STUDENT RECORD MANAGEMENT SYSTEM

Project submitted to the

Wentworth Institute, Elizabeth Street

for the partial fulfilment of the requirements to award the degree of

**Bachelor of Interactive Media**

Submitted By

|  |  |
| --- | --- |
| **Puspanjali Neupane** | **988636** |

Under the Guidance of

[**Weiyi Wang (Michelle)**](mailto:weiyi.wang@win.edu.au)

**&**

[**Mahtab Khani Khani**](mailto:mahtab.khani@win.edu.au)



**Wentworth Institute, 302-306 Elizabeth Street, Surry Hills, NSW 2010, Australia**

Table of Contents

[1. Introduction 3](#_Toc211433981)

[1.1. Development Process 3](#_Toc211433982)

[2. System Design and Architecture 4](#_Toc211433983)

[2.1. Architectural Overview 4](#_Toc211433984)

[2.2. Technology Stack 4](#_Toc211433985)

[2.3. Design Principles 4](#_Toc211433986)

[3. Implementation Details 5](#_Toc211433987)

[3.1. Backend Module (student\_manager.py) 5](#_Toc211433988)

[3.2. Frontend Module (main.py) 5](#_Toc211433989)

[3.3. Data File (students.txt) 5](#_Toc211433990)

[4. Testing and Validation 6](#_Toc211433991)

[5. Challenges Faced 7](#_Toc211433992)

[6. User Manual 8](#_Toc211433993)

[6.1. Launching the Application 8](#_Toc211433994)

[6.2. Using the Interface 8](#_Toc211433995)

[6.3. Auto-save and Backup 8](#_Toc211433996)

[7. Project Structure 9](#_Toc211433997)

[Before Login 10](#_Toc211433998)

[After Login 11](#_Toc211433999)

[9. Conclusion 12](#_Toc211434000)

# Introduction

The Student Record Management System (SRMS) is a Python-based desktop application developed using the Tkinter library for the frontend and a custom backend module for data management. The primary objective of this project is to design and implement a user-friendly, reliable, and persistent system that allows academic staff to manage student records efficiently. The system enables functionalities such as adding new students, enrolling subjects, marking completed subjects, searching for records, and generating statistics.

### **Development Process**

The development of the **Student Record Management System** followed an **iterative and modular** software development approach. The goal was to ensure smooth progression from concept to implementation while maintaining flexibility for improvements.

The process began with a clear understanding of project requirements creating a desktop-based student management application with role-based access, record management, and a clean graphical user interface.

Development was divided into manageable stages:

1. **Planning and Analysis:** Defined the core objectives, user roles (Admin, Teacher, Viewer), and required features such as add, update, search, and remove functionalities.
2. **Design:** Focused on modular architecture separating backend logic (data management) from the frontend interface (Tkinter-based GUI).
3. **Implementation:** Each module was coded, tested, and refined incrementally, ensuring integration between user actions and data storage.
4. **Testing and Debugging:** Performed iterative testing to ensure functionality, accuracy, and reliability.

The outcome was a robust, well-documented system that aligns with real-world record management workflows.

# System Design and Architecture

### **Architectural Overview**

The SRMS follows a modular architecture, consisting of three core components:

1. Backend Logic (student\_manager.py)
2. Frontend GUI (main.py)
3. Persistent Data Storage (students.txt)

The backend handles data operations and file management, while the frontend provides an interactive graphical interface. The persistent data file ensures data integrity between sessions.

### **Technology Stack**

1. Programming Language: Python 3.10+
2. GUI Framework: Tkinter
3. Data Storage: Text file (students.txt)
4. Additional Libraries: datetime, shutil, typing

### **Design Principles**

The system design follows key software engineering principles:

1. Separation of Concerns: GUI and backend are independent modules.
2. Modularity: Each feature is encapsulated in a dedicated method.
3. Error Handling: Robust exception handling ensures stability.
4. Scalability: Designed to support additional features like API integration, RBAC, and analytics.

# Implementation Details

### **Backend Module (student\_manager.py)**

The backend module manages all logical and data operations. It defines two main classes:

1. Student: Represents individual student entities with attributes such as ID, name, enrolled subjects, completed subjects, and marks.
2. StudentManager: Manages CRUD operations, data persistence, auto-backups, undo functionality, and statistics generation.

