TECHNOLOGY STACK

GravityCore is built using a robust, modern technology stack:

Component	**Technology**	**Description**
Frontend (View)	Tkinter (Python GUI)	Used to develop the desktop graphical user interface.
Backend (Controller)	Python	Implements the core application logic and business rules.
Database (Model)	MySQL	The relational database used for persistent data storage.

LIBRARIES USED

The core functionality is supported by the following libraries:

- **Tkinter/ttk:** Standard Python libraries for the GUI framework and modern themed widgets (Combobox, Treeview).
- **mysql-connector-python:** For database connectivity and executing SQL queries.
- **bcrypt** (or hashlib): Used for password hashing to secure user credentials.
- **CSV, os, datetime:** Standard libraries for bulk imports/exports, file handling, and timestamping.

3.2 SYSTEM ARCHITECTURE

The project follows a modular, layered architecture based on the Model-View-Controller (MVC) design pattern, ensuring a clear separation of concerns to improve maintainability, testability, and scalability.

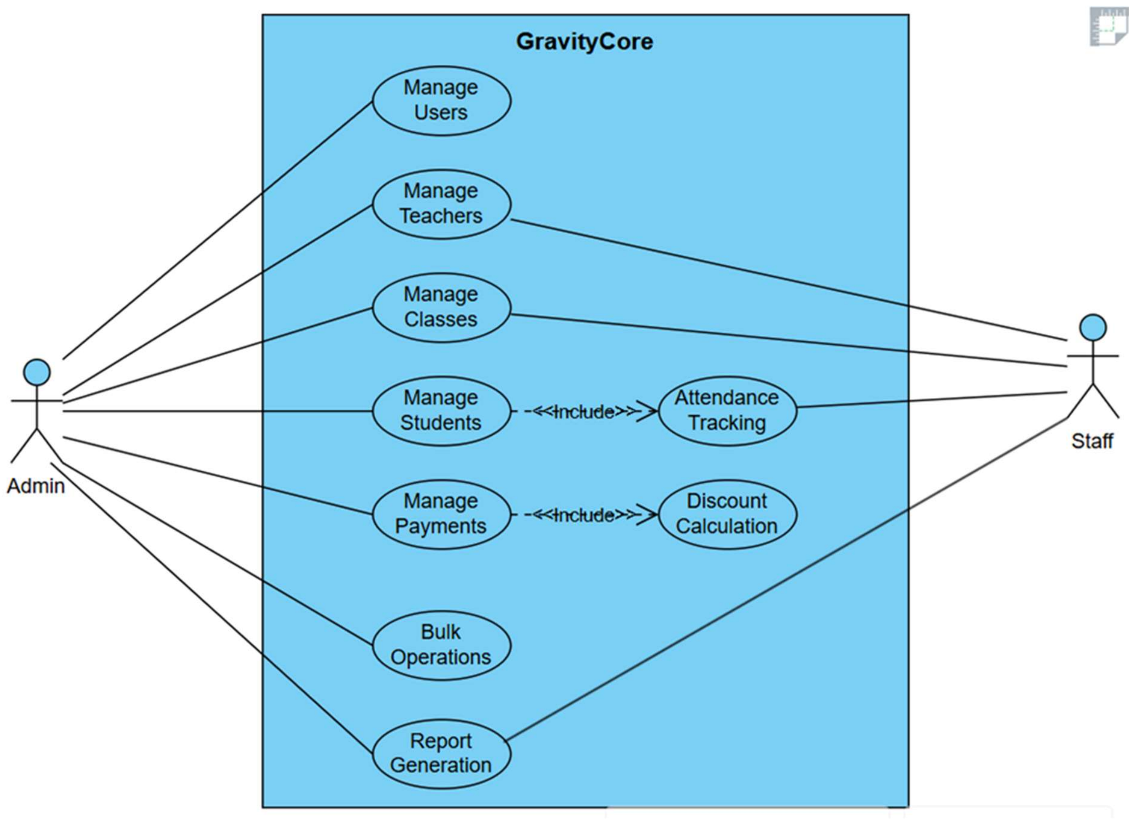
ARCHITECTURE LAYERS

