****

**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** |

# Table of Contents

[Team Members 2](#_Toc209728283)

[Table of Contents 3](#_Toc209728284)

[Chapter 1: INTRODUCTION 5](#_Toc209728285)

[1.1 background 5](#_Toc209728286)

[1.2 Problem Statement 5](#_Toc209728287)

[1.3 Objectives 5](#_Toc209728288)

[1.4 Project Scope 7](#_Toc209728289)

[Chapter 2: SYstem design 9](#_Toc209728290)

[2.1 system architecture 9](#_Toc209728291)

[2.2 High-Level Use Case Diagram 10](#_Toc209728292)

[2.3 Class diagram 10](#_Toc209728293)

[2.4 project management 11](#_Toc209728294)

[2.4.1 Backlog 12](#_Toc209728295)

[2.5 Functional Requirements 12](#_Toc209728296)

[2.6 Non-Functional Requirements 13](#_Toc209728297)

[3. Implementation 14](#_Toc209728298)

[3.1 Technology Stack 14](#_Toc209728299)

[3.2 iNTERFACE Design 14](#_Toc209728300)

[2.11.1 Key Interface Elements 14](#_Toc209728301)

[2.11.2 Libraries Used 15](#_Toc209728302)

[2.11.3 Screen Layouts 15](#_Toc209728303)

[2.9 Testing 18](#_Toc209728304)

[2.10 Deployment Strategy 20](#_Toc209728305)

[Chapter 4: Conclusion 21](#_Toc209728306)

[List of Figures 21](#_Toc209728307)

# Chapter 1: INTRODUCTION

## 1.1 background

GravityEdu is a leading private educational center that provides Advanced Level (A/L) tuition to students. The institute is known for its quality teaching, focus on student success, and dedication to excellence. However, the management of students, teachers, class schedules, and payments has traditionally been handled manually using spreadsheets and paper-based processes. These manual workflows have led to challenges such as:

* Errors in data entry
* Difficulty in tracking outstanding payments
* Inefficient reporting
* Time-consuming administrative work

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

## 1.2 Problem Statement

Right now, the educational center struggles with disconnected and manual processes. Managing student enrollments, payments, and class schedules using spreadsheets is time-consuming and error-prone. It’s hard to get a clear view of finances and overall operations, which slows down decision-making. On top of that, the administrative workload keeps growing, leaving staff overwhelmed with routine tasks instead of focusing on what really matters-supporting students and improving the learning experience.

## 1.3 Objectives

* To design and develop a centralized, user-friendly management system that automates key processes.

The primary objective of this project is to build a single, integrated platform where all tuition center operations can be managed efficiently. By centralizing data and processes into one system, administrators will no longer need to switch between multiple spreadsheets or paper-based records. The user interface will be intuitive and accessible, making it easy for non-technical staff to navigate and perform day-to-day tasks. Automation will handle repetitive processes such as fee calculations, payment verification, and report generation, freeing staff from manual workload and minimizing errors.

* To provide features like student management, teacher management, class scheduling, payment tracking, and reporting.

GravityCore is designed to be a comprehensive management solution that covers all major aspects of running a tuition center:

* Student Management: Register students, update their information, and track attendance and tute distribution.
* Teacher Management: Maintain teacher profiles with subject expertise and contact details.
* Class Scheduling: Organize classes by category (Theory/Revision), type (Group/Hall), time slots, and teacher assignments.
* Payment Tracking: Record monthly tuition fees, manage discounts, and monitor outstanding balances.
* Reporting: Generate income reports, outstanding payment summaries, and registration statistics to support informed decision-making.
* To reduce manual work and improve data accuracy.

Manual data entry and paper-based records often lead to duplication, data loss, and calculation mistakes. By automating processes, GravityCore ensures consistent, accurate, and up-to-date information across all modules. Staff effort is reduced significantly, allowing them to focus on more meaningful tasks such as improving student experience and academic outcomes. Accurate data also facilitates faster decision-making and better financial tracking.

## 1.4 Project Scope

GravityCore provides:

* **User Authentication**

The system provides a secure login mechanism 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 might only be able to update attendance or record payments. This role-based control keeps data secure and prevents unauthorized changes.

