SAVITRIBAI PHULE PUNE UNIVERSITY

A MINI PROJECT REPORT ON

Students Management System

Submitted by

Name : Pratik Vijay Manjarekar **Roll no : 57**

CLASS: TE **DIV: A**

Under the Guidance of

Guide Name: Mr. Tanmay Pawar

Sinhgad Institutes

DEPARTMENT OF COMPUTER ENGINEERING

RMD SINHGAD SCHOOL OF ENGINEERING

WARJE, PUNE 411058

2024 - 2025

Sinhgad Institutes

DEPARTMENT OF COMPUTER ENGINEERING

RMD SINHGAD SCHOOL OF ENGINEERING

WARJE, PUNE 411058

CERTIFICATE

This is to certify that the project report entitles

"Students Management System"

Submitted by

Name: Pratik Vijay Manjarekar PRN No :72262586D

is a bonafide work carried out by them under the supervision of Mr. Tanmay Pawar. And it is submitted towards the partial fulfillment of the requirement of Savitribai Phule Pune University for Third Year.

(Mr. Tanmay Pawar**)**
Guide
Department of Computer Engineering

(Dr. Deepali Newaskar)
Head,
Department of Computer Engineering

(Dr. V. V. Dixit)
Principal,
RMD Sinhgad School of Engineering Pune – 58

Place: Pune Date:

II

Certificate by Guide

This is to certify that Mr. Pratik Vijay Manjarekar has completed the MINI Project work under my guidance and supervision and that, I have verified the work for its originality in documentation, problem statement, implementation and results presented in the Project.

Place: Pune Date:

Signature of Guide
(Mr. Tanmay Pawar**)**

ACKNOWLEDGEMENT

It is our pleasure to acknowledge sense of gratitude to all those who helped us in making this Mini Project. We thank our Mini Project Guide Mr. Tanmay Pawar for helping us and providing all necessary information regarding our project.

We are also thankful to **Dr. Deepali Newaskar (Head - Department of Computer Engineering)** for providing us the required facilities and helping us while carrying out this work.

Finally we wish to thank all our teachers and friends for their constructive comments, suggestions and criticism and all those directly or indirectly helped us in completing this Mini Project.

NAME OF THE STUDENT

Pratik Vijay Manjarekar

INDEX

1. Introduction

1.1 Objectives

The main objective of the Student Management System (SMS) is to provide a digital platform to manage and maintain student data efficiently within an educational institution. The system is designed to:

- Allow admin users to add, update, and delete student records.
- Manage academic and personal details of students.
- Enable quick access to student information when required.

1.2 Problem Statement

Managing student information manually using paper-based methods or spread Sheets is error-prone and inefficient, especially for institutions handling large volumes of data.These traditional methods lack data integrity, quick retrieval, and security. The proposed system aims to overcome these issues by digitizing student records, enabling accurate storage, easy management, and real-time access to student information, thus improving administrative efficiency and academic tracking.

1.3 Abstract

The Student Management System is a robust desktop application designed to simplify and streamline the process of managing student records. The system allows administrators to handle all relevant student data, including personal details, academic performance, attendance, and contact information. Built using Python (with Tkinter for GUI) and MySQL as the backend, the system offers a secure, modular, and user-friendly interface. It enhances the accuracy, accessibility, and scalability of student data management, making it an essential tool for educational institutions looking to digitize their operations and improve workflow efficiency.

2. Requirement Specification

2.1 Introduction

The Student Management System (SMS) is designed to manage academic and personal information of students within an educational institute. This includes operations such as storing, updating, deleting, and retrieving data related to students like roll number, name, class, contact information, and academic details. The system enhances administrative efficiency and provides easy access to structured student records. The system needs to be efficient, user-friendly.