1. **Model Layer (Data Access):**
   - **Purpose:** Handles all database operations, data access, and core business logic.
   - **Examples:** Classes like Users, Students, Attendance that encapsulate SQL queries and data manipulation (e.g., located in Schema.py, Connection.py).
2. **View Layer (User Interface):**
   - **Purpose:** Manages the user interface using Tkinter, displays forms, tables, and handles user input.
   - **Examples:** GUI classes such as UserRegistration.py, ClassRoomRegister.py, ImportStudentData.py.
3. **Controller/Process Layer (Application Logic):**
   - **Purpose:** Orchestrates the flow between the view and model, processes user actions, and coordinates data import/export.
   - **Examples:** Classes like BaseRegistration.py, StudentAttendanceProcess.py, and report generators.
4. **Utility/Config Layer:**
   - **Purpose:** Provides essential utilities for configuration, security (e.g., password hashing), and logging.
   - **Examples:** Settings.py, Security.py, Logger.py.

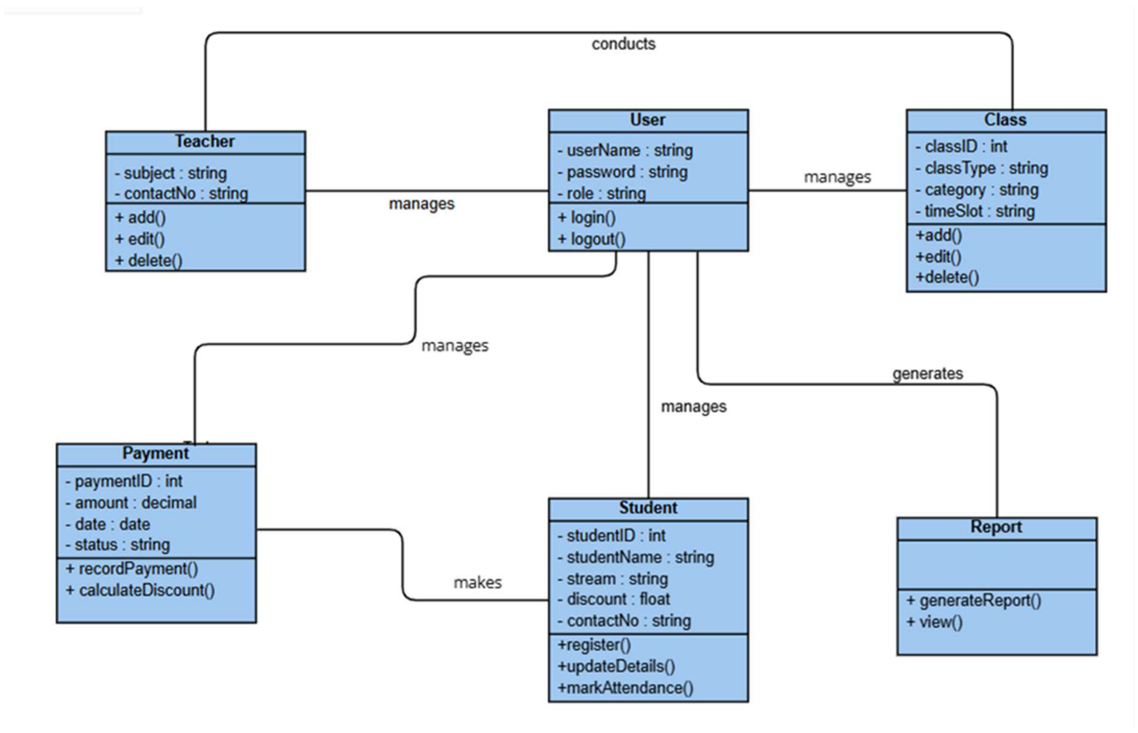
3.3 SYSTEM MODELS AND DIAGRAMS

These diagrams visually represent the static and dynamic structure of the system.

