

A PROJECT REPORT ON

Student Database Management System

SUBMITTED BY

Rushikesh Atole ,
Rushikesh Shiral ,
Pavan Bargal

IN THE PARTIAL FULFILLMENT OF MCA, FIRST-YEAR Sem-II EXAMINATION

TO

SANT GADGEBABA AMRAVATI UNIVERSITY, AMRAVATI

Under The Faculty of Science & Technology



SARASWATI COLLEGE, SHEGAON

A.Y. 2024-2025

SARASWATI COLLEGE, SHEGAON

DEPARTMENT OF MASTER IN COMPUTER APPLICATION



CERTIFICATE

Date: /05/2025

This is to certify that,

Mr./Miss.-----

has successfully completed his/her project work on **Student Database management System** in the partial fulfillment of MCA, First Year Sem-II Examination of Sant Gadgebaba Amravati University, Amravati, during the academic year 2021-2022.

Principal

HOD

Saraswati College, Shegaon
Shegaon

Department of MCA, Saraswati College,

(Internal Guide/Examiner)

(External Examiner)

Place: Shegaon

Declaration by the Student

I (Name of student)-----hereby declared
that I have completed my project for the fulfillment of MCA 1st year, Sem-II
examination for the academic year 2024-2025 (Name of
Project)-----.

The above-mentioned project fulfills all the requirements given by Sant
Gadgebaba Amravati, University, Amravati.

If the above-mentioned project proves fake or does not meet any
requirement given by Sant Gadgebaba Amravati, University, Amravati, in this
case, I will be fully responsible for all the consequences.

Date : /05/202

Signature of declaratory

(Name of student)

Place: Shegaon

ACKNOWLEDGEMENT

Inspiration and guidance are valuable in every aspect of life, especially in the field of academics, which I have received from our respected project guide **Prof.B. L. Rathi**, Prof. Prabhanjan Chaudhari, HOD, MCA and Dr. Santosh Bothe, Principal Saraswati, College, Shegaon.

I am thankful for his/her guidance from commencement to completion of this project. I am also thankful to all the teachers of our department.

I remain always thankful to my parents and friends for their support and encouragement.

Date :-___/05/2025

Signature

(Name of student)

CONTENT AT GLANCE

Table of Contents

1. About the Project
 - Introduction
 - My Role (as Team Lead) / Team Structure
 - List of Group Members
 - Specific Contributions
2. Existing System vs. Proposed System Overview
3. Dependencies
4. Current Problem Diagram
5. Proposed Solution Diagram
6. System Requirement Specification
7. Logical and Physical Design
8. System Flow Chart
9. Data Flow Diagram
10. Entity Relationship Diagram
11. Database Description
12. Module Description
13. Screenshots
14. Coding Details
15. System Testing
16. Validation Checks
17. Limitations
18. Future Enhancements
19. Conclusion
20. Bibliography

About the Project

1.1 Introduction :

The Student Database Management System is a desktop application developed using Python, Tkinter, and SQLite. Its primary objective is to provide an efficient and user-friendly platform for educational institutions to manage student records, academic information, payments, and related administrative tasks. The system aims to replace manual or fragmented data management processes with a centralised, secure, and easily accessible digital solution. Key features include user authentication, comprehensive student data management (CRUD operations), report generation, ID card creation, fee receipt generation, basic analytics, and a feedback mechanism. The use of tkbootstrap enhances the graphical user interface (GUI) for a modern look and feel, while Pillow is utilised for image processing tasks.

My Role (as Team Lead) / Team Structure:

As the Team Lead for this project, my role involved guiding the overall project direction, from initial concept through to final implementation and documentation. This included architectural design, database structuring, leading the development of core application components, and ensuring effective collaboration among team members. The team was structured to leverage individual strengths across different aspects of the project.

List of Group Members:

- Rushikesh Atole (Team Lead)
- Rushikesh Shiral
- Pavan Bargal

Specific Contributions:

- **Rushikesh Atole (Team Lead):**
 - **Project Management & Coordination:** Led project planning, task allocation, and ensured timely completion of milestones. Facilitated communication and integration efforts within the team.
 - **System Architecture & Design:** Designed the overall system architecture, including the modular structure of the application and the custom title bar functionality.
 - **Database Design:** Led the design and normalisation of the SQLite database schema (init_db structure and relationships).
 - **Core Module Development:** Developed the primary user authentication system (Login, Registration, Update Password windows) and the main application framework (MainApplication class structure, notebook setup).

- **UI/UX Oversight:** Provided direction for the overall user interface design and user experience, ensuring consistency and intuitiveness.
 - **Final Integration & Reporting:** Oversaw the integration of all modules and compiled the final project report.
 - **Rushikesh Shiral:**
 - **Student Management Module:** Led the development of the comprehensive "Student Management" tab, including all CRUD (Create, Read, Update, Delete) operations, student search functionality, and Treeview data display.
 - **Profile Picture Handling:** Implemented the functionality for uploading, storing, and displaying student profile pictures.
 - **ID Card Generation Module:** Developed the "ID Card Generation" tab, including student data retrieval, image processing using Pillow for card creation, and preview/save features.
 - **Analytics & Insights Module:** Implemented the backend logic and UI for the "Analytics & Insights" tab, including queries for students per course, average marks, and enrollment status breakdown.
 - **Unit Testing:** Conducted unit testing for the modules developed, ensuring their functionality and robustness.
 - **Documentation:** Contributed to the documentation of the Student Management, ID Card, and Analytics modules.
 - **Pavan Bargal :**
 - **Reports Module:** Developed the "Reports" tab, implementing functionality for generating student enrollment reports, marks reports (with filtering by course/semester), and payment history reports.
 - **Receipt Generation Module:** Implemented the "Receipt Generation" tab, including payment data input, database recording, and formatted receipt display.
 - **Feedback Module:** Developed the "Feedback" tab, enabling users to submit their suggestions and ensuring feedback is stored in the database.
 - **System & Integration Testing:** Played a key role in overall system testing, including integration testing of different modules and end-to-end validation checks.
 - **Database Population & Helper Functions:** Assisted in populating initial database lookup tables (faculties, academic years, courses) and developed helper functions for fetching data for UI comboboxes.
 - **Documentation:** Contributed to the system testing documentation and sections of the user manual related to reports, receipts, and feedback.
-

Existing System vs. Proposed System Overview

Existing System (Assumed Generic):

Many educational institutions, especially smaller ones, might rely on:

- **Manual Paper-Based Systems:** Records kept in physical files, registers, and ledgers. This leads to difficulties in data retrieval, high risk of data loss or damage, redundancy, and time-consuming administrative tasks.
- **Spreadsheet-Based Systems (e.g., Excel):** While an improvement over paper, spreadsheets can become unwieldy for large datasets, lack robust data validation, offer limited security, and are not ideal for concurrent access or complex querying.
- **Fragmented Digital Systems:** Different departments might use separate, non-integrated software, leading to data silos and inconsistencies.

Problems with Existing Systems:

- Inefficient data retrieval and updates.
- Prone to human errors and data inconsistencies.
- Lack of data security and access control.
- Difficult to generate comprehensive reports or perform analytics.
- Time-consuming administrative overhead.
- Poor scalability.