* **Teacher Management**

This feature allows the administrator to create and maintain teacher profiles in the system. Each profile includes details like the teacher’s name, subject, contact number, and email address. If a teacher’s information changes (e.g., contact number), it can be quickly updated, 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.

* **Bulk Operations**

Bulk operations save 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: SYstem design

## 2.1 system architecture

Project follows a modular, layered architecture with clear separation between:

* Data access (Model)
* User interface (View)
* Application logic/processes (Controller)
* Utilities (Config, Security, Logging)

This structure improves maintainability, testability, and scalability of the application.

**1. Model Layer**

* Located in: Schema.py, Connection.py, Configuration.py
* Purpose: Handles all database operations, data access, and business logic.
* Example: Classes like Users, Students, Attendance etc., encapsulate SQL queries and data manipulation.

**2. View Layer**

* Located in GUI classes such as UserRegistration.py, ClassRoomRegister.py, ImportStudentData.py etc
* Purpose: Manages the user interface using Tkinter, displays forms, tables, dialogs, and handles user input.

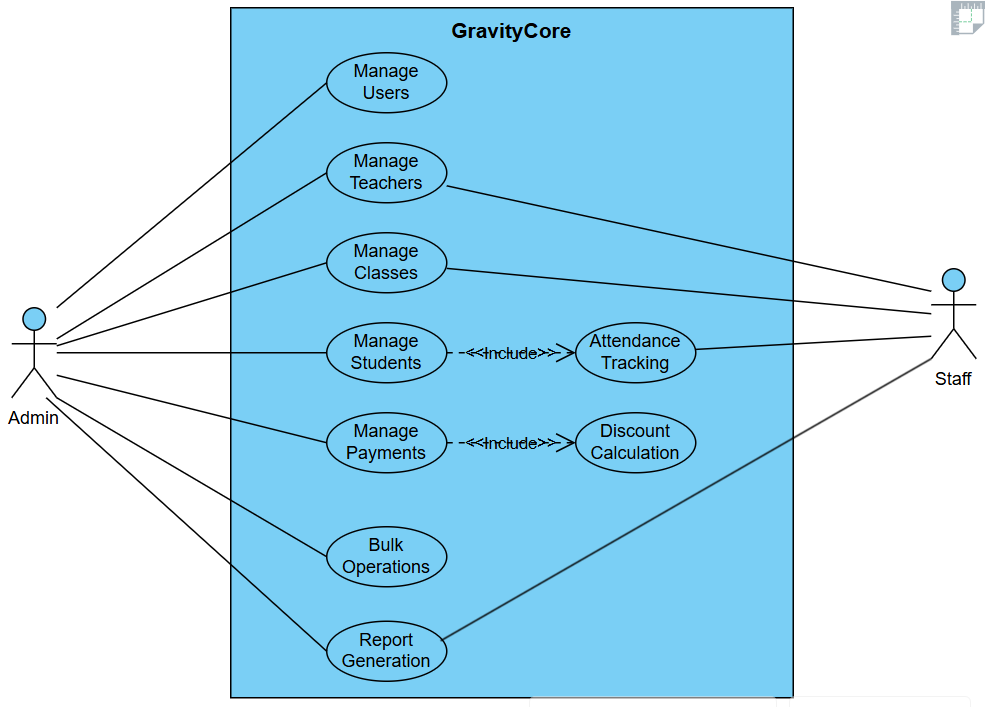
**3. Controller/Process Layer**

* Located in: Classes like BaseRegistration, StudentAttendanceProcess, and report generators.
* Purpose: Orchestrates the flow between the view and model, processes user actions, and coordinates data import/export.