HIGH-LEVEL USE CASE DIAGRAM:



CLASS DIAGRAM:



3.4 INTERFACE DESIGN

The system is implemented as a **desktop GUI application** using Python Tkinter, following a form-based navigation model.

3.4.1 KEY INTERFACE ELEMENTS

Login Window: Features username and password entry fields with role-based authentication.

Main Dashboard: Provides a clear menu or navigation panel for accessing modules (Users, Teachers, Students, Classes, Payments, Reports).

Data Display: Utilizes ttk.Treeview tables for viewing lists of records and includes columns for Edit/Delete actions.

Forms: Uses standard Tkinter Entry widgets and ttk.Combobox for data input and selections.

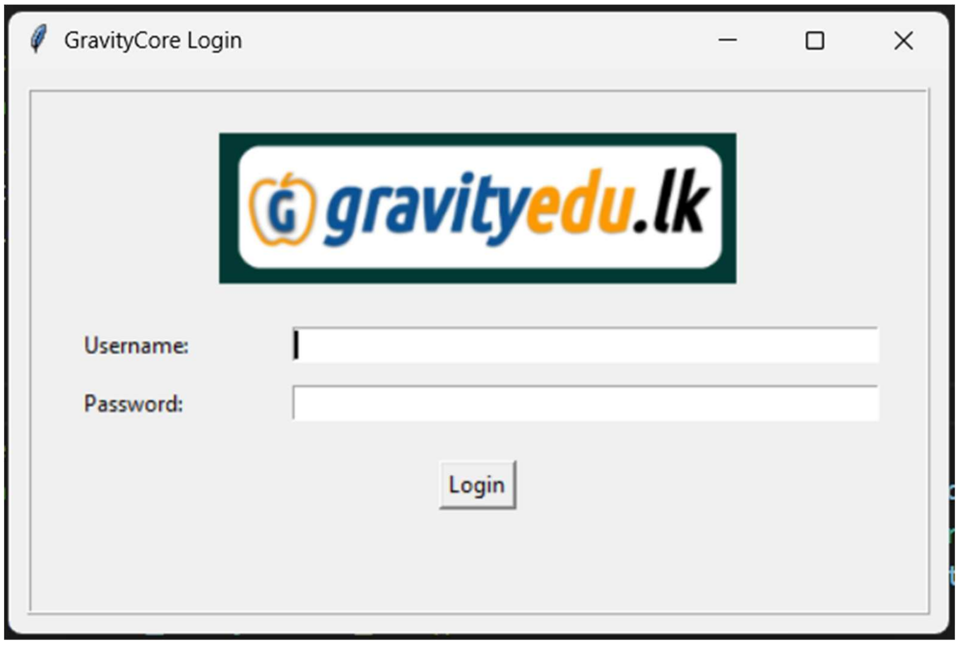
Dialogs: Employs messagebox for confirmations, errors, and success messages.

Reports Interface: Features Treeviews for data display, filters (month, year, teacher), and export options (CSV/TXT).