Proposed System (This Project)

The developed Student Database Management System offers:

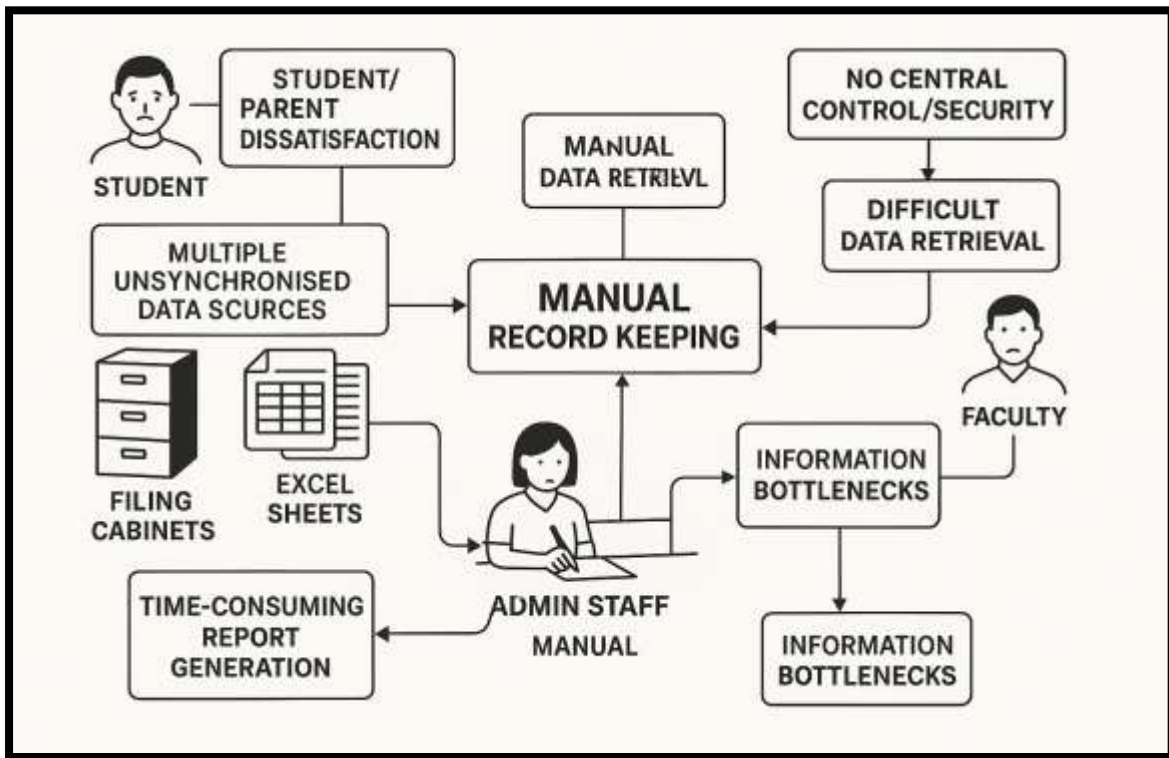
- **Centralised Database:** All student-related information is stored in a single SQLite database, ensuring data integrity and consistency.
 - **Graphical User Interface (GUI):** A user-friendly interface built with Tkinter and ttkbootstrap, making it easy for non-technical staff to operate.
 - **User Authentication:** Secure login for authorised users, with password hashing.
 - **Comprehensive Student Management:** CRUD operations for detailed student profiles, including personal, academic, and contact information, and profile picture uploads.
 - **Automated Processes:** Generation of ID cards, payment receipts, and various reports.
 - **Data Validation:** Input checks to minimise errors.
 - **Basic Analytics:** Insights into student enrollment, academic performance, etc.
 - **Feedback Mechanism:** Allows users to provide suggestions for system improvement.
 - **Improved Efficiency:** Reduces manual effort and speeds up administrative tasks.
 - **Enhanced Security:** Password protection and controlled access.
-

Dependencies

The project relies on the following Python libraries and external assets:

- **Python 3.x:** Core programming language.
 - **tkinter:** Standard Python interface to the Tk GUI toolkit.
 - `tkinter.ttk`: Themed Tkinter widgets.
 - `tkinter.messagebox`: Standard dialogs.
 - `tkinter.filedialog`: File open/save dialogues.
 - **sqlite3:** DB-API 2.0 interface for SQLite databases.
 - **os:** Miscellaneous operating system interfaces (used for file path checks).
 - **ttkbootstrap:** A collection of modern, flat themes for Python's Tkinter Ttk framework.
 - **datetime (from the datetime module):** For handling dates and timestamps (e.g., enrollment date, payment date, feedback timestamp).
 - **Pillow (PIL Fork):** Python Imaging Library, used for image processing (loading, resizing, creating ID cards).
 - `PIL.Image`
 - `PIL.ImageDraw`
 - `PIL.ImageFont`
 - `PIL.ImageTk`
 - **io:** Core tools for working with I/O streams.
 - **hashlib:** Secure hashes and message digests (used for SHA-256 password hashing).
 - **Image Assets (External Files):**
 - `logo.png`: Application logo.
 - `college_info.png`: Image for the home page.
 - `collegeview.jpeg`: Background image for the main application.
 - `identitycard.jpg`: Background template for student ID cards.
 - Font files (e.g., `arialbd.ttf`, `arial.ttf`) are attempted for ID card text; system default fonts are used as a fallback.
-

Current Problem Diagram



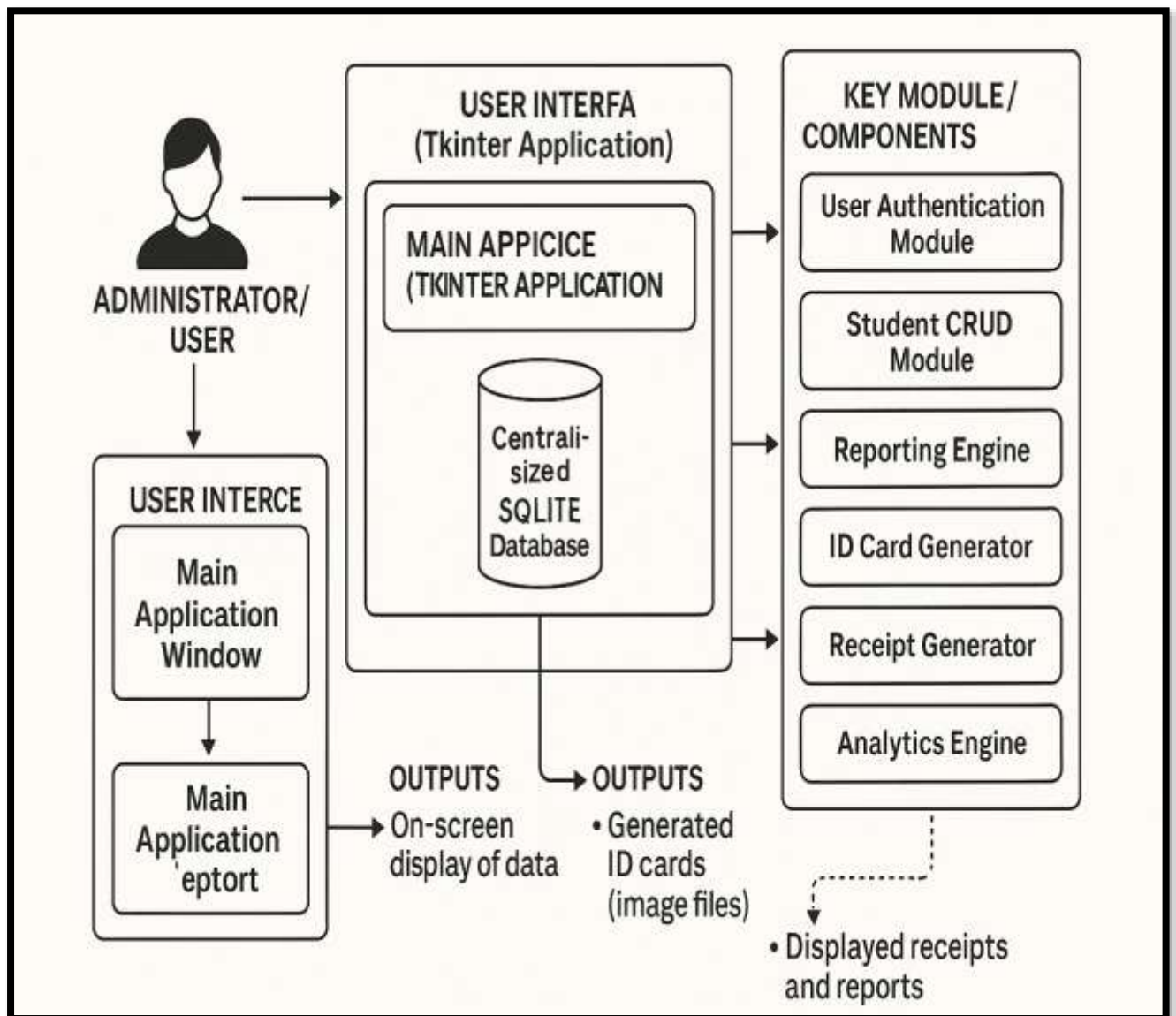
A diagram here would visually represent the inefficiencies and challenges of a manual or outdated student record management system.

Key elements to depict:

- **Multiple Unsynchronised Data Sources:** Separate piles of paper records, isolated spreadsheets.
- **Manual Data Entry & Updates:** Staff members' handwriting information or typing into basic files, leading to high error rates and redundancy.
- **Difficult Data Retrieval:** Staff searching through physical files or complex, unindexed spreadsheets.
- **No Central Control/Security:** Records susceptible to physical damage, unauthorised access, or loss.
- **Time-Consuming Report Generation:** Manual compilation of data from various sources.
- **Student/Parent Dissatisfaction:** Due to service delays, errors in records, or difficulty accessing information.
- **Information Bottlenecks:** Specific staff members being the only ones who know how to find certain information.