**4. Utility/Config Layer**

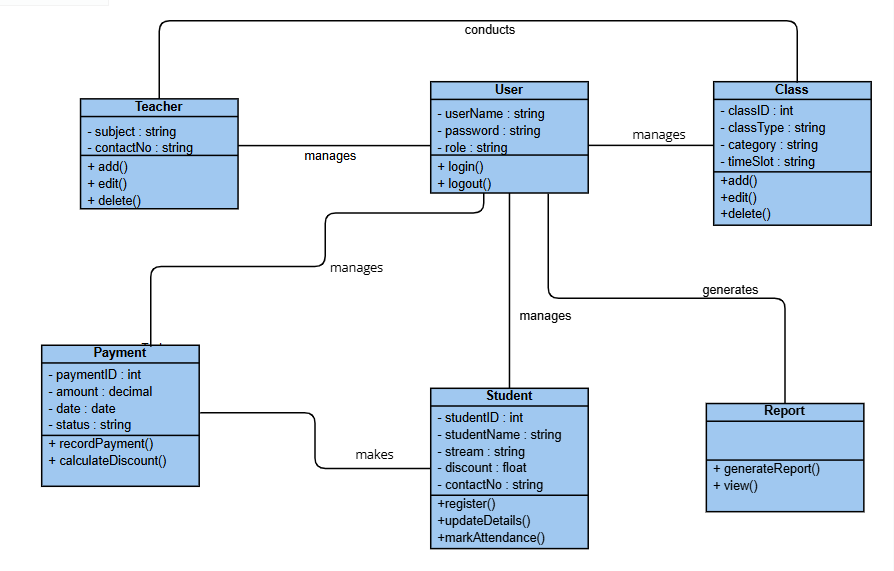
* Located in: Settings.py, Security.py, Logger.py
* Purpose: Provides configuration, security (e.g., password hashing), and logging utilities.

## 2.2 High-Level Use Case Diagram

****

## 2.3 Class diagram

The class diagrams presented here show the static structure of the GravityCore system, illustrating the relationships between different classes and their attributes. These diagrams provide a detailed view of the system's object-oriented design, showing how different entities like students, teachers, classes, and users are related to each other. The diagrams help understand the system's data model and the inheritance relationships between different classes. They serve as a blueprint for the system's implementation, showing how different components interact at the code level.



## 2.4 project management

Effective project management was crucial to the success of the GravityCore system. Key

aspects of the project management contribution are outlined below.

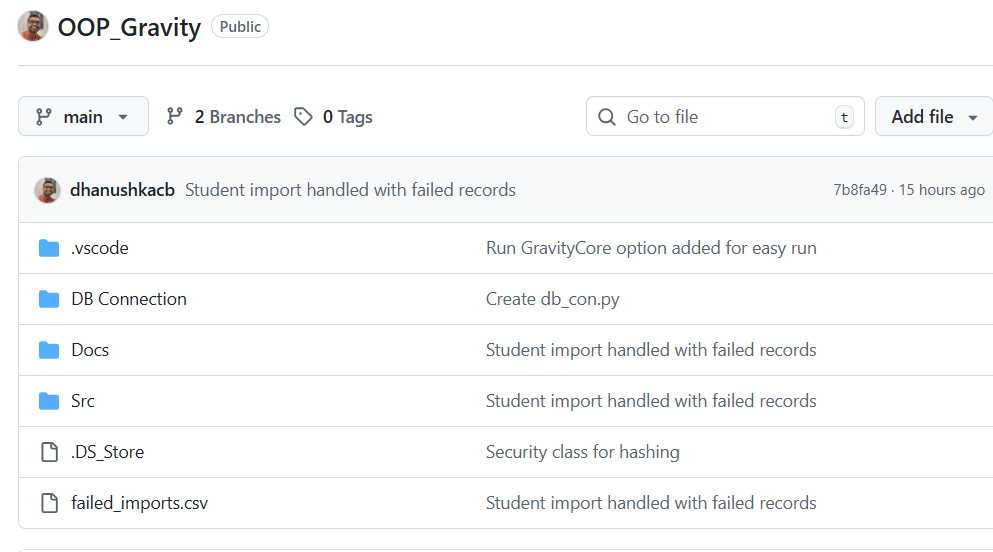


Figure 1: Git Hub Repository

### 2.4.1 Backlog

To facilitate transparent and efficient workflow tracking, we utilized GitHub’s project

management features. Tasks were organized into project boards, with each board representing a specific development phase. Issues were created for individual tasks, assigned to relevant team members, and tracked using GitHub’s timeline and status indicators. This system allowed the entire team to stay updated on progress, address bottlenecks quickly, and

maintain clear accountability.