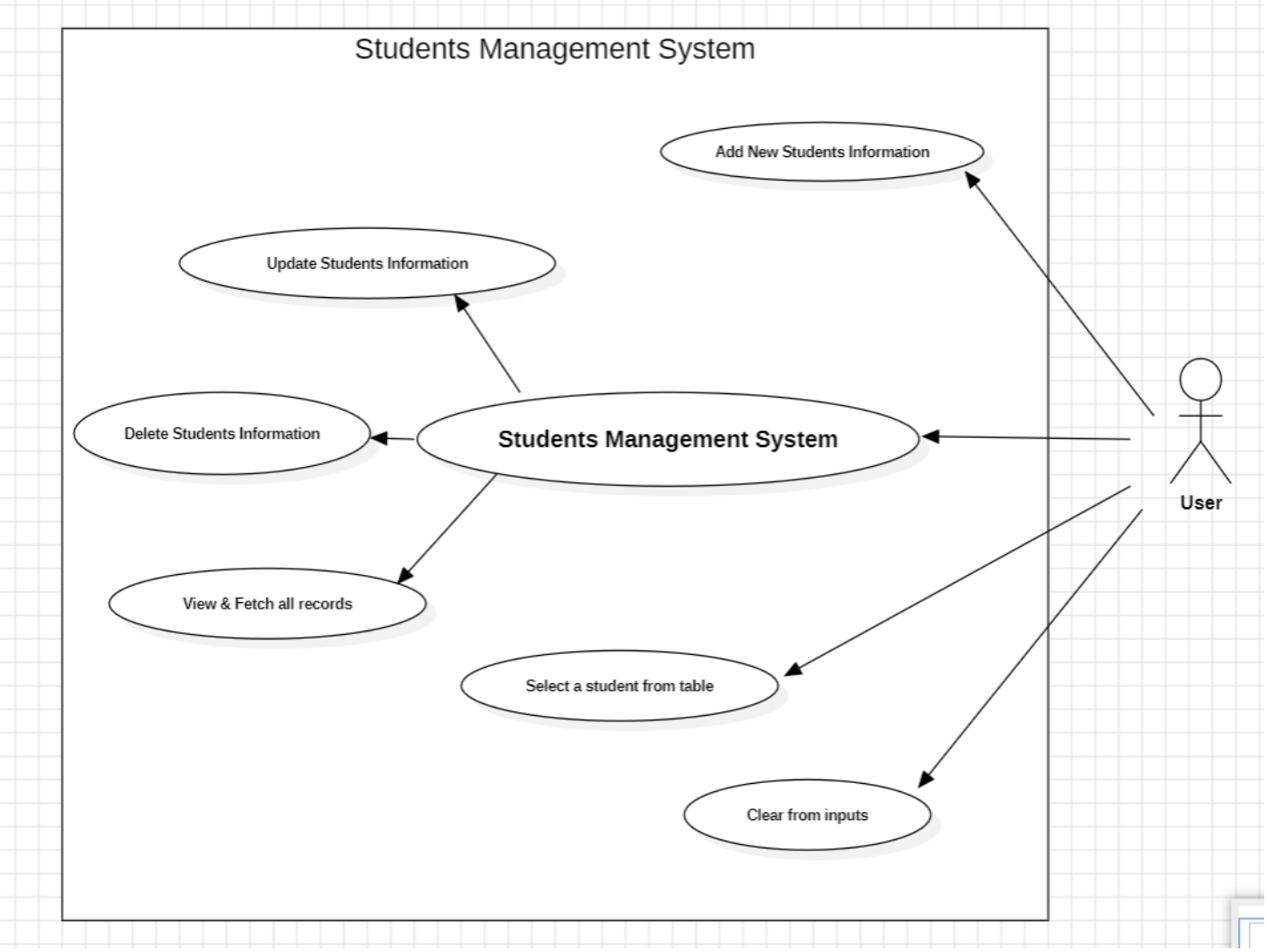
2.2 Functional Requirements

1. Admin Operations:
   - Add new student records to the database.
   - Edit existing student records.
   - Delete student records when necessary.
   - View the complete list of students.

2.3 Non-Functional Requirements

- **Performance**: Optimized to handle large datasets efficiently. Queries and data fetch operations respond swiftly.
- **Robustness**: Capable of handling invalid or incomplete inputs gracefully through proper validation and error handling.
- **Testability**: The system can be tested using typical use cases such as CRUD operations (Create, Read, Update, Delete) on student data.
- **Usability**: Designed with a user-friendly GUI using Tkinter to make navigation and operations intuitive.

2.4 Use Case Diagram



3. System Design

3.1 Architectural Style

The Student Management System follows a Layered Architecture, which helps separate concerns and promotes scalability and maintainability. The layers are:

• **Presentation Layer:** This is the user interface created using Tkinter. It allows the admin to perform operations like adding, updating, deleting, and searching student records.