The diagram would use symbols for processes (e.g., "Manual Record Keeping"), data stores (e.g., "Filing Cabinets," "Excel Sheets"), and external entities (e.g., "Students," "Admin Staff," "Faculty"), with arrows showing cumbersome and error-prone data flows.

5. Proposed Solution Diagram



A diagram here would illustrate the architecture and streamlined workflow of the proposed Student Database Management System.

Key elements to depict:

- **Centralised SQLite Database:** A single data store at the core of the system, holding all student, course, faculty, payment, and user information.
- **User Interface (Tkinter Application):** The main point of interaction for users (Admin Staff).
 - Login Module: Secure access point.
 - Main Application Window (with Tabs): Student Management, Reports, ID Card, Receipts, Analytics, Feedback.

- **Key Modules/Components (as blocks within the application):**
 - User Authentication Module
 - Student CRUD Module
 - Reporting Engine
 - ID Card Generator
 - Receipt Generator
 - Analytics Engine
 - Feedback Module
- **Data Flows:**
 - Users interacting with the GUI.
 - GUI modules reading from and writing to the SQLite database.
 - Data processing for reports, ID cards, etc.
- **External Entities:**
 - Administrator/User: Interacting with the system.
- **Outputs:**
 - On-screen display of data.
 - Generated ID cards (image files).
 - Displayed receipts and reports.

The diagram would show a clean, integrated system where data flows logically through defined modules to and from the central database, highlighting efficiency and ease of use.

System Requirement Specification

Functional Requirements:

- **User Management:**
 - FR1: System shall allow new user registration with a unique username and hashed password.
 - FR2: System shall authenticate users based on username and password.
 - FR3: System shall allow users to update their passwords after verifying their old password.
- **Student Data Management:**
 - FR4: The System shall allow adding new student records with comprehensive details.
 - FR5: The System shall allow viewing a list of all students.
 - FR6: The System shall allow searching for students by roll number or name.
 - FR7: The System shall allow updating existing student records.
 - FR8: The System shall allow deleting student records.
 - FR9: The System shall allow uploading and associating a profile picture.
- **Report Generation:**
 - FR10: The System shall generate student enrollment reports.
 - FR11: System shall generate mark reports filtered by course and semester.
 - FR12: System shall generate payment history reports.
- **ID Card Generation:**
 - FR13: System shall generate a printable student ID card.
 - FR14: The System shall allow saving the generated ID card.
- **Receipt Generation:**
 - FR15: The System shall allow recording student payments.
 - FR16: The System shall generate a payment receipt.
- **Analytics & Insights:**
 - FR17: The System shall provide analytics on students per course.
 - FR18: The System shall provide analytics on average marks per course.
 - FR19: System shall provide an enrollment status breakdown.
 - FR20: The System shall provide insights into faculty academic performance.
- **Feedback:**
 - FR21: System shall allow users to submit feedback.
- **Database Initialisation:**
 - FR22: System shall create necessary database tables if they do not exist.
 - FR23: System shall pre-populate essential data.

Non-Functional Requirements:

- **Usability (NFR1):** User-friendly graphical interface.
- **Performance (NFR2):** Reasonable system response times.
- **Security (NFR3):** Hashed passwords, authenticated access.
- **Reliability (NFR4):** Data integrity, graceful error handling.
- **Maintainability (NFR5):** Modular and well-commented code.
- **Portability (NFR6):** Capable of running on an OS supporting Python and dependencies.
- **Scalability (NFR7):** Efficient for moderate-sized institutions.

Hardware Requirements (Minimum):

- Processor: Dual-core CPU
- RAM: 4 GB
- Storage: 500 MB free disk space
- Display: 1024x768 resolution.

Software Requirements:

- Operating System: Windows 7/8/10/11, macOS X, or a modern Linux distribution.
 - Python: Version 3.7 or higher.
 - SQLite: Version 3.x.
 - Required Python Libraries (as listed in Dependencies).
-

Logical and Physical Design

Logical Design:

The logical design focuses on the abstract representation of the system's data, processes, and their interactions.

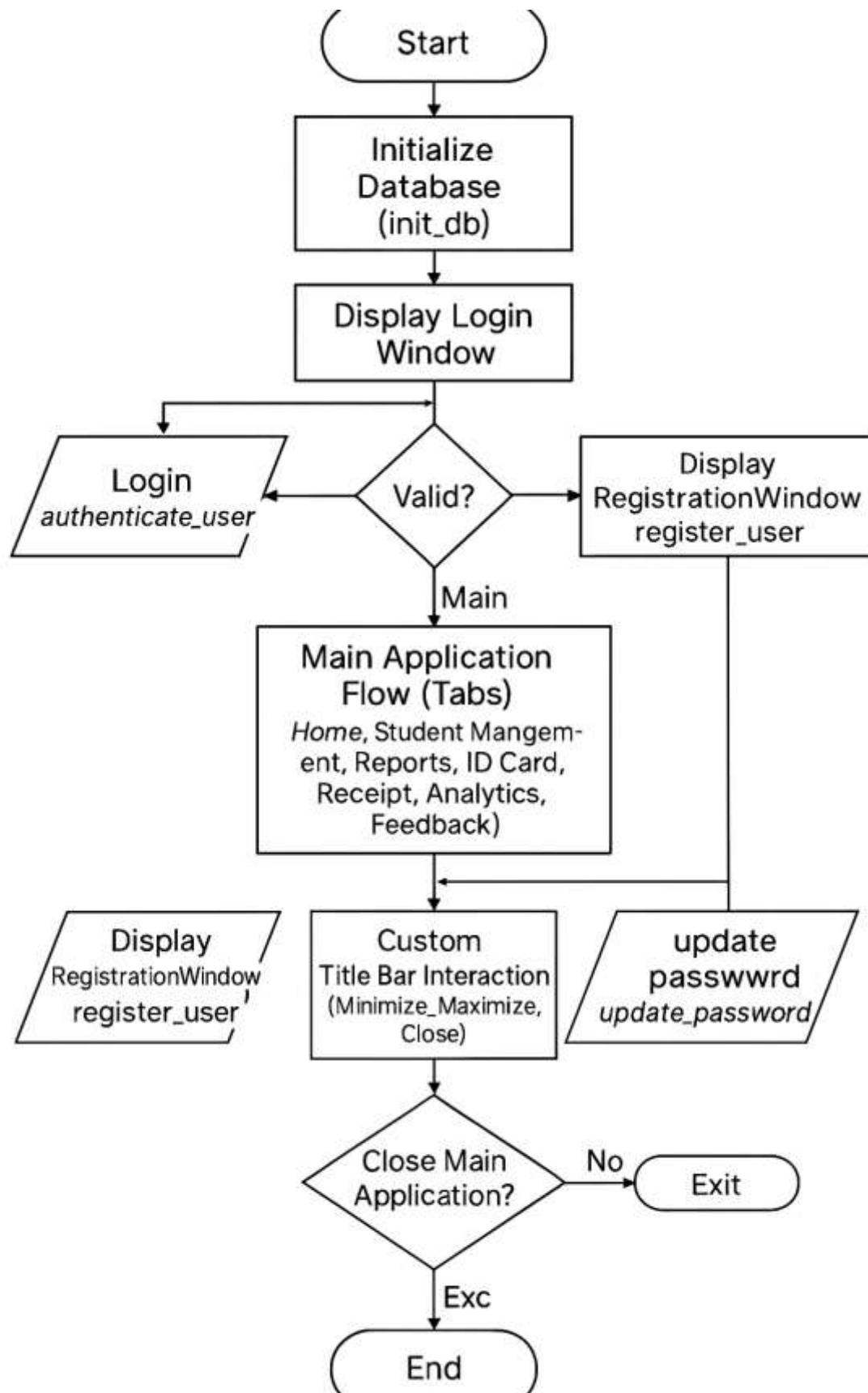
- **Data Entities:** Users, Students, Courses, Faculties, Academic Years, Marks, Payments, Feedback.
- **Process Flow:** Authentication -> Main Application -> Module Interaction (Tabs) -> Data Handling (DB operations) -> Output Generation.
- **Class Structure:** CustomTitleBar, RegistrationWindow, UpdatePasswordWindow, LoginWindow, MainApplication.
- **Data Validation:** Input data is validated before database operations.

