**Abstract**

The Student Database Management System is a console program developed in the C programming language for efficient management of student and teacher information. The system supports simple operations such as adding, displaying, updating, searching, and deleting data using structures and file-handling techniques. The system has data persistence by storing data in external files so that data isn't disrupted even after the program execution has ended.

Especially designed to be easy and user-friendly, the system is a helpful and lightweight means of managing scholarly information, especially in small and medium-sized schools. The app contains a role-based access control system with administrator, teachers', and students' modules with their own respective privileges and functionalities. For example, administrators can manage users and see the whole database, while teachers can manage student performance, attendance and assignments.

With the application of modular programming and menu-driven interface, the system enjoys user-friendliness and maintainability. Its text-based user interface gives it platform-independence and ability to work in low-resource environments. Overall, the system presents a simple example of the application of C language to real-world applications in database management.

**Table of Contents**

[1 Introduction 1](#_Toc204515443)

[2 Problem Statement 2](#_Toc204515444)

[3 Objective 3](#_Toc204515445)

[4 Background Study 4](#_Toc204515446)

[5 Requirement Document 5](#_Toc204515447)

[5.1 Functional Requirement 5](#_Toc204515448)

[5.2 Non-Functional Requirements 6](#_Toc204515449)

[6 System Design 7](#_Toc204515450)

[6.1 Feasibility study 7](#_Toc204515451)

[6.2 Algorithm 9](#_Toc204515452)

[6.3 Flowchart 11](#_Toc204515453)

[6.4 Context Diagram 12](#_Toc204515454)

[6.5 Data Flow Diagram 13](#_Toc204515455)

[6.6 Use Case Diagram 17](#_Toc204515456)

[7 Development 18](#_Toc204515457)

[7.1 Implementation 18](#_Toc204515458)

[7.2 Methodology 19](#_Toc204515459)

[7.3 Gantt Chart 21](#_Toc204515460)

[8 Testing 23](#_Toc204515461)

[9 Project Result 26](#_Toc204515462)

[10 Future Enhancements 27](#_Toc204515463)

[11 Conclusion 28](#_Toc204515464)

[12 Reference 29](#_Toc204515465)

[13 Annex 31](#_Toc204515466)

**List of Figures**

[Figure 1 Flowchart Main Function 11](#_Toc203295163)

[Figure 2 Context Diagram 12](#_Toc203295164)

[Figure 3 Level 0 DFD 13](#_Toc203295165)

[Figure 4 Level 1 DFD 13](#_Toc203295166)

[Figure 5 Level 2 DFD Login Function 14](#_Toc203295167)

[Figure 6 Level 2 DFD Teacher Dashboard 14](#_Toc203295168)

[Figure 7 Level 2 DFD Student Dashboard 15](#_Toc203295169)

[Figure 8 Level 2 DFD Assignment Management 15](#_Toc203295170)

[Figure 9 Level 2 DFD Admin Dashboard 16](#_Toc203295171)

[Figure 10 Use Case Diagram 17](#_Toc203295172)

[Figure 11Waterfall Model 19](#_Toc203295173)

[Figure 12 Gantt Chart 21](#_Toc203295174)

**List of Tables**

[Test Case 1 23](#_Toc203295205)

[Test Case 2 23](#_Toc203295206)

[Test Case 3 24](#_Toc203295207)

[Test Case 4 24](#_Toc203295208)

[Test Case 5 25](#_Toc203295209)

**List of Abbreviations**

|  |  |
| --- | --- |
| DFD | Data Flow Diagram |
| GPA | Grade Point Average |
| GUI | Graphical User Interface |
| SDMS | Student Database Management System |
| UI | User Interface |

# Introduction

The design and implementation of a comprehensive student information system and user interface is to replace the current paper records. It is a centralized software application that helps streamline the management of student data, reduce administrative workload, and improve accuracy and accessibility of information. The system aims to reduce manual paperwork, improve accuracy, and streamline academic processes such as student enrollment, teacher management, attendance and assignment tracking. By providing a centralized platform, it enhances communication, data security, and overall productivity in managing academic records. This study makes a notable contribution in the realm of student management systems by proposing the development of an effective and budget-friendly solution through the incorporation of a programmable device programming system.

Furthermore, it supports automation of routine work, eliminates the chance of losing data, and offers immediate access to academic information by the stakeholders, hence supporting decision-making and institutional performance. The system is expandable, user-friendly, and applicable to various educational environments. Its systematic mechanism for storing, retrieving, and manipulating data not only increases transparency but also supports academic institutions with a reliable information infrastructure that is sustainable.

# Problem Statement

In many educational institutions, student data such as personal information, course details, attendance, and academic performance is still managed manually or using basic tools like spreadsheets. It also lacks proper data security, quick accessibility, and centralized control. Therefore, there is a need for an automated Student Database Management System that can securely store, manage, and retrieve student information efficiently and accurately.

* Difficulty to search record when there is no computerized system there is always difficulty in searching of records if the records are large in number. [(p. 9)](#Jaryal)
* Manual student data management is time-consuming and inefficient because it requires a lot of human effort to record and update information. This slows down administrative tasks and causes delays in accessing student records.

# Objective

The objective of Student database management system is to allow the administrator of any organization to edit and find out the personal details of a student  [(p. 3)](#Kumar). This system is intended to reduce the burden of manual work, improve accuracy, and provide real-time access to data for students, faculty, and administrators. It focuses on enhancing efficiency in handling student enrollment, course registration, grading, and assignment through a centralized and secure platform.