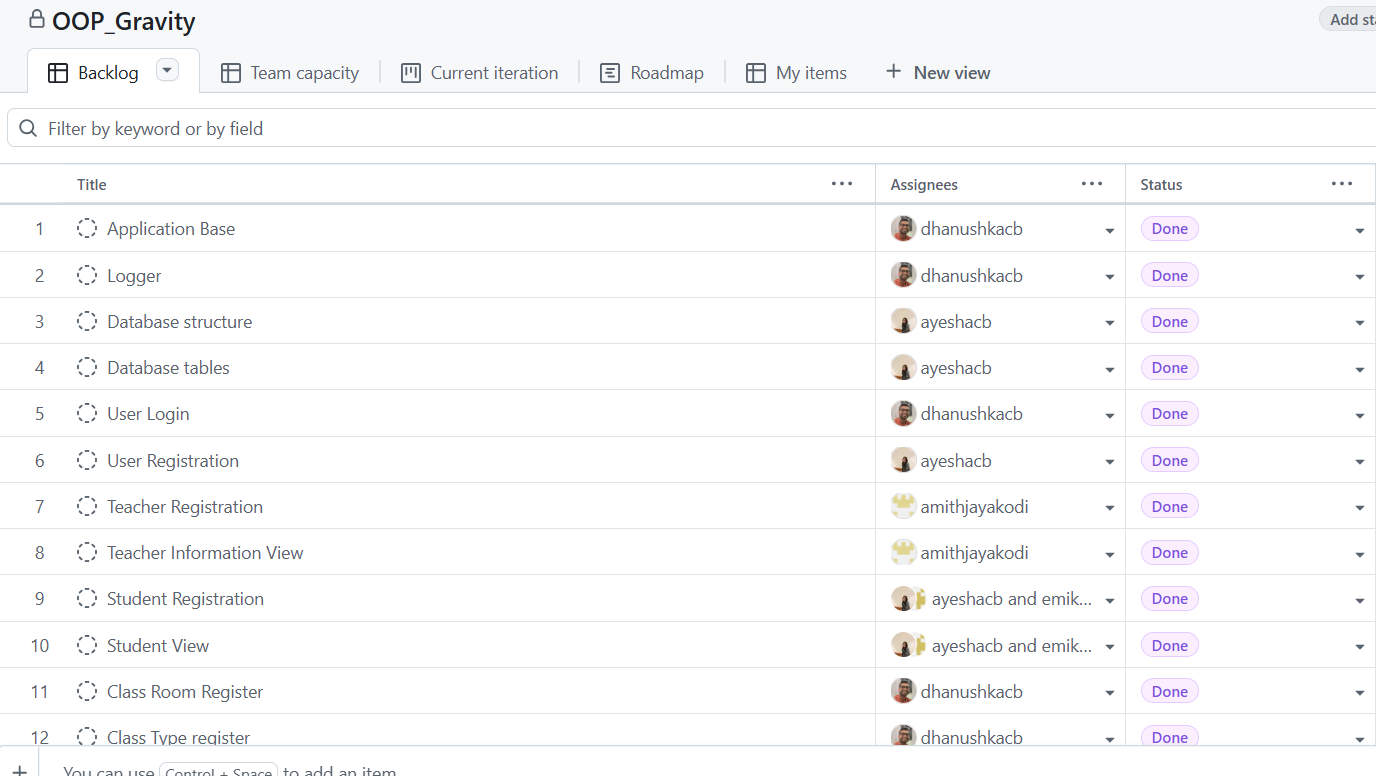


Figure 2: Backlog

2.5 Functional Requirements

* 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**.
* The system shall allow adding new teachers with details
* The system shall allow viewing, editing, and deleting teacher records.
* The system shall allow registering students with details (Name, Registration Year & Month, Contact Number, Email, Stream, Discount %).
* The system shall allow updating or deleting student details.
* The system shall allow bulk import of student details via **CSV upload**.
* The system shall allow scheduling classes with Subject, Teacher, Class type (Group/Hall), Category (Theory/Revision), Time slot, Classroom
* The system shall allow editing or deleting scheduled classes.
* The system shall allow recording **monthly tuition fee payments** for students.
* The system shall apply **discounts/exemptions** when calculating payment.
* The system **shall** provide functionality for generating reports:
* Total income
* Outstanding balances
* Student-wise payment history
* The system shall allow marking student attendance for each class session.