Physical Design:

The physical design translates the logical design into specific software and database structures.

- **Database Implementation:** Single SQLite file (student_records.db) with tables: users, faculties, academic_years, students, courses, marks, payments, feedback. Relationships via Foreign Keys.
 - **Application Files:** new_update.py (main script), image assets (logo.png, etc.).
 - **User Interface:** Implemented using Tkinter and ttkbootstrap.
 - **Deployment:** Standalone desktop application requiring Python and dependencies.
-

System Flow Chart

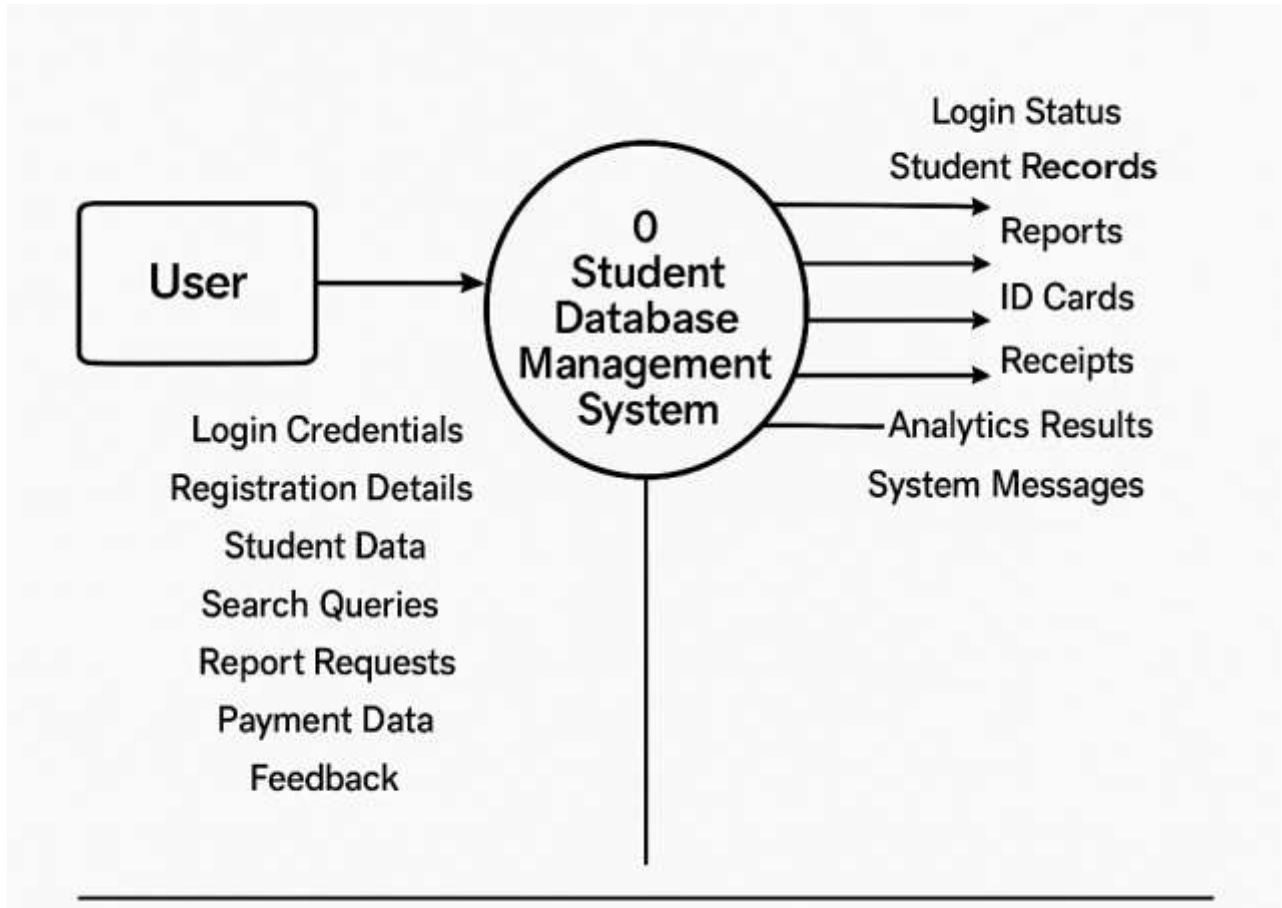


Description of diagram:

- 1. Start**
 - 2. Initialise Database (init_db)**
 - 3. Display Login Window**
 - 4. Decision: Login / Register / Update Password?**
 - **Login:** authenticate_user. Valid? Yes -> Main App. No -> Error.
 - **Register:** Display RegistrationWindow. register_user.
 - **Update Password:** Display UpdatePasswordWindow. update_password.
 - 5. Main Application Flow (Tabs):**
 - User selects a Tab (Home, Student Management, Reports, ID Card, Receipt, Analytics, Feedback).
 - Each tab has specific workflows for data entry, processing, and display. (e.g., Student Management: Add, Update, Delete, Search, Display in Treeview; Reports: Select, Generate, Display).
 - 6. Custom Title Bar Interaction (Minimise, Maximise, Close)**
 - 7. Decision: Close Main Application? Yes -> Exit. No -> Continue.**
 - 8. End**
-

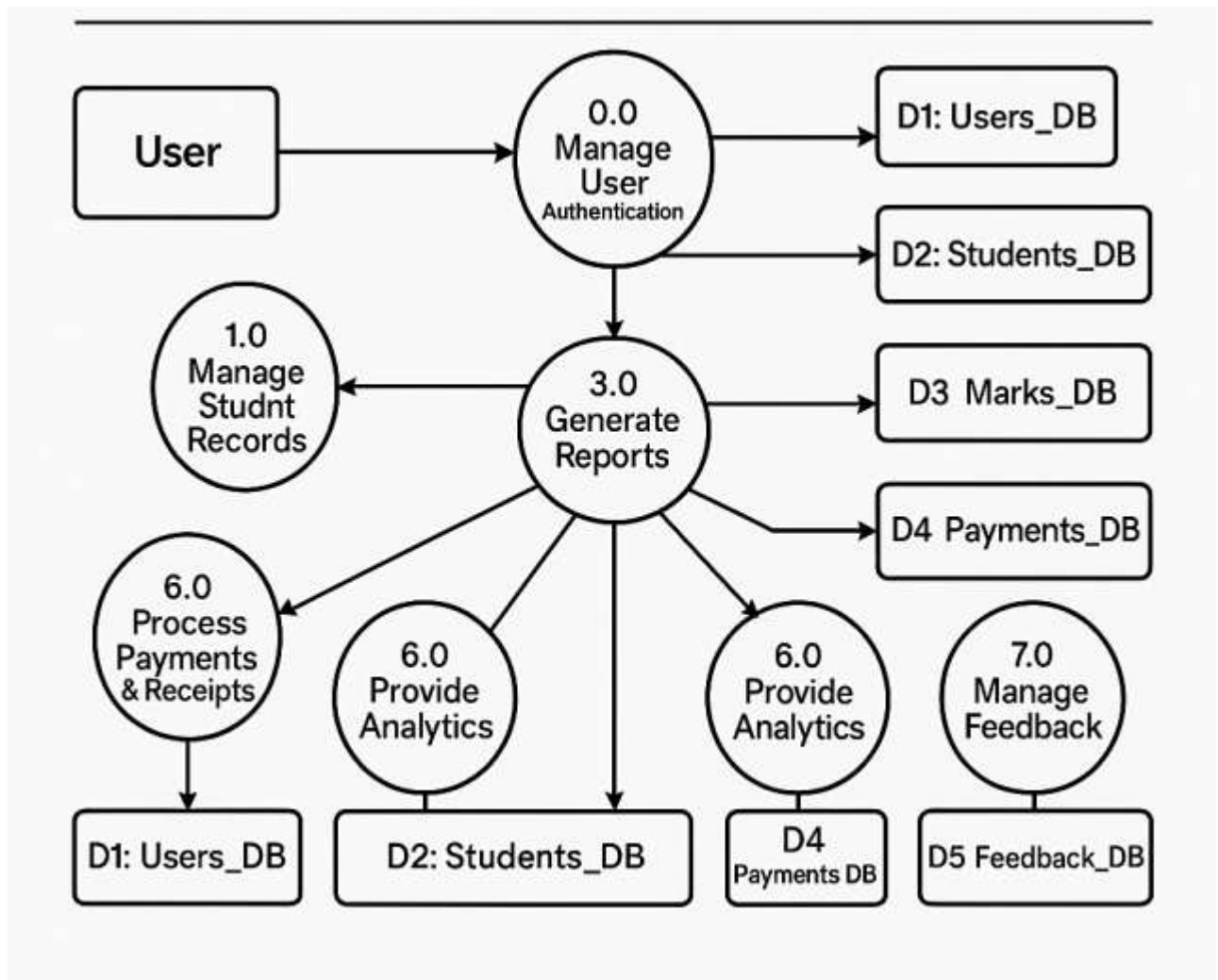
Data Flow Diagram (DFD)