3.4.2 SCREEN LAYOUTS

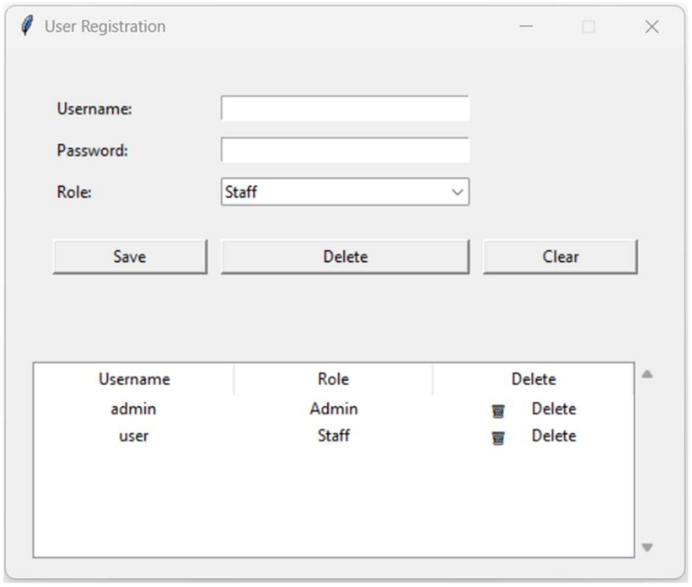
1. User Login

Provides fields for username and password input, with role-based authentication. Incorrect login attempts trigger an error message.



2. User Registration

Allows Admins to create, edit, and delete system user accounts. Includes fields for username, password, and role selection (Admin/Staff).



Username:

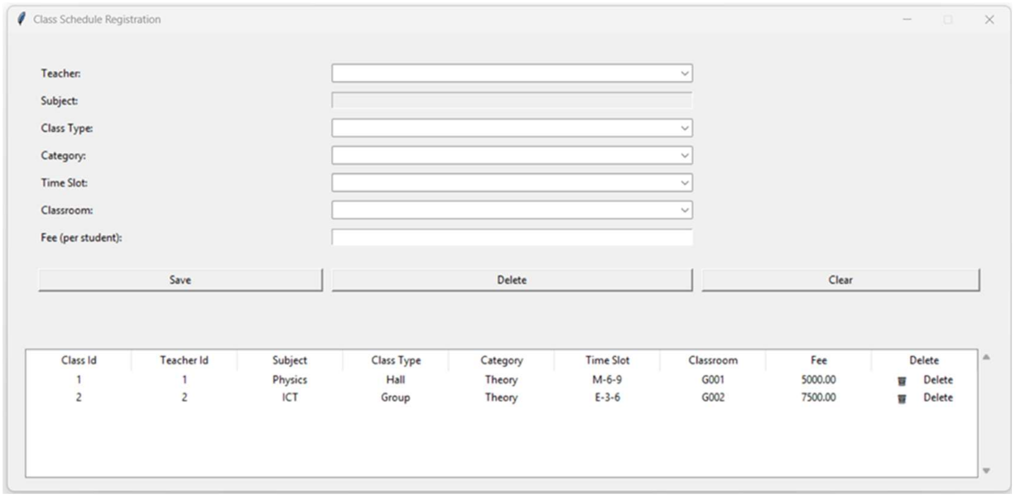
Password:

Role: Staff

Username	Role	Delete
admin	Admin	Delete
user	Staff	Delete

3. **Class Scheduling**

   Allows scheduling of classes with fields for Subject, Teacher, Category (Theory / Revision), Type (Group/Hall), Date/Time, and Classroom. Detects and warns about schedule conflicts.