2.6 Non-Functional Requirements

* **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, role-based access control.
* **Usability**: The system **shall** feature a **consistent and intuitive GUI** using Tkinter and easy navigation.
* **Maintainability**: The system **shall** adopt modular code design to facilitate smooth updates and enhancements.
* **Reliability**: The system **shall** achieve 99.9% uptime and perform daily backups to ensure data safety and availability.

# 3. Implementation

## 3.1 Technology Stack

* **Frontend (View):** Tkinter (Python GUI)
* **Backend (Controller):** Python classes and business logic
* **Database (Model):** MySQL

## 

## 3.2 iNTERFACE Design

The system is designed as a desktop GUI application using Python Tkinter. The interface follows a form-based navigation model, where each module (User Management, Teacher, Student, Class Scheduling, Payments, Reports) is displayed in its own Toplevel window.

### 2.11.1 Key Interface Elements

 **Login Window**

* Username, password entry fields.
* Role-based authentication (Admin, Staff).

 **Main Dashboard**

* Menu bar or navigation panel with options: Users, Teachers, Students, Classes, Payments, Reports.

 **Forms**

* Standard Tkinter Entry widgets for text input.
* ttk.Combobox for dropdown selections (e.g., Role, Class Type, Category).
* Button widgets for actions (Register, Save, Delete).

 **Data Display**

* ttk.Treeview tables for showing lists of users, teachers, students, payments.
* Columns for Edit/Delete actions.

 **Dialogs**

* messagebox for confirmations, errors, and success messages.
* Pop-up Toplevel windows for editing existing records.

 **Reports Interface**

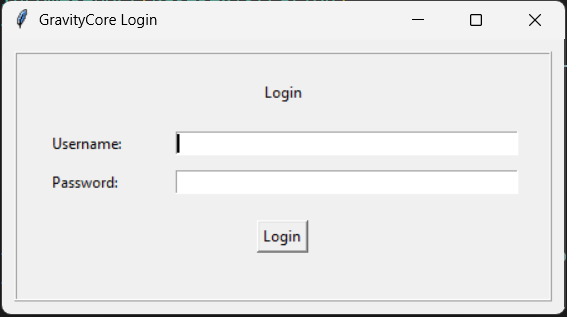
* Treeviews and export options (CSV/TXT).
* Filters for generating reports (month, year, teacher, stream).

### 2.11.2 Libraries Used

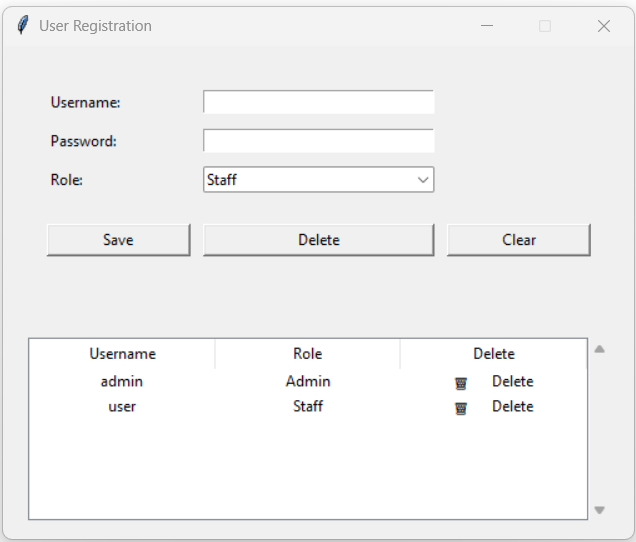
1. **Tkinter (Standard Python Library)**
   * Core GUI framework.
   * Tk, Toplevel, Frame, Label, Entry, Button, messagebox.
2. **Tkinter.ttk (Themed Tkinter Widgets)**
   * Provides modern widgets (Combobox, Treeview, styled buttons).
   * Used for dropdowns and tables.
3. **mysql-connector-python**
   * Database connectivity library.
   * Used for executing SQL queries, inserting/updating/fetching records.
4. **bcrypt (or hashlib)**
   * Used for **password hashing** to ensure secure storage.
5. **CSV (Standard Python Library)** *(optional, for bulk imports/exports)*
   * Reads/writes CSV files for bulk student/payment uploads.
6. **os / datetime (Standard Python Library)**
   * For file handling, reports export, and timestamping logs.