Level 0 DFD (Context Diagram):



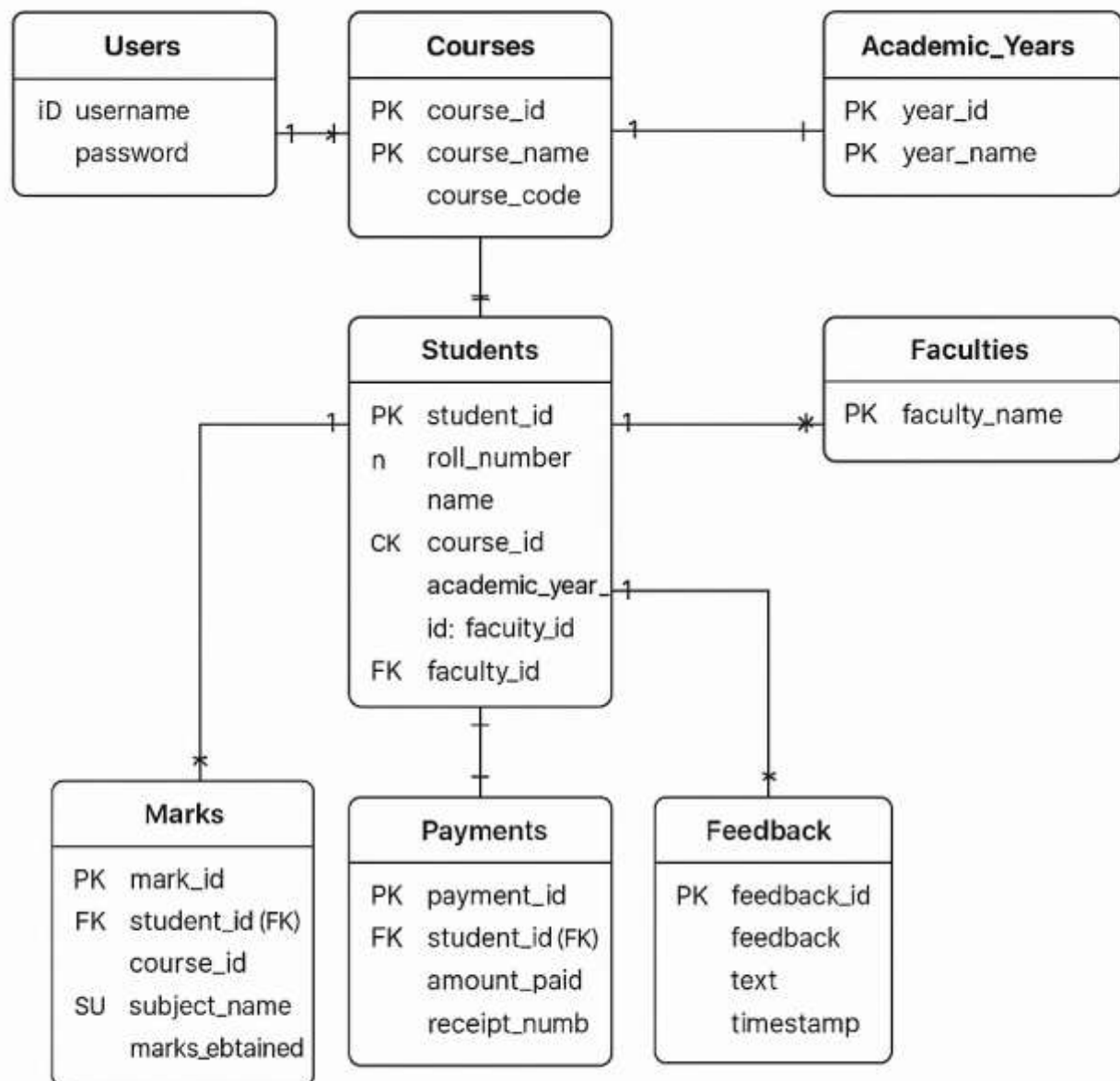
- **External Entity:** User (Administrator/Staff)
- **Process:** 0. Student Database Management System
- **Data Flows (Input from User):** Login Credentials, Registration Details, Student Data, Search Queries, Report Requests, Payment Data, Feedback, etc.
- **Data Flows (Output to User):** Login Status, Student Records, Reports, ID Cards, Receipts, Analytics Results, System Messages, etc.

Level 1 DFD (Partial):



- **External Entity:** User
- **Data Stores:** D1: Users_DB, D2: Students_DB, D3: Marks_DB, D4: Payments_DB, D5: Feedback_DB
- **Processes:**
 1. 1.0 Manage User Authentication (Interacts with D1)
 2. 2.0 Manage Student Records (Interacts with D2)
 3. 3.0 Generate Reports (Reads from D2, D3, D4)
 4. 4.0 Generate ID Cards (Reads from D2)
 5. 5.0 Process Payments & Receipts (Interacts with D4, reads from D2)
 6. 6.0 Provide Analytics (Reads from D2, D3)
 7. 7.0 Manage Feedback (Interacts with D5)

Entity Relationship Diagram (ERD)



Entities & Primary Attributes (PK = Primary Key, FK = Foreign Key):

1. **Users**: id (PK), username, password
2. **Faculties**: faculty_id (PK), faculty_name
3. **Academic_Years**: year_id (PK), year_name
4. **Courses**: course_id (PK), course_name, course_code
5. **Students**: student_id (PK), roll_number, name, ..., course_id (FK), academic_year_id (FK), faculty_id (FK)
6. **Marks**: mark_id (PK), student_id (FK), course_id (FK), subject_name, marks_obtained
7. **Payments**: payment_id (PK), student_id (FK), amount_paid, receipt_number
8. **Feedback**: feedback_id (PK), feedback_text, timestamp

Relationships (Cardinality):

- Courses 1 --- M Students
 - Academic_Years 1 --- M Students
 - Faculties 1 --- M Students
 - Students 1 --- M Marks
 - Courses 1 --- M Marks (related via students taking subjects of a course)
 - Students 1 --- M Payments
-

Database Description

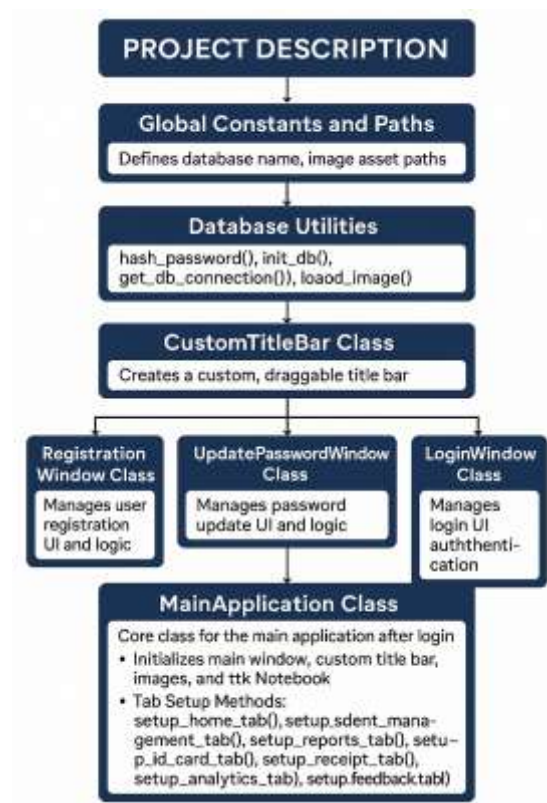
users	faculties
id (PK)	faculty_id (PK)
username (UNIQUE)	faculty_name (UNIQUE)
password	
academic_years	courses
year_id (PK)	course_id (PK)
year_name (UNIQUE)	course_name (UNIQUE)
	course_code (UNIQUE)
	duration
	department

payments
payment_id (PK)
student_id (FK) → students.student_id
amount_paid, payment_date
payment_type, receipt_number (UNIQUE)
description
feedback
feedback_id (PK)
name, email
feedback_text, timestamp
students
student_id (PK)
roll_number (UNIQUE)
name, contact_number, email, ...
course_id (FK) → courses.course_id
academic_year_id (FK) → academic_years.year_id
faculty_id (FK) → faculties.faculty_id
marks
mark_id (PK)
student_id (FK) → students.student_id
course_id (FK) → courses.course_id
subject_name, semester
marks_obtained, max_marks, grade