* To allow users to easily add, update, or delete student and course records, keeping the database current and avoiding confusion caused by outdated information.
* To improve data accuracy, eliminate redundancy, and avoid manual errors by validating data and reducing duplication, thereby ensuring reliable and clean records.

# Background Study

In today’s digital era, the management of student information plays a critical role in the operation and administration of educational institutions. Traditionally, student records were managed using paper-based systems or spreadsheets, which are prone to errors, difficult to update, and lack security. With the rapid advancement of technology, especially in computing and database systems, institutions now aim to implement automated solutions to enhance accuracy, efficiency, and accessibility.

A Student Database Management System (SDMS) is software designed to store, manage, and retrieve student-related information such as personal details, academic performance, assignments, attendance and more. It can be used by any educational institute or colleges to maintain records of students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information can be very time consuming. All these problems are solved using this project [(p. 6)](#Divya)

Additionally, Student Database Management System not only automates record-keeping but also improves communication and decision-making for students, faculty, and administrators. Real-time access to data, role-based access control, and integration with other institutional systems (such as attendance, grading, or course management) enable stakeholders to make knowledgeable decisions promptly. SDMS also has the ability to provide insightful reports and analytics, which assist teachers in monitoring student progress, pinpointing areas of improvement, and designing learning strategies accordingly. As schools and universities move forward with digital transformation, the adoption of a comprehensive SDMS becomes instrumental in attaining operational excellence and enabling student success.

# Requirement Document

We have collected several requirements for the project from our initial research, website visits, and interviews with the personnel concerned, as well as their experiences regarding the concept of its development. Here, we analyzed, documented, validated, and managed the software or system requirements.

## Functional Requirement

Functional requirements describe the core features of the system. They tell what the system should do based on user needs.

* User Authentication: The system allows different types of users (Admin, Teacher, Student) to log in with valid credentials.
* Add Student: Ability to add new student records, including details like name, roll number, class, and course.
* View Students: Ability to display a list of all students with their respective details.
* Edit Student: It allows modification of existing student records based on student ID or roll number.
* Delete Student: Enable removal of student records from the database.
* Teacher Management: Admin can add, view, edit, and delete teacher records.
* Password Management: Users can change their passwords securely.
* Data Persistence: All changes are saved persistently to a file or database to ensure data is retained between sessions.

## Non-Functional Requirements

Non-functional requirements focus on how the system performs its functions. They are related to performance, usability, reliability, etc.

* Usability: The system provides a user-friendly console interface that is easy to navigate for all user types.
* Performance: The system handles data operations quickly without noticeable delays.
* Reliability: Data operations such as add, edit, and deleting are performed without errors and data loss.
* Security: User authentication and password management prevent unauthorized access.
* Portability: The system is compatible and executable on any platform supporting the C programming environment.
* Maintainability: The code is modular and well-documented to facilitate future updates and debugging.

# System Design

System Design is the one of phases in the developing of system. It is the process of designing the elements of a system such as modules, interfaces and the data that goes through that system.

## Feasibility study

Whatever we think may not necessarily be feasible. It is wise to consider the feasibility of any problem we undertake. Feasibility is the study of the impact caused by the development of a system within the organization. This impact can be either positive or negative. When the positive impacts outweigh the negatives, the system is considered feasible. [(p. 1)](#Acharya)

The feasibility study can be performed in five ways:

1. Technical **Feasibility**

The proposed system is technically feasible. All required hardware, software, and human resources are already available within the organization. There are no significant obstacles in terms of technology or tools needed for both the development and maintenance of the system. The organization is leveraging its existing infrastructure effectively. The technical evaluation provides information on issues including whether the system's required technology is available, how challenging its construction will be, and if the company has sufficient experience with it. [(p. 7)](#praasd)

1. Economic Feasibility

Economic feasibility is also referred to as cost/benefit analysis. It is the most frequently used method for evaluating the effectiveness of a new system. [(p. 10)](#Kamal) The environment for building the system can be prepared with low investment and efficient supervision. The return on investment (ROI) is expected to be high as the system improves efficiency with minimal cost.

1. Operational Feasibility

Operational feasibility considers whether the system will function effectively in the intended environment and be accepted by its users. The system is designed with a user-friendly, menu-driven interface that is easy to operate for all types of users admin, teachers, and students. It improves the accuracy of data handling, enhances accessibility, and increases productivity. Because of its simplicity and effectiveness, the system is operationally feasible and aligns well with the institution’s daily operations.

1. Schedule Feasibility

Schedule feasibility addresses whether the project can be completed within the given time frame. The project followed a well-planned Gantt chart that defined clear timelines for each development phase, including problem identification, requirement analysis, design, coding, testing, and documentation. Each task was carried out on schedule, and the overall project was completed within the allotted academic period. Therefore, the project is considered scheduled feasible.

1. Legal Feasibility

Legal feasibility involves ensuring that the system complies with all legal and ethical standards. The SDMS project does not collect sensitive or confidential data and adheres to institutional and academic guidelines. It uses only legally licensed or open-source tools, which ensure compliance with software usage policies. Hence, the system does not face any legal or ethical issues.

## Algorithm

Step 1: Start the program.

Step 2: Load data from files into memory (e.g., students, teachers, admin info).

Step 3: Call the loadData() function to initialize data structures.

Step 4: Enter the main menu loop (repeat until the user selects Exit).

Step 5: Clear the screen.

Step 6: Display the main menu options:

Admin Login

Teacher Login

Student Login

Exit

Step 7: Get user input for menu choice.

Step 8: Switch based on user choice:

If Admin Login:

Step 8.1: Prompt the user to enter admin username and password.