### 2.11.3 Screen Layouts

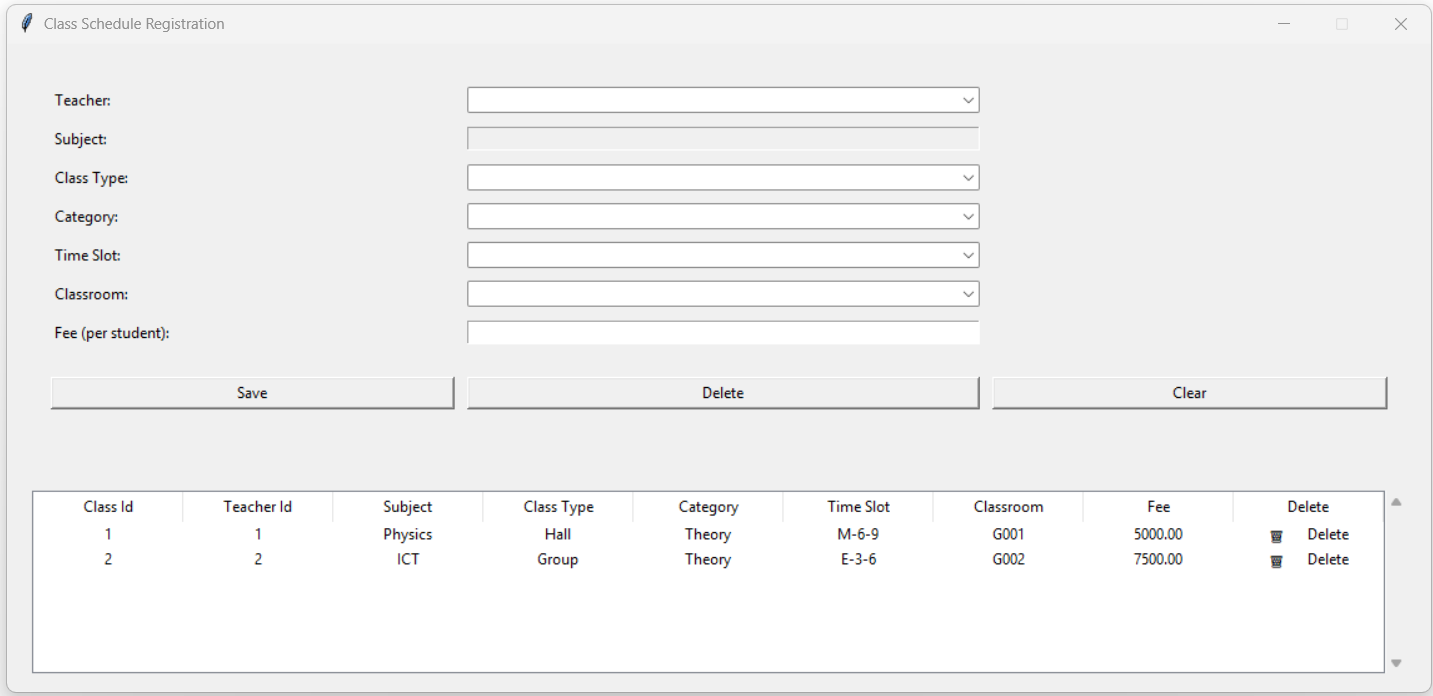
1. User Login



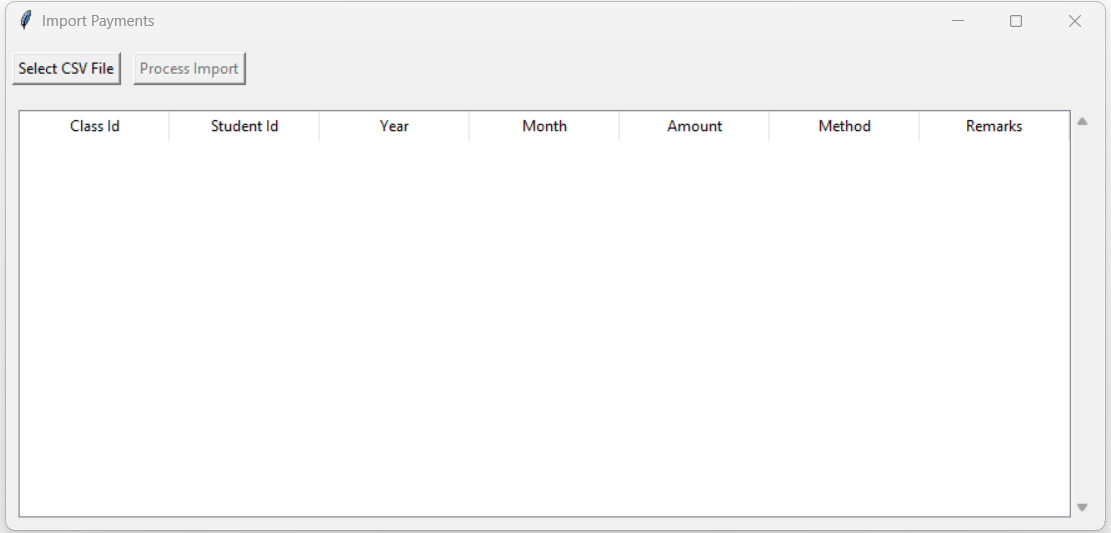
1. User Registration



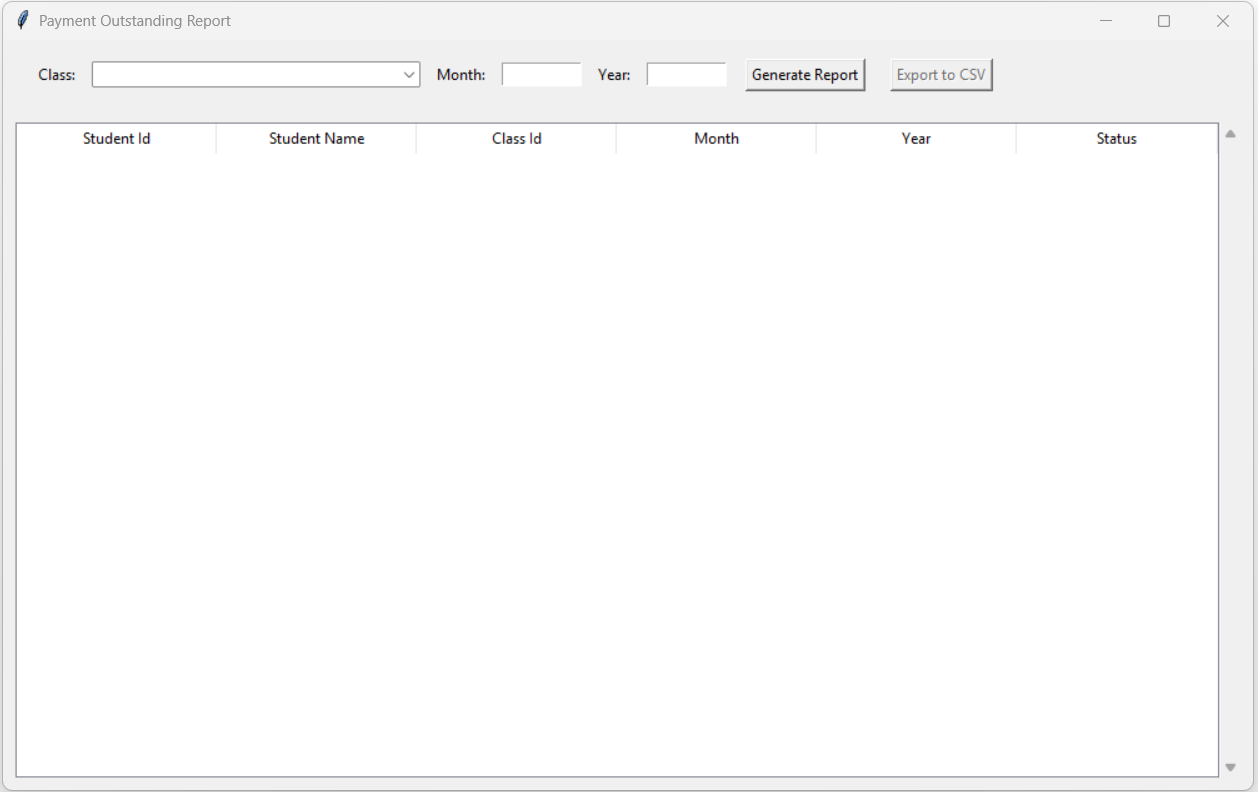
1. Class Scheduling



1. Bulk Upload



1. Reports



## 2.9 Testing

| **Test Case ID** | **Module** | **Test Scenario** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| UAT-01 | User Authentication | Login with valid admin credentials | 1. Open login page2. Enter valid admin username and password3. Click Login | Admin is successfully logged in and redirected to the dashboard |
| UAT-02 | User Authentication | Login with invalid credentials | 1. Enter incorrect username or password2. Click Login | System displays an error message and prevents login |
| UAT-03 | Teacher Management | Add a new teacher | 1. Navigate to Teacher Management2. Click Add Teacher3. Enter Name, Subject, Contact No, Email4. Click Save | New teacher is added and visible in the teacher list |
| UAT-04 | Teacher Management | Edit teacher details | 1. Select a teacher from the list2. Click Edit3. Update details4. Save changes | Teacher information is updated correctly |
| UAT-05 | Student Management | Add a new student | 1. Navigate to Student Management2. Click Add Student3. Enter Name, Registration Year & Month, Contact No, Discount %, Email, Stream4. Save | Student is added and visible in the student list |
| UAT-06 | Student Management | Edit student information | 1. Select a student2. Click Edit3. Update details4. Save | Student information is updated successfully |