The system utilises an SQLite database named student_records.db. The schema includes the following tables:

1. **Table: users** (id PK, username UNIQUE, password)
 2. **Table: faculties** (faculty_id PK, faculty_name UNIQUE)
 3. **Table: academic_years** (year_id PK, year_name UNIQUE)
 4. **Table: courses** (course_id PK, course_name UNIQUE, course_code UNIQUE, duration, department)
 5. **Table: students** (student_id PK, roll_number UNIQUE, name, ..., course_id FK, academic_year_id FK, faculty_id FK)
 6. **Table: marks** (mark_id PK, student_id FK, course_id FK, subject_name, semester, marks_obtained, max_marks, grade)
 7. **Table: payments** (payment_id PK, student_id FK, amount_paid, payment_date, payment_type, receipt_number UNIQUE, description)
 8. **Table: feedback** (feedback_id PK, name, email, feedback_text, timestamp)
-

Module Description

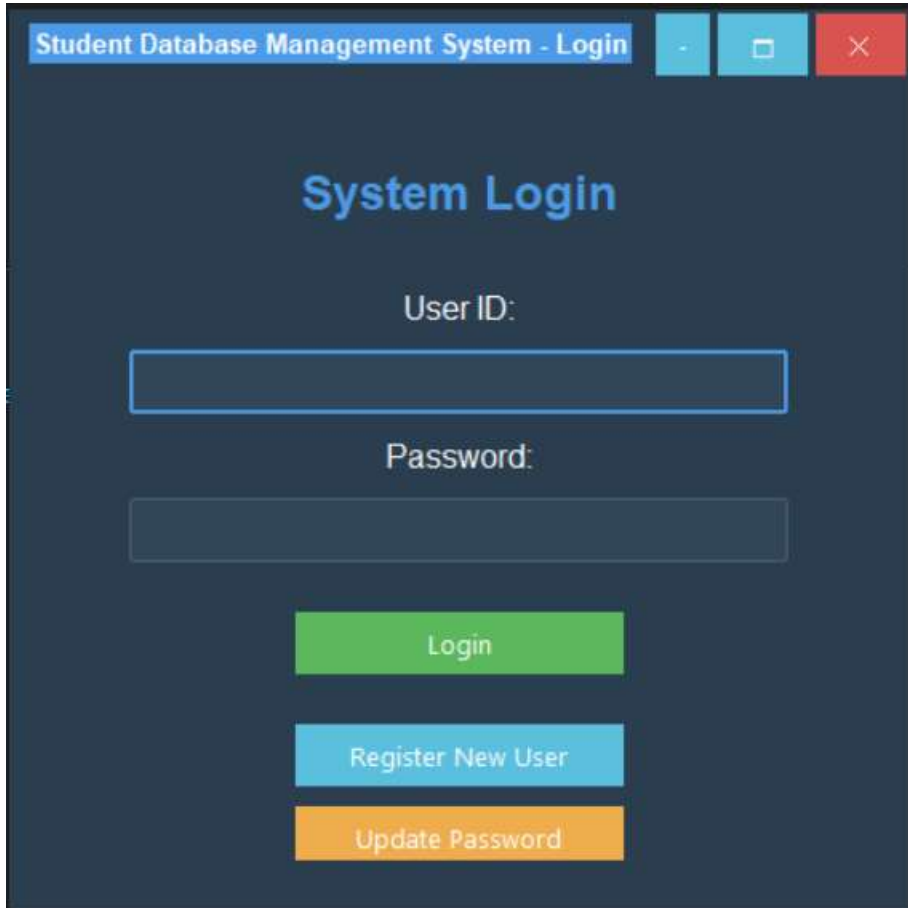


The application is structured into several Python classes and functions:

1. **Global Constants and Paths:** Defines database name, image asset paths.
2. **Database Utilities:** hash_password(), init_db(), get_db_connection(), load_image().
3. **CustomTitleBar Class:** Creates a custom, draggable title bar.
4. **RegistrationWindow Class:** Manages user registration UI and logic.
5. **UpdatePasswordWindow Class:** Manages password update UI and logic.
6. **LoginWindow Class:** Manages login UI and authentication.
7. **Main Application Class:** Core class for the main application after login.
 - Initialises main window, custom title bar, images, and ttk.Notebook.
 - **Tab Setup Methods:** setup_home_tab(), setup_student_management_tab(), setup_reports_tab(), setup_id_card_tab(), setup_receipt_tab(), setup_analytics_tab(), setup_feedback_tab().
 - **Helper Methods:** For student CRUD, report generation, ID card creation, receipt processing, analytics, and feedback submission.

Screenshots:

1. **Login Window:** Custom title bar, User ID, Password fields, Login button, Register New User button, Update Password button.



The screenshot shows a window titled "Student Database Management System - Login". The window has a dark blue background. In the center, the text "System Login" is displayed in a light blue font. Below this, there are two input fields: "User ID:" and "Password:". Below the input fields, there are three buttons: a green "Login" button, a light blue "Register New User" button, and an orange "Update Password" button. The window has a standard Windows-style title bar with minimize, maximize, and close buttons.

2. **Registration Window:** Custom title bar, Full Name, User ID, Password, Confirm Password fields, Register Button, Back button.



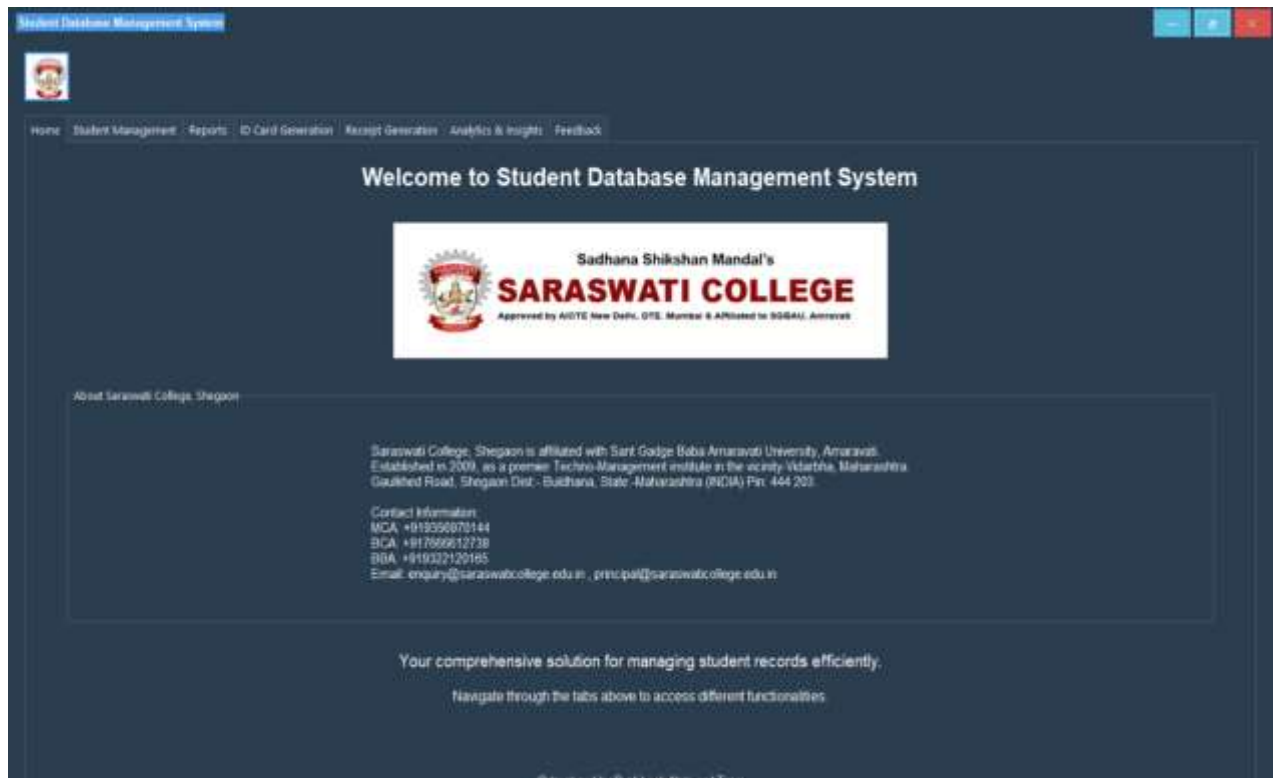
The screenshot shows a window titled "Register New User". The window has a dark blue background. In the center, the text "New User Registration" is displayed in a light blue font. Below this, there are four input fields: "Full Name:", "User ID:", "Password:", and "Confirm Password:". Below the input fields, there are two buttons: a green "Register" button and a grey "Back to Login" button. The window has a standard Windows-style title bar with minimize, maximize, and close buttons.