Step 8.2: If credentials match:

Call the Admin Menu.

Step 8.3: Else:

Show "Invalid credentials" message.

Wait for a key press to return to the main menu.

If Teacher Login:

Step 8.4: Prompt for Teacher ID and password.

Step 8.5: Search the teachers array for matching ID and password.

Step 8.6: If found:

Call the Teacher Menu.

Step 8.7: Else:

Show "Invalid credentials" message.

Wait for key press.

If Student Login:

Step 8.8: Prompt for Student Roll number and password.

Step 8.9: Search the students array for matching roll and password.

Step 8.10: If found:

Call the Student Menu.

Step 8.11: Else:

Show "Invalid credentials" message.

Wait for key press.

If Exit:

Step 8.12: Call saveData() to save all data to file.

Step 8.13: Exit the program.

Default (Invalid Choice):

Step 8.14: Show "Invalid choice" message.

Step 8.15: Wait for key press.

Step 9: Repeat from Step 4 unless the user exits.

Step 10: End the program.

## Flowchart

A flowchart is a visual diagram that represents the sequence of steps or decisions needed to perform a process or solve a problem. It uses standardized symbols like arrows, rectangles, diamonds, and ovals to show the flow of control from start to finish.

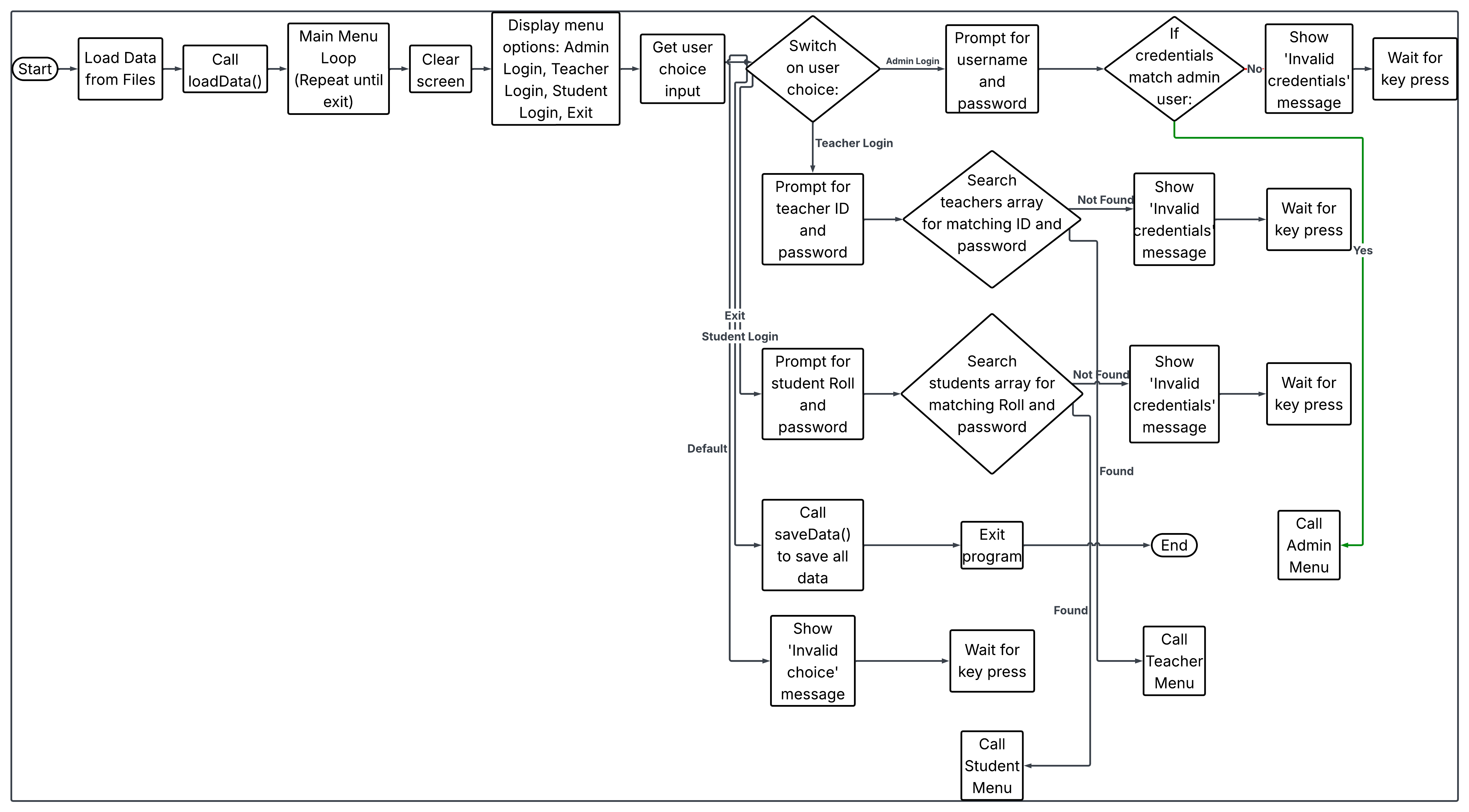


Figure 1 Flowchart Main Function

## Context Diagram

A diagram of a student database management system

AI-generated content may be incorrect.

Figure 2 Context Diagram

## Data Flow Diagram

A diagram of a student database management system

AI-generated content may be incorrect.

Figure 3 Level 0 DFD

A diagram of a program

AI-generated content may be incorrect.