| UAT-07 | Class Management | Schedule a new class | 1. Navigate to Class Management2. Enter Classroom, Class Type, Category, Teacher, Subject, Time Slot3. Save | Class is scheduled and appears in the timetable without conflicts |
| UAT-08 | Payment Management | Record student fee | 1. Go to Payment Management2. Select student3. Enter payment details and discount if any4. Save | Payment is recorded and balance updated correctly |
| UAT-09 | Attendance Tracking | Mark attendance | 1. Go to Student and Tute Tracking2. Select class session3. Mark present/absent for each student4. Save | Attendance is recorded and updated for the session |
| UAT-10 | Bulk Operations | CSV upload for students | 1. Go to Bulk Operations2. Upload CSV containing multiple student records3. Submit | System successfully imports student records and displays them in the list |
| UAT-11 | Reports | Generate outstanding payments report | 1. Navigate to Reports2. Select Outstanding Payments3. Generate report | Report is displayed correctly with accurate outstanding balances |

## 2.10 Deployment Strategy

**Development Workflow**

1. Version Control

* Feature branching
* Pull requests
* Code review process
* Continuous integration

2. Deployment Pipeline

* Database migrations
* Environment configuration

**Deployment Environments**

* Development
* Production
* Backup and recovery

This technology stack and design approach ensures a robust, scalable, and maintainable incident management system that meets modern development standards and best practices.

# Chapter 4: Conclusion

GravityCore represents a significant step forward in modernizing tuition center operations. By digitizing and centralizing key processes such as student registration, class scheduling, payment tracking, and reporting, the system eliminates the inefficiencies and errors that often arise from manual record-keeping. Its user-friendly interface ensures that staff members can quickly adapt to the system, while the underlying MVC architecture provides a solid foundation for scalability and future feature expansion.

The automation of routine tasks not only reduces administrative workload but also ensures that critical information is always accurate, up to date, and easily accessible. With powerful reporting capabilities, management gains real-time insight into income, outstanding payments, and student performance, enabling faster and better-informed decision-making.

Most importantly, GravityCore allows the GravityEdu team to shift their focus away from time-consuming paperwork and toward what truly matters — delivering high-quality education and supporting student success. This system positions the institute for sustainable growth, improved operational efficiency, and a better experience for students, parents, and educators in an increasingly digital world.

# List of Figures

[Figure 1: Git Hub Repository 11](#_Toc209728907)

[Figure 2: Backlog 12](#_Toc209728908)