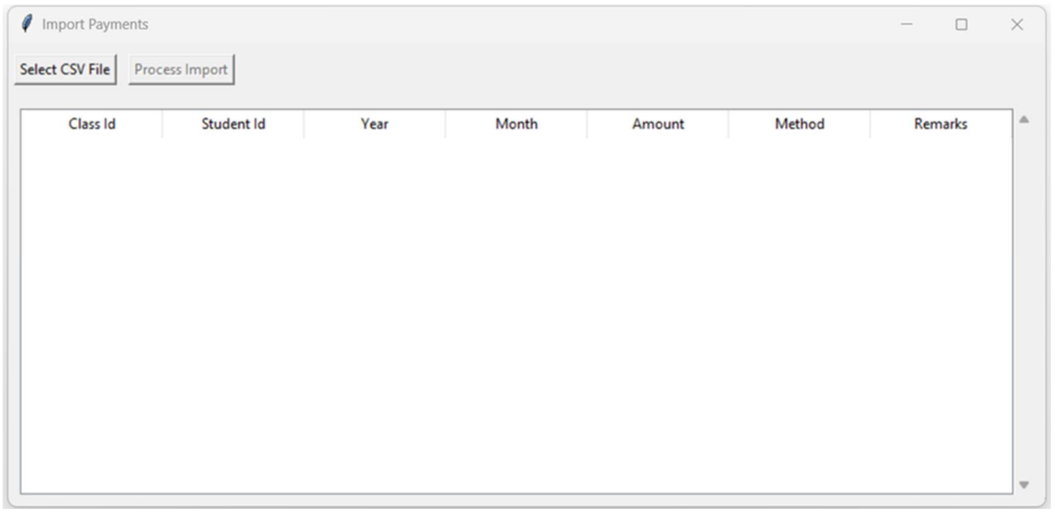
Class Schedule Registration

Teacher:
Subject:
Class Type:
Category:
Time Slot:
Classroom:
Fee (per student):

Class Id	Teacher Id	Subject	Class Type	Category	Time Slot	Classroom	Fee	Delete
1	1	Physics	Hall	Theory	M-6-9	G001	5000.00	Delete
2	2	ICT	Group	Theory	E-3-6	G002	7500.00	Delete

4. **CSV Upload and Process**

   Provides a file upload option to import student data or payment records from CSV files. Displays a summary of imported records and any errors encountered. Allows the user to process the files.

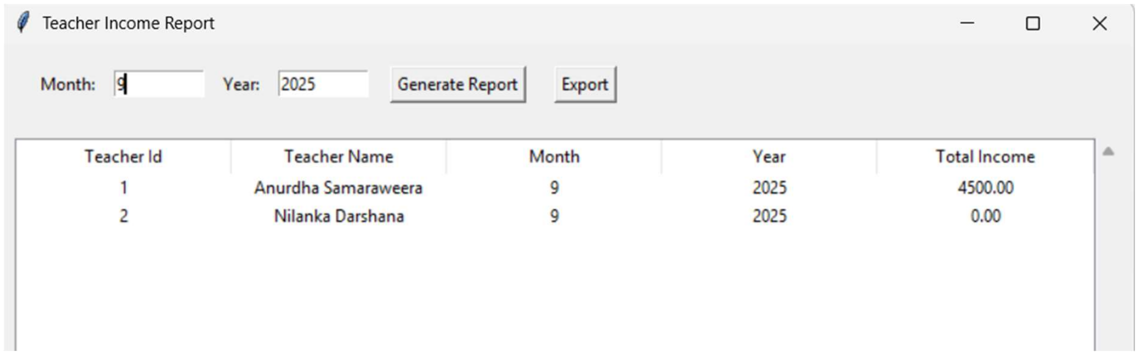


Import Payments

Class Id	Student Id	Year	Month	Amount	Method	Remarks

5. **Report Generation**

Generates reports such as total income, outstanding payments, and student-wise payment history. Provides filtering options (month/year) and export as TXT files.

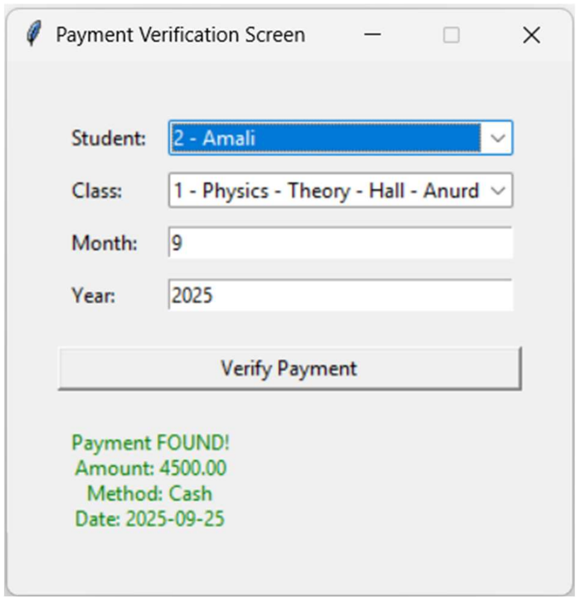


Teacher Income Report

Month: 9 Year: 2025 Generate Report Export

Teacher Id	Teacher Name	Month	Year	Total Income
1	Anurdha Samaraweera	9	2025	4500.00
2	Nilanka Darshana	9	2025	0.00