Figure 4 Level 1 DFD

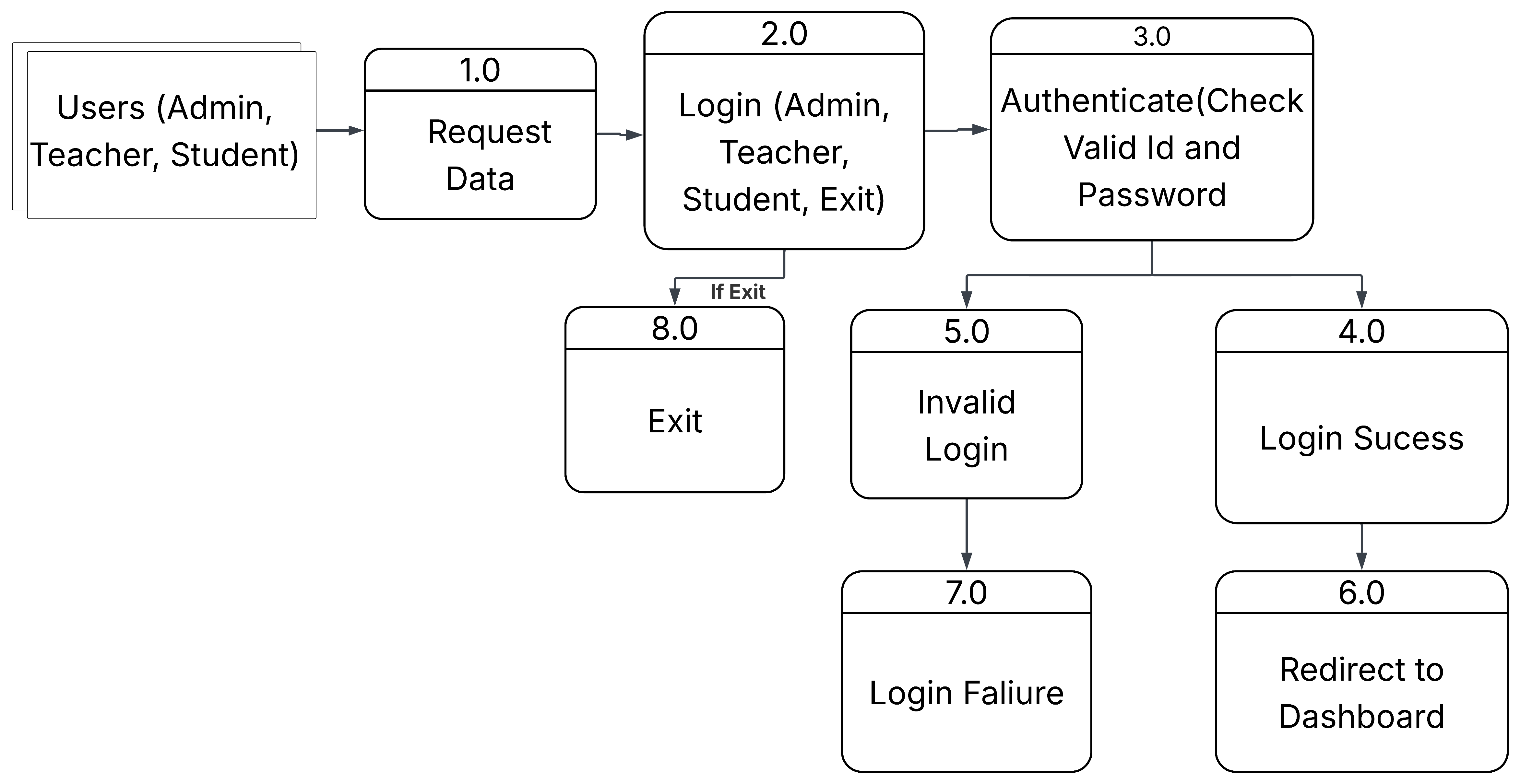


Figure 5 Level 2 DFD Login Function

A diagram of a teacher

AI-generated content may be incorrect.