Advanced features implemented include auto-save with backups and undo-last-action using a stack data structure.

### **Frontend Module (main.py)**

The frontend provides an interactive GUI built with Tkinter. It includes a role-based login system supporting three roles: Admin, Teacher, and Viewer. Each role has specific access rights:

1. Admin: Full access (add/update/remove)
2. Teacher: Limited access (enroll and mark completion)
3. Viewer: Read-only access

The GUI features multiple tabs for adding students, enrolling subjects, marking completions, and searching records. It also includes live statistics and a Treeview-based student list display.

### **Data File (students.txt)**

The students.txt file stores all student data in a structured format, ensuring persistence. Each line corresponds to one student's record with comma-separated fields for ID, name, subjects, and marks..

AP22110011492,Student Name,COMP101;MATH201,PHYS101;CHEM101,95;92

# Testing and Validation

Extensive testing was conducted to ensure the reliability of all system components. Unit tests were implemented in the backend module to verify each core functionality including data loading, saving, search, enrollment, subject completion, and undo operations.

Manual GUI testing covered the following scenarios:

1. Role-based login authentication
2. Adding and removing students
3. Enrolling and marking completed subjects
4. Searching for student records
5. Displaying system statistics
6. Undoing the last operation

All test cases passed successfully without runtime errors.

# Challenges Faced

During the development of the Student Management System, several challenges were encountered:

1. Designing a modular yet cohesive architecture that integrates GUI and backend seamlessly.
2. Managing data consistency when subjects are updated or removed.
3. Implementing an effective undo feature using a stack structure.
4. Ensuring persistent auto-saving and backup creation to prevent data loss.
5. Debugging Tkinter layout responsiveness and managing widget state across roles

# User Manual

### **Launching the Application**

1. First Install latest Python in your device from [Click Here to Download](https://www.python.org/ftp/python/3.14.0/python-3.14.0-amd64.exe)
2. For your ease install PyCharm Software and import python file as new project and choose global env to create python environment.
3. Run the main.py file in PyCharm.
4. The login window appears with role selection and demo credentials:

* Admin: admin / admin123
* Teacher: teacher / teach123
* Viewer: viewer / view123

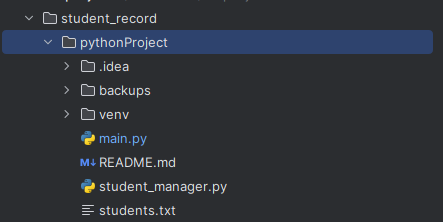
### **Using the Interface**

1. Add Student Tab: Enter student ID, name, and subjects to add a new record.
2. Enroll Subject Tab: Add new subject enrollment for an existing student.
3. Mark Completed Tab: Record completion and marks for enrolled subjects.
4. Search Student Tab: Search records using student ID.
5. Student Records Panel: Displays all records with subject and mark summaries.
6. Bottom Panel: Includes buttons for Refresh, View Statistics, Undo, and Remove.

### **Auto-save and Backup**

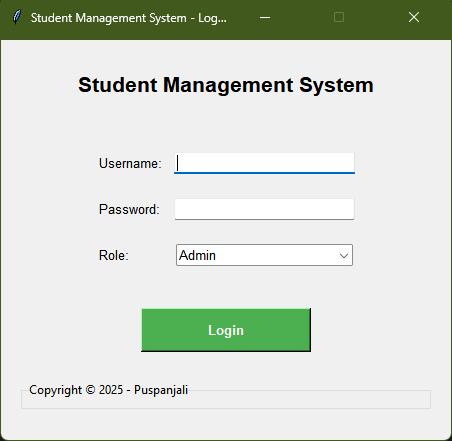
All modifications are automatically saved after each operation. Backup files are created in a 'backups' folder with timestamped filenames to prevent data loss.

# Project Structure

********

1. ****Projects Execution****

## Before Login



## After Login

# Conclusion

This project provided valuable experience in software design, GUI programming, and project management. Developing the Student Record Management System improved my understanding of modular design, object-oriented programming, and real-world application of software engineering principles. Integrating role-based access control, backups, and undo functionality demonstrated the importance of user experience and reliability. Overall, this project strengthened both my technical and analytical skills.