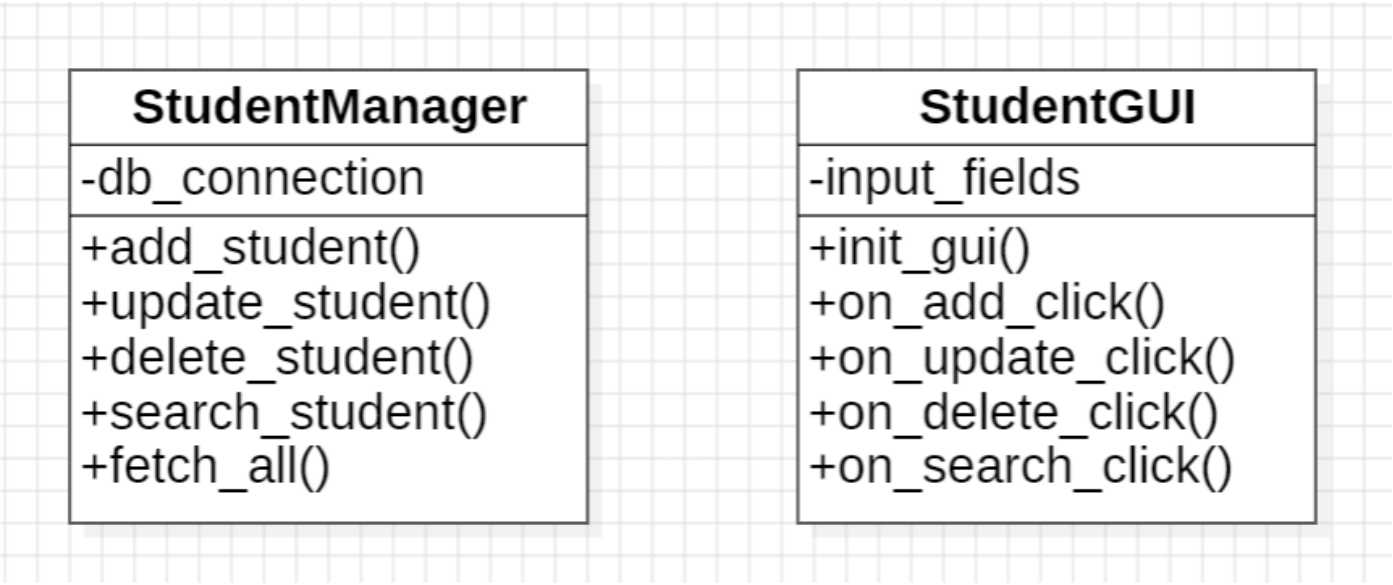
• **Business Logic Layer:** Handles the core logic for input validation, data manipulation, and decision-making (e.g., when a field is left empty, it shows an error).

• **Data Access Layer:** Communicates with the MySQL database using pymysql for data operations like SELECT, INSERT, UPDATE, DELETE.

3.2 Design Pattern

- Singleton: Can be implemented to ensure only one database connection exists during runtime (currently not enforced but a best practice).

3.3 Class Diagram



4. Design Patterns and Architecture

4.1 Code Structure

The Student Management System project is organized into modular parts for clarity and maintainability:

Main GUI (Tkinter):
Responsible for layout, input fields, buttons, and events.

Database Interaction:
Uses the pymysql library to perform database operations.
All queries (insert, update, delete, select) are handled in response to GUI events.

Functions:
add_fun(): Adds a new student record.
update_fun(): Updates existing student details.
delete(): Deletes a student record.
search_data(): Searches for students based on selected criteria.
fetch_data(): Retrieves and displays all student records.

4.2 Code Snippets

- **Connecting to MySQL Database**

```
conn = pymysql.connect(host="localhost", user="root", password="Mypvm@27", database="sms1")
curr = conn.cursor()
```

- **Adding a Student Record**

```
curr.execute("INSERT INTO data VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)", (rollno.get(), name.get(), class_var.get(), division.get(), contact.get(), fathername.get(), address.get(), gender.get(), dob.get()))
```

- **Updating a Record**

```
curr.execute("""
        UPDATE data SET name=%s, class=%s, division=%s, contact=%s, fathername=%s, address=%s, gender=%s, dob=%s WHERE rollno=%s
        """, (name.get(), class_var.get(), division.get(), contact.get(), fathername.get(), address.get(), gender.get(), dob.get(), rollno.get()))
```

- **Searching Records**
  query = f"SELECT * FROM data WHERE {field} LIKE '%{search_value}%'"
  curr.execute(query)

4.3 Testing

Testing was carried out in the following areas:

- **Functional Testing:**
  - Verified that the Add, Update, Delete, and Search operations perform correctly.
  - Checked that data displayed in the TreeView matches database records.
- **GUI Testing:**
  - Ensured all input fields and buttons are responsive.
  - Verified that validation messages appear for incomplete inputs.
- **Database Testing:**
  - Checked for correct data insertion, update propagation, and safe deletion.
  - Verified the use of parameterized queries to prevent SQL injection.
- **Exception Handling:**
  - Errors such as blank fields or invalid queries prompt informative error messages using messagebox..

5. Implementation and Evaluation

5.1 Pattern Effectiveness

The system loosely follows the MVC pattern:

- **Model**: MySQL stores student data.
- **View**: Tkinter provides the user interface.
- **Controller**: Python functions link the UI with database actions.
  This structure improves code organization and maintainability.

5.2 Performance Evaluation

- **Responsiveness**: CRUD operations are fast for up to ~1,000 records.
- **Database**: Efficient for basic queries; may slow down without indexing.
- **GUI**: Tkinter remains responsive during frequent updates.
- **Resources**: Low memory and CPU usage on standard systems.

5.3 Challenges Faced

- **Input Validation:** Fixed by adding checks and error messages.
- **DB Connections:** Resolved locked tables by properly closing connections.
- **SQL Injection:** Prevented by switching to parameterized queries.
- **GUI Overlap:** Solved with manual layout adjustments.
- **Scalability:** Slight lag with large data; pagination is recommended.

6. Conclusion

The Student Management System effectively handles student records with a user-friendly Tkinter GUI and reliable MySQL backend. It supports core operations like add, update, delete, and search efficiently.
Overall, the project meets its goals and provides a solid base for advanced features like attendance tracking and performance reports.