Figure 6 Level 2 DFD Teacher Dashboard

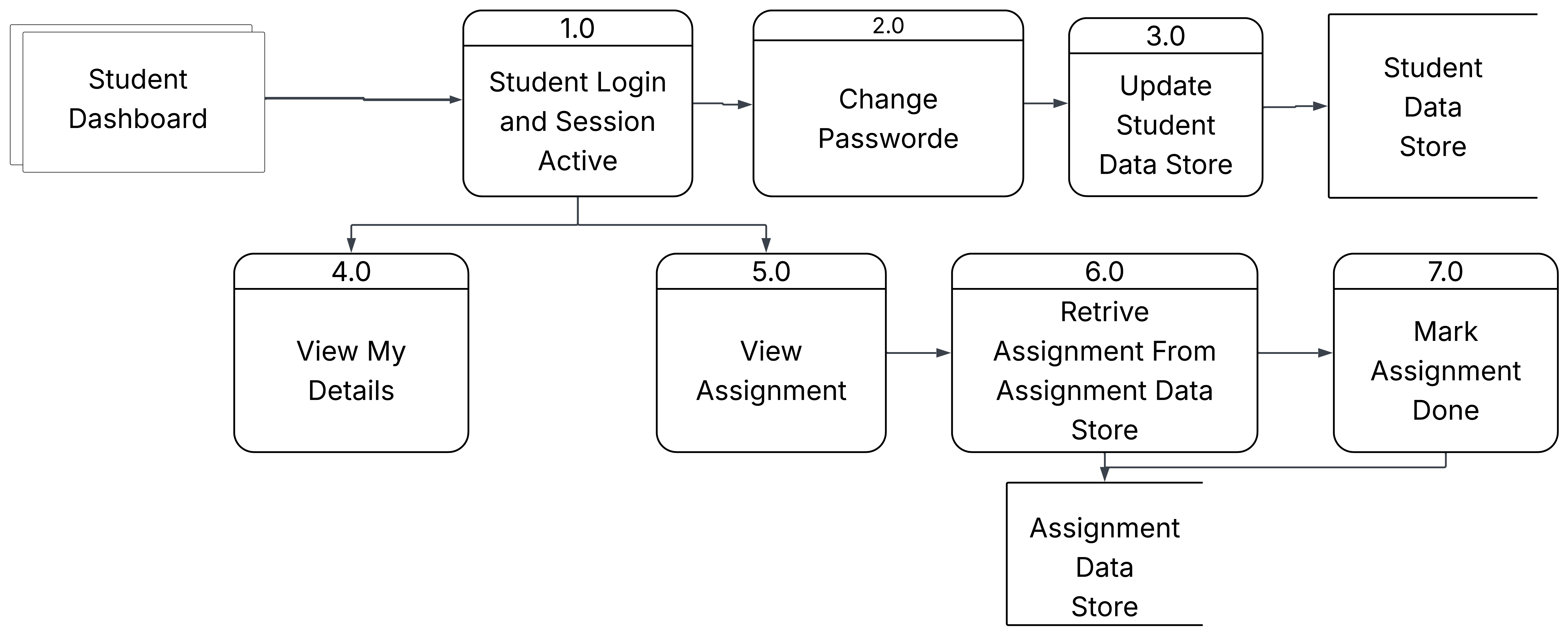


Figure 7 Level 2 DFD Student Dashboard

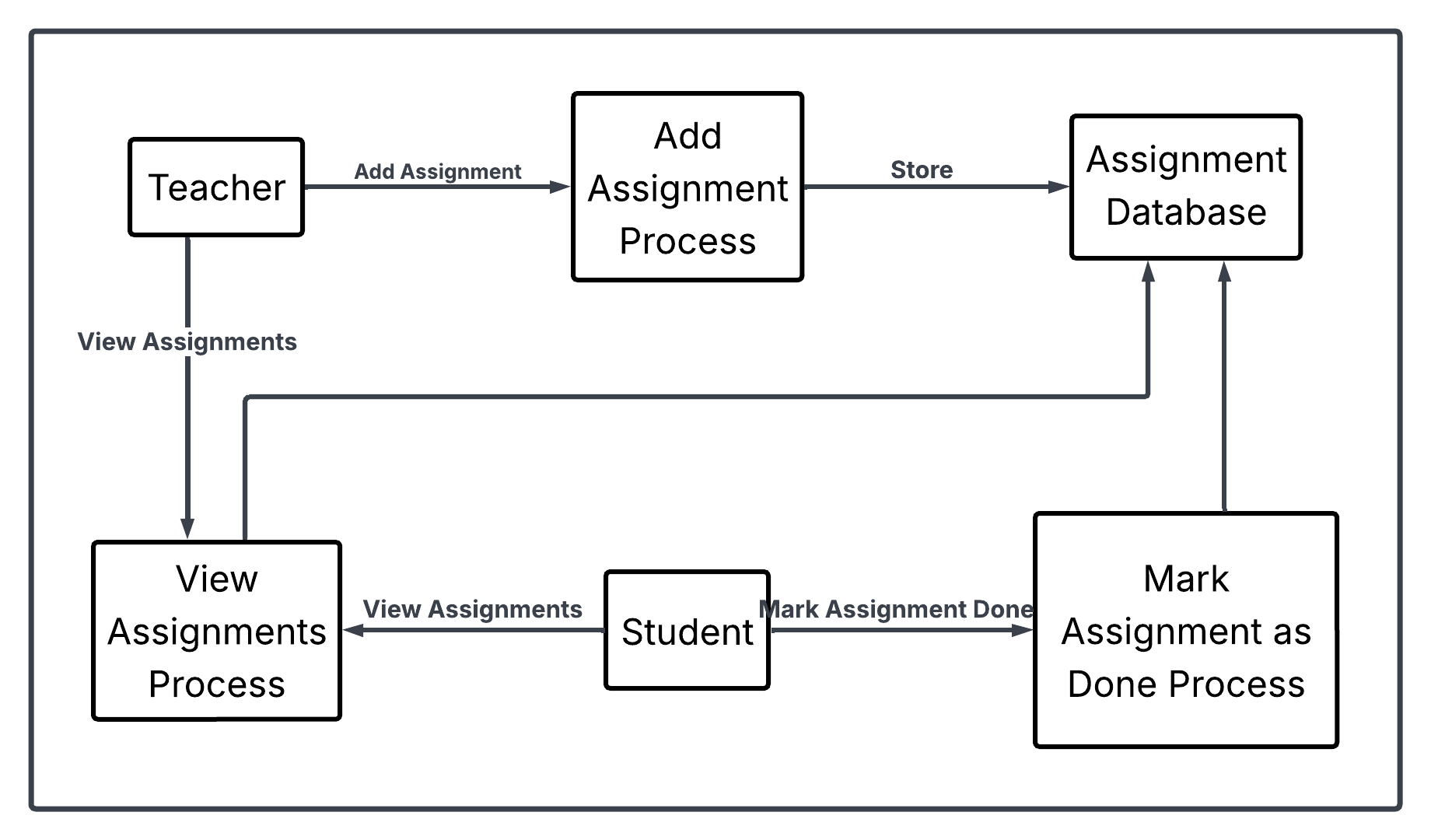


Figure 8 Level 2 DFD Assignment Management

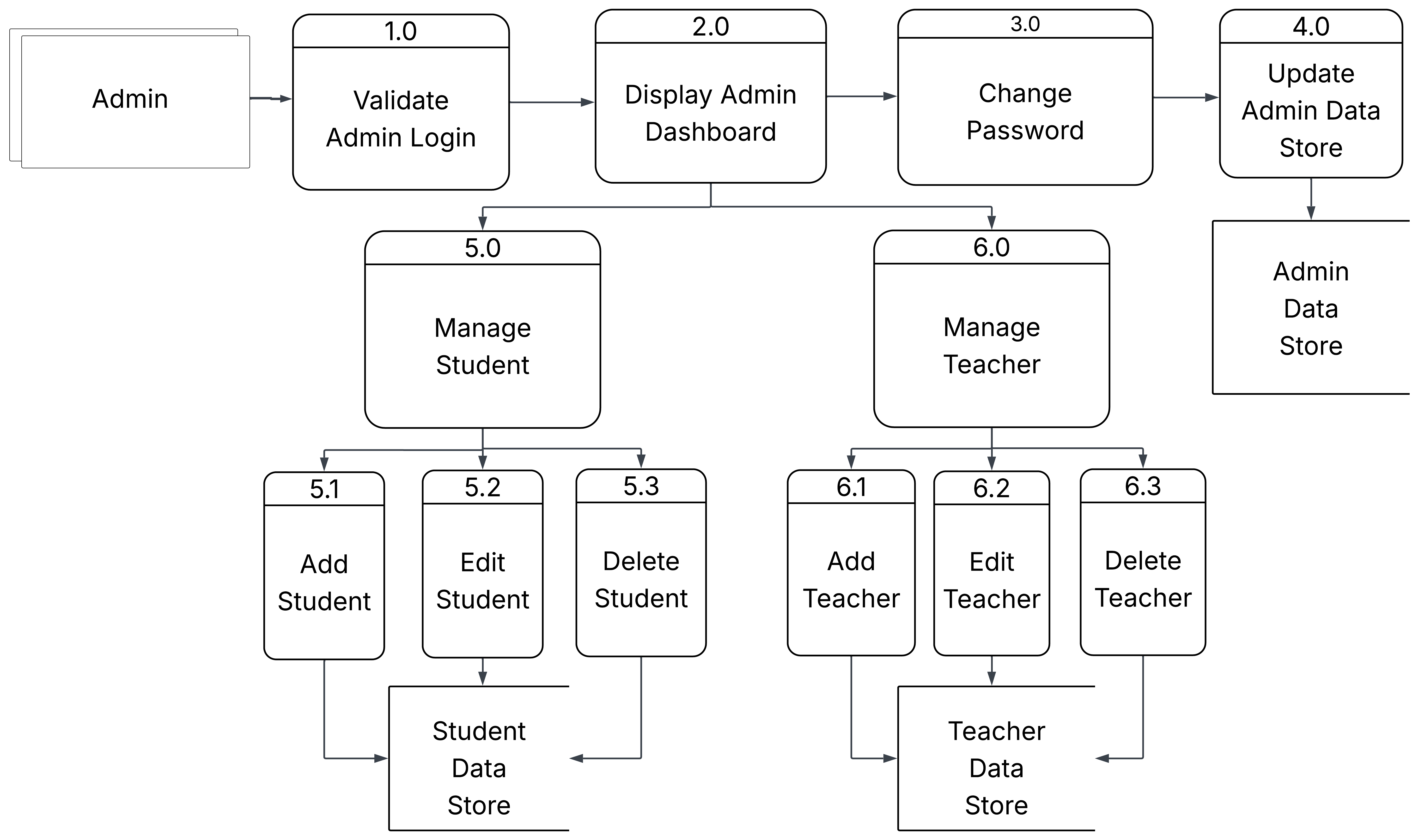


Figure 9 Level 2 DFD Admin Dashboard

## Use Case Diagram

A diagram of a teacher

AI-generated content may be incorrect.

Figure 10 Use Case Diagram

# Development

## Implementation

1. Development Environment

The Student Database Management System was developed in the C programming language using a console-based interface. The development environment included the following tools and technologies:

* Language: C
* Compiler: GCC (GNU Compiler Collection)
* IDE/Editor: Code::Blocks / Dev C++ / VS Code
* Operating System: Windows 10
* File Handling: Text files (.txt) were used for persistent data storage.
* Version Control: Manual backups

This environment was chosen for its lightweight requirements and compatibility with

procedural programming principles.

1. Operation Procedure

The system follows a modular and menu-driven approach to perform various operations. The execution begins with the login screen, offering role-based access for:

* Administrator: Can add/view/edit/delete students and teachers, manage passwords, and oversee the system.
* Teacher: Can view student data, update marks, and provide feedback.
* Student: Can view personal details, performance records, and assignment status.

Each module is implemented as a separate function, allowing for code reusability and

easier maintenance. File handling is used extensively to read and write data to external

storage, ensuring that information persists between sessions.