3. **Password update windows:** Custom title bar, User ID, old Password fields, New Password fields, Confirmation New Password fields, Update Password button, Back to login button.



The screenshot shows a window titled "Update Password" with a dark blue background. It features five input fields: "User ID", "Old Password", "New Password", and "Confirm New Password". Below these fields are two buttons: a green "Update Password" button and a grey "Back to Login" button. The window has a standard OS-style title bar with minimize, maximize, and close buttons.

4. **Main Application - Home Tab:** Active "Home" tab, welcome message, college info image, text details, background image, developer credit.



The screenshot displays the "Student Database Management System" home page. At the top, there's a navigation bar with tabs: Home, Student Management, Reports, ID Card Generation, Receipt Generation, Analytics & Insights, and Feedback. The main content area has a dark blue background with a white box containing the college's logo and name: "Sadhana Shikshan Mandal's SARASWATI COLLEGE". Below this, there's a section titled "About Saraswati College, Shegaon" which provides details about the college's affiliation with Sant Gadge Baba Amravati University and its location. Contact information is also listed, including phone numbers for MCA, BCA, BBA, and an email address. At the bottom, there's a message: "Your comprehensive solution for managing student records efficiently. Navigate through the tabs above to access different functionalities." and a small credit line: "©developed by Rushabh Khire and Team".

- Main Application - Student Management Tab (Data Entry):** Input fields for all student details, profile picture upload area, and CRUD buttons, Search and view Students.

Student Database Management System

Home | Student Management | Reports | ID Card Generation | Receipt Generation | Analytics & Insights | Feedback

Student Record Management (CRUD Operations)

Student Details

Roll No:	01	Name:	Rushabh Hemantraj Kote	Profile Picture Upload Image
Contact No:	9011861230	Email:	ratn451@gmail.com	
Address:	Jalgaon	Address No:	9001861141	
Date of Birth (YYYY-MM-DD):	1999-03-19	Gender:	Male	
TEN No:	01	TEN No:	01	
Blood Group:	A+	Mother's Name:	Sharda	
Enrollment Date (YYYY-MM-DD):	2024-10-17	Enrollment Status:	Yes	
Course:	Computer Applications	Academic Year:	First Year	
Faculty:	MCA			

[Add Student](#)
[Update Student](#)
[Delete Student](#)
[View Field](#)

Search & View Student

Search by Roll No/Name: [Search](#) [Refresh](#)

ID	Roll No	Name	Contact	Email	Address	Address No	DOB
----	---------	------	---------	-------	---------	------------	-----

- Main Application - Student Management Tab (Data Display):** ttk.Treeview with student records, search bar, and refresh button.

Student Database Management System

Home | Student Management | Reports | ID Card Generation | Receipt Generation | Analytics & Insights | Feedback

Student Record Management (CRUD Operations)

Student Details

Roll No:	01	Name:	Rushabh Hemantraj Kote	Profile Picture Upload Image
Contact No:	9045071423	Email:	ratn451@gmail.com	
Address:	Jalgaon Jethod	Address No:	9001861141	
Date of Birth (YYYY-MM-DD):	2000-04-17	Gender:	Male	
TEN No:	01	TEN No:	01	
Blood Group:	A+	Mother's Name:	Sharda	
Enrollment Date (YYYY-MM-DD):	2024-10-17	Enrollment Status:	Yes	
Course:	Computer Applications	Academic Year:	First Year	
Faculty:	BCA			

[Add Student](#)
[Update Student](#)
[Delete Student](#)
[View Field](#)


Search & View Student

Search by Roll No/Name: [Search](#) [Refresh](#)

ID	Roll No	Name	Contact	Email	Address	Address No	DOB
2	01	Rushabh Hemantraj Kote	9045071423	ratn451@gmail.com	Jalgaon Jethod	9001861141	2000-04-17
1	01	Rushabh Hemantraj Kote	9011861230	ratn451@gmail.com	Jalgaon	9001861141	1999-03-19

7. **Main Application - Reports Tab:** Options for Enrollment, Marks (with filters), Payment reports. Text area for report output.

Student Database Management System



HomeStudent ManagementReportsID Card GenerationReceipt GenerationAnalytics & InsightsFeedback

Reports and Data Export

Generate Reports

Student Enrollment Report

Generate Enrollment Report

Marks Report (By Course & Semester)

Course:

Semester:

Generate Marks Report

Payment History Report


Generate Payment Report

Report Output

Student Enrollment Report

Roll No	Name	Enroll Date	Course	Acad Year	Faculty	Status
01	Ravi Patel	2023-06-05	Business Administration	First Year	BBA	Active
02	Roshni Mishra	2023-06-12	Computer Applications	First Year	BSC	Active
03	Anshika Mishra	2023-06-17	Computer Applications	First Year	BSC	Active

Student Database Management System



HomeStudent ManagementReportsID Card GenerationReceipt GenerationAnalytics & InsightsFeedback

Reports and Data Export

Generate Reports

Student Enrollment Report

Generate Enrollment Report

Marks Report (By Course & Semester)

Course:

Semester:

Generate Marks Report

Payment History Report

Generate Payment Report

Report Output

Marks Report For Computer Applications, Semester I

Roll No	Name	Subject	Marks	Pass	Grade
No marks found for the selected criteria.					

8. **Main Application - ID Card Generation Tab (with Preview):** Roll Number input, Generate button. Canvas with ID card preview.

The screenshot shows the 'ID Card Generation' tab of the 'Student Database Management System'. The interface includes a navigation bar with links to Home, Student Management, Reports, ID Card Generation, Receipt Generation, Analytics & Insights, and Feedback. The main heading is 'Generate Student ID Cards'. Below this, there is a 'Student Selection' section with an input field for 'Enter Student Roll Number:' containing the value '01', and a green 'Generate ID Card' button. The 'Generated ID Card Preview:' section displays a sample ID card for 'Saraswati College, Shegaon'. The ID card contains the following details: Student Name: Rushikesh Hemantkumar Ahir, Roll No: 01, Course: Computer Applications (First Year), DOB: 1999-05-18, Email: College@, Contact: 8011341134, and Expiry Date: 2024-10-17. A modal dialog titled 'ID Card Generated' is open, displaying a message: 'ID Card generated successfully! Do you want to save it as an image?' with 'Yes' and 'No' buttons.

9. **Main Application - Receipt Generation Tab (with Preview):** Payment detail inputs, Generate button. Text area with a receipt.

The screenshot shows the 'Receipt Generation' tab of the 'Student Database Management System'. The interface includes a navigation bar with links to Home, Student Management, Reports, ID Card Generation, Receipt Generation, Analytics & Insights, and Feedback. The main heading is 'Generate Payment Receipts'. Below this, there is a 'Payment Details' section with input fields for 'Student Roll Number:' (01), 'Amount Paid (Rs):' (5000), 'Payment Type:' (Tuition Fee), and 'Description (Optional):' (Payment received). A green 'Generate Receipt' button is located below these fields. The 'Generated Receipt Preview:' section displays a sample receipt for 'Saraswati College, Shegaon'. The receipt includes the following details: Receipt No: 49C-562566421117-01, Date: 2023-09-04 23:43:57, Student Name: Rushikesh Hemantkumar Ahir, Roll Number: 01, Course: C, Amount Paid: 5000.0000, Payment Type: Tuition Fee, and Description: payment received. There is a line for 'Signature' and a closing statement 'Thank you for your payment!'. A modal dialog titled 'Receipt Generated' is open, displaying a message: 'Receipt generated and recorded successfully!' with an 'OK' button.