6. **Payment Verification**



Payment Verification Screen

Student: 2 - Amali

Class: 1 - Physics - Theory - Hall - Anurd

Month: 9

Year: 2025

Verify Payment

Payment FOUND!
Amount: 4500.00
Method: Cash
Date: 2025-09-25

3.5 SYSTEM TESTING

System testing was performed to ensure the system meets all specified functional requirements. Below are key test scenarios:

Test Case ID	Module	Test Scenario	Expected Result
TC-01	User Authentication	Login with valid admin credentials.	Admin is successfully logged in and redirected to the main window.
TC-02	User Authentication	Login with invalid credentials.	The system displays an error message and prevents login.
TC-03	Teacher Management	Add a new teacher.	A new teacher is added and visible in the teacher list.
TC-04	Student Management	Add a new student.	A student is added and visible in the student list.
TC-05	Class Management	Schedule a new class.	Class is scheduled and appears in the timetable without conflicts.
TC-06	Payment Management	Record student fee.	Payment is recorded and balance updated correctly.
TC-07	Batch Processing	CSV upload for students' attendance.	The system successfully imports student records and saves them in the database.
TC-08	Reports	Generate outstanding payments report.	The report is displayed correctly with accurate outstanding balances.

CHAPTER 4: CONCLUSION

GravityCore represents a significant step forward in modernizing the tuition center's operations. By successfully digitizing and centralizing key processes—including student registration, class scheduling, payment tracking, and reporting—the system effectively eliminates the inefficiencies and errors inherent in the previous manual, paper-based workflows.

The system's user-friendly Tkinter interface ensures a quick staff adaptation, while the underlying MVC architecture provides a solid, maintainable foundation for scalability and future feature expansion. The automation of routine tasks dramatically reduces administrative workload and ensures that critical information is consistently accurate and accessible.

Crucially, the powerful reporting capabilities offer management real-time insight into income and outstanding payments, enabling faster and better-informed decision-making.

Ultimately, GravityCore allows the GravityEdu team to shift their focus from time-consuming paperwork toward their core mission: delivering high-quality education and supporting student success. This system positions the institute for sustainable growth, improved operational efficiency, and an enhanced experience for all stakeholders.

CHAPTER 5: FUTURE ENHANCEMENTS

The following features are identified as potential improvements for future versions of GravityCore:

- **QR-Based Attendance Marking:**
  Replace manual attendance marking with QR code scanning for faster, contactless attendance tracking.

- **SMS/Email Notifications:**
  Automatically notify students/parents about outstanding payments, class reschedules, or important announcements.

- **Student/Parent Portal:**
  Provide a secure web portal where students and parents can view attendance history, payment records, and class schedules.

CHAPTER 6: LEARNINGS

Strengthen technical skills:

During the development of **GravityCore**, we strengthened several key technical skills:

- **Layered Architecture:** Learned to separate concerns effectively, building modular, maintainable, and scalable applications by organizing code into Models (data handling), Views (Tkinter UI), and Controllers (logic flow).

- **Database Design & MySQL Integration:** Designed ER diagrams, normalized tables, implemented relationships, and wrote SQL queries (CRUD, joins, reports) to manage student, teacher, class, and payment data efficiently.

- **GUI Development with Tkinter:** Built intuitive, form-based interfaces with Treeviews, entry fields, and navigation panels, improving user experience and reducing errors.

Problem-Solving:

Improved analytical skills by troubleshooting database connection issues, UI glitches, and logic errors during development.

Time Management:

Learned to plan sprints, set realistic deadlines, and deliver features on time.

Communication & Documentation:

Enhanced the ability to write clear technical documentation and communicate progress to stakeholders.