## Methodology

Waterfall Model is a software development methodology that was first introduced by Winston W. Royce in 1970. It is a linear and sequential approach to software development that consists of several phases that must be completed in a specific order. [(p. 2)](#Luettavissa)

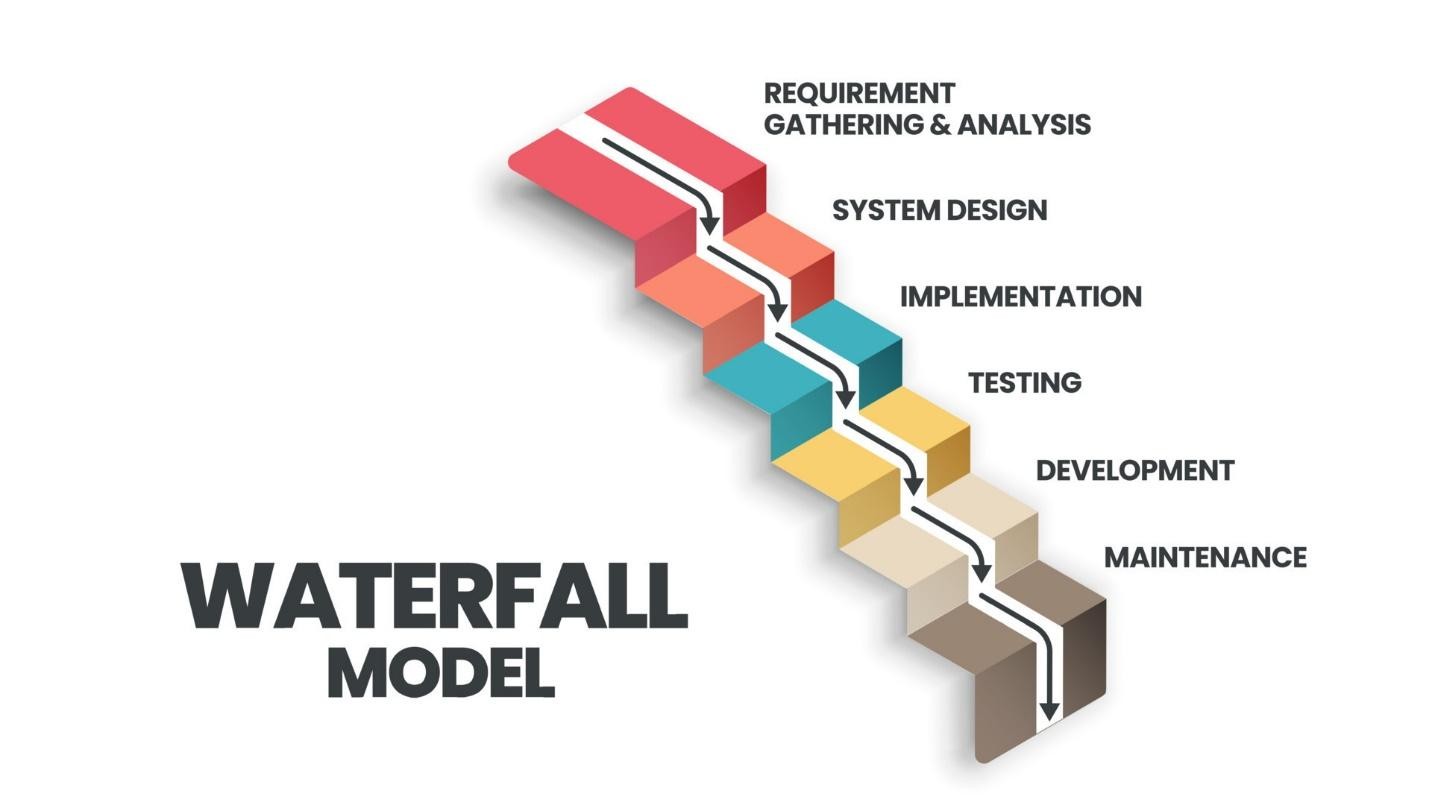


Figure 11Waterfall Model

1. Requirement Gathering and Analysis

The planning stage determines the objectives of the project and whether the project should be given the green light to proceed. This is where the proposal submission comes into picture. [(p. 8)](#Ajay) The proposal outlines the goals, scope, and resources required, providing a clear roadmap for the project.

1. System Design

There is then the System Design phase, at which point the developers define how to create the system based on the requirements. It has two levels of design: high-level design (system architecture, module design) and low-level design (detailed logic, database schemas, data flow diagrams). This phrase translates the "what" of the requirements into the "how" of the system structure.

1. Implementation

Implementation follows the design phase where the developers start coding the actual code according to the design specifications. This is the phase where the system is developed using programming languages and tools. The coding, integration, and preparation of all modules for testing are accomplished.

1. Testing

After the system is developed, it enters the Testing stage. The Testing stage ensures that the software is working properly and free from bugs. Various testing methods like unit testing, integration testing, system testing, and user acceptance testing are done. Issues identified are reported and fixed before the system is alive.

1. Deployment

Once testing is successful, the software is deployed to the production environment. Users start using the system, and training or documentation may be provided.

1. Maintenance

The final stage is Maintenance, which commences as soon as the software is delivered and installed. Here, the software is monitored and maintained over a period to facilitate its future functioning. It involves error correction that was missing during earlier stages, improving performance, and changing the system as and when user requirements change.

Briefly, the Waterfall Model is best suited for SDMS, whose requirement is well established and has a clear, fixed sequence of steps. Its simplicity and structured process make it easy to manage, especially for simple or small systems

## Gantt Chart

A graph with orange rectangles

AI-generated content may be incorrect.

Figure 12 Gantt Chart

This Gantt chart outlines the phases of a software development project from late April to the end of June 2025. Here's a breakdown of each phase and its timeline:

1. Problem Identification (April 28, 2025 – May 4, 2025)

We were able to identify the key issues and project scope within a week, which allowed us to establish a clear direction from the beginning.