10. **Main Application - Analytics & Insights Tab:** Combobox for insight selection, Generate button. Text area with analytics output.

The screenshot shows the 'Analytics & Insights' tab of the 'Student Database Management System'. The interface has a dark blue theme. At the top, there's a navigation bar with links: Home, Student Management, Reports, ID Card Generation, Receipt Generation, Analytics & Insights (active), and Feedback. Below the navigation bar, the title 'Analytics and Performance Insights' is centered. The main content area is divided into two sections. The first section, 'Generate Insight', contains a dropdown menu labeled 'Choose insight' with 'Students per Course' selected, and a blue 'Generate Insight' button. The second section, 'Analytics Output', displays a table titled 'Students Enrolled Per Course'. The table has two columns: 'Course' and 'Total Students'. The data rows are: Computer Applications (3), Business Administration (1), and Science (6).

Course	Total Students
Computer Applications	3
Business Administration	1
Science	6

11. **Main Application - Feedback Tab:** Name, Email, Feedback text area, Submit button.

The screenshot shows the 'Feedback' tab of the 'Student Database Management System'. The interface has a dark blue theme. At the top, there's a navigation bar with links: Home, Student Management, Reports, ID Card Generation, Receipt Generation, Analytics & Insights, and Feedback (active). Below the navigation bar, the title 'Provide Feedback or Suggestions' is centered. The main content area is divided into two sections. The first section, 'Your Feedback', contains three input fields: 'Your Name (Optional)' (with a placeholder 'Full Name'), 'Your Email (Optional)' (with a placeholder 'rpat@ic@gmail.com'), and a large text area for 'Feedback/Suggestion' (with a placeholder 'I want to extend to take and admission to your'). At the bottom of the form is a green 'Submit Feedback' button.

9. Coding Details:

- **Programming Language:** Python 3.x
- **IDE (Assumed):** VS Code, PyCharm, or IDLE.
- **Core Libraries:** Tkinter & ttk, ttkbootstrap, SQLite3, Pillow, hashlib, os, datetime.
- **Key Algorithms/Logic:**
 - Password Hashing: SHA-256 via hashlib.
 - CRUD Operations: Standard SQL queries via sqlite3.
 - Data Display: ttk.Treeview.
 - Image Handling (ID Card): Pillow for drawing text and pasting images.
 - Dynamic UI Updates, Event Handling, Custom Window Behaviour, Modular Design.

System Testing:

System Testing

Testing phases included:

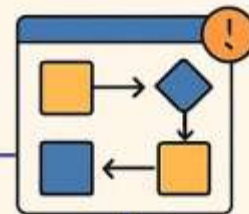
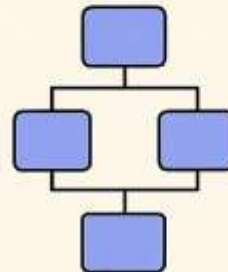
1. Unit Testing (Conceptual & Manual)

Testing individual functions
(DB ops, UI rendering, Input
validation, image loading)



2. Integration Testing (Manual)

Testing interactions between modules
(Login-Main App, Registration-Login,
Student Data-Display, ID Card-Student Data,
Payment-Student Data, DB interactions)



3. System Testing (End-to-End Manual Scenarios)

Scenario 1: New User Onboarding & Student Entry
(Register, Login, Add Student, Generate ID, Record Payment)
Scenario 2: Data Modification and Reporting
(Login, Search, Update Student, Generate Reports)
Scenario 3: Error Handling & Invalid Inputs
(Incorrect login, existing username, missing fields,
non-numeric data, non-existent roll number)



4. Usability Testing (informal)

Checking ease of use, intuitive design,
title bar functions, UI responsiveness
were Implemented based on testing,

Testing phases included:

1. **Unit Testing (Conceptual & Manual):** Testing individual functions (DB ops, UI rendering, input validation, image loading).
2. **Integration Testing (Manual):** Testing interactions between modules (Login-Main App, Registration-Login, Student Data-Display, ID Card-Student Data, Payment-Student Data, DB interactions).
3. **System Testing (End-to-End Manual Scenarios):**
 - Scenario 1: New User Onboarding & Student Entry (Register, Login, Add Student, Generate ID, Record Payment).
 - Scenario 2: Data Modification and Reporting (Login, Search, Update Student, Generate Reports).
 - Scenario 3: Error Handling & Invalid Inputs (Incorrect login, existing username, missing fields, non-numeric data, non-existent roll number).
4. **Usability Testing (Informal):** Checking ease of use, intuitive design, title bar functions, UI responsiveness.

Test Results Overview: Core functionalities performed as required. Minor UI adjustments and error message enhancements were implemented based on testing.

Validation Checks

The system includes:

1. **User Registration & Login:** Required fields, password mismatch, User ID uniqueness, authentication.
 2. **Student Management:** Required fields, numeric fields, date format validation, database-level unique constraints (roll_number, aadhaar_no), foreign key integrity, selection for update/delete.
 3. **ID Card and Receipt Generation:** Roll number existence, positive amount paid.
 4. **Reports and Analytics:** Filter input presence and type.
 5. **Feedback:** Non-empty feedback text.
 6. **File Handling:** try-except for image loading, fallback for missing ID card background.
 7. **General Error Handling:** try-except for database operations (sqlite3.Error).
-

Limitations

1. **Desktop Application Only:** No web/mobile access.
 2. **Single User Environment:** Not designed for concurrent multi-user operations.
 3. **Limited Scalability (Very Large Institutions):** SQLite performance concerns with massive datasets.
 4. **No Advanced Role-Based Access Control (RBAC).**
 5. **Basic Analytics and Reporting:** Text-based, no graphical charts.
 6. **No Automated Backup and Recovery.**
 7. **Image Path Dependency.**
 8. **Manual Dependency Installation** (if not packaged).
 9. **Font Dependencies for ID Card.**
 10. **Limited UI Customisation by End-User.**
-

Future Enhancements

1. **Web-Based Platform** (e.g., using Django/Flask).
 2. **Role-Based Access Control (RBAC)**.
 3. **Advanced Analytics and Dashboards** (graphical charts).
 4. **Data Import/Export** (CSV, PDF, Excel).
 5. **Automated Backup and Restore**.
 6. **Email/Notification System**.
 7. **Enhanced ID Card and Document Management**.
 8. **Audit Trails**.
 9. **Integration with Other Systems** (LMS, accounting).
 10. **Improved Image Management** (relative paths, BLOBs).
 11. **Batch Operations**.
 12. **Multi-Language Support**.
 13. **Packaging as Executable** (PyInstaller, cx_Freeze).
-

Conclusion

The Student Database Management System has been successfully developed as a comprehensive desktop application. Utilising Python, Tkinter, ttkbootstrap, and SQLite, it provides an efficient, user-friendly, centralised solution for managing student records, authentication, academic details, and payments. Key features include ID card and receipt generation, reporting, and basic analytics.

The system effectively addresses the core data management needs of educational institutions. Its modular design supports maintainability and future expansion. While current limitations exist, such as its desktop-only nature and basic analytics, it serves as a valuable tool for small to medium-sized institutions. The outlined future enhancements offer a roadmap for evolving it into an even more powerful and accessible platform, demonstrating the practical application of software development to solve real-world educational administration challenges.

Bibliography

- Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media.
- Summerfield, M. (2008). *Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming*. Prentice Hall.
- Roseman, J. (2000). *Tkinter GUI Application Development Hotshot*. Packt Publishing.
- Shipman, J. W. (2013). *Tkinter 8.5 reference: a GUI for Python*. New Mexico Tech Computer Center.
- Van Rossum, G., & Drake, F. L. (Eds.). (2023). *The Python Language Reference*. Python Software Foundation.
- Python Software¹ Foundation. (2023). *SQLite3 — DB-API 2.0 interface for SQLite databases*.
- Allen, G., & Owens, R. C. (2011). *The Definitive Guide to SQLite* (2nd ed.). Apress.
- Israeli, I. (2020). *ttkbootstrap Documentation*.
- Clark, K., & Garriss, J. (Eds.). (2023). *Pillow (PIL Fork) Documentation*.
- Pressman, R. S., & Maxim, B. R. (2019). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill.
- Elmasri, R., & Navathe, S. B. (2017). *Fundamentals of Database Systems* (7th ed.). Pearson.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley.
- Myers, G. J., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing* (3rd ed.). Wiley.
- W3Schools. (2023). *Python Tutorial*.