1. Requirement Analysis (May 4, 2025 – May 12, 2025)

We've put more time into meticulously gathering and analyzing all the system requirements. While doing this, we had rigorous discussions, collected feedback from potential users (Admin, Teacher, Student), and recorded both functional and non-functional requirements. This meticulous and thorough process helped us avoid being vague and ensure that system design would be a true reflection of user expectations. We've also mapped features extensively to prioritize functionality needed at the first instance, accelerating the subsequent steps.

1. Design (May 12, 2025 – May 16, 2025)

We have made system architecture and data flow models within four days, creating precise context diagrams to describe relationships and processes.

1. Coding (May 25, 2025 – May 25, 2025):

We have carried out coding within ten days by following the structured design and modular programming methods for rapid development and easy debugging.

1. Testing and Debugging (May 25, 2025 – June 3, 2025):

We've devoted enough time to strenuous testing and debugging over nine days. This involved unit testing every module to check every component worked as expected integration testing to ensure the modules collaborated without a glitch, and system testing for execution. We documented all test cases, ran some test cycles, and fixed bugs successfully. This extended testing duration allowed us to identify potential issues and improve system stability, thus reducing the likelihood of implementation and maintenance faults.

1. Implementation (June 3, 2025 – June 9, 2025):

We were able to implement the system successfully within a week, preparing it for use ahead of time.

1. Maintenance (June 9, 2025 – June 13, 2025):

We were able to address minor issues instantly and have the system ready to be executed continuously.

1. Documentation (April 28, 2025 – June 13, 2025):

We have retained documentation of the development of the life cycle in its original form, with every stage sufficiently documented and understandable.

# Testing

Title: Admin Login Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Id | Module | Test Description | Input Data | Expected Result | Status  (Pass/Fail) |
| TC 001 | Admin | Login with valid credentials | Username: admin, Password: correct | Login successful, admin menu displayed | Pass |
| TC 002 | Admin | Login with invalid credentials | Username: admin, Password: wrong | Login failed, error message shown | Fail |

Test Case 1

Title: Admin Student Management Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Id | Module | Test Description | Input Data | Expected Result | Status  (Pass/Fail) |
| TC 003 | Admin | Add new students | Valid student details | Student added, confirmation message | Pass |
| TC 004 | Admin | Add student with existing roll number | Roll: existing roll | Error message: "Student already exists" | Fail |

Test Case 2

Title: Admin Password Management Tests.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Id | Module | Test Description | Input Data | Expected Result | Status  (Pass/Fail) |
| TC 005 | Admin | Change admin password with correct old pwd | Old pwd: correct, New pwd: newpwd | Password changed confirmation | Pass |
| TC 006 | Admin | Change admin password with wrong old pwd | Old pwd:  Wrong | Error message: "Incorrect current password" | Fail |

Test Case 3

Title: Teacher Dashboard Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Id | Module | Test Description | Input Data | Expected Result | Status  (Pass/Fail) |
| TC 007 | Teacher | Login with valid credentials | Login with valid credentials | Login successful, teacher menu displayed | Pass |
| TC 008 | Teacher | Add assignment | Valid assignment details | Assignment added confirmation | Pass |
| TC 009 | Teacher | View students | N/A | List of students displayed | Pass |

Test Case 4

Title: Student Dashboard Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Id | Module | Test Description | Input Data | Expected Result | Status  (Pass/Fail) |
| TC 010 | Student | Login with valid credentials | Roll: 1, Password: pwd | Login successful, student menu displayed | Pass |
| TC 011 | Student | View own details | N/A | Student details displayed | Pass |
| TC 012 | Student | Mark assignment as done | Assignment  ID:1 | Assignment marked as completed confirmation | Pass |
| TC 013 | Student | Mark assignment as done  (Invalid ID) | Assignment  ID:7 | Error message: “Assignment not found or not assigned to this student” | Fail |

Test Case 5

# Project Result

We were successful in creating and deploying an integrated **Student Database Management System** that streamlines the management of academic data for a school. The system supports three major user roles—**Admin, Teacher, and Student**—each with distinct features to ensure proper and secure data handling.

As Admins, we implemented a secure login mechanism that allows us to maintain all teacher and student records effectively. We developed functionalities to add new students and teachers, edit their information, and prevent duplicate entries by validating distinct identifiers such as roll numbers. This helped reduce manual errors and ensured accurate, centralized records. We also included a password management feature to enhance system security by enabling password changes with appropriate authentication.

For the Teacher role, we developed modules that allow teachers to securely log in and manage assignments. Teachers can create new assignments, view the student list, and monitor assignment submissions. These features enhance communication between teachers and students, and support better academic progress tracking.

Under the Student module, we provided functionality for students to log in using their roll number and password to view their personal details and academic information. Students can also view their assignments and mark them as completed, promoting active engagement in coursework. The system handles incorrect inputs gracefully to ensure a smooth user experience.

We conducted extensive testing and debugging during the project to ensure that each module functions as intended, offering a stable and reliable system. Through accurate requirement analysis and comprehensive testing, we were able to complete the project ahead of schedule. We also ensured proper and detailed documentation, which will aid in future maintenance and potential enhancements.

**Screenshots** illustrating each feature have been included in the **Annex section** for reference.

# Future Enhancements

We plan to enhance the Student Database Management System by using a database to improve data management, creating a user-friendly interface, and adding features to improve communication and security. The following improvements can be considered for upcoming versions of the system:

1. Graphical User Interface (GUI)

Transitioning from a console-based interface to a graphical user interface would make the system more user-friendly and visually appealing. This would improve usability for less technically inclined users, especially students.

1. Database Integration

Instead of relying on file handling for data storage, future versions could incorporate a relational database system (e.g., MySQL or SQLite) to handle larger volumes of data more efficiently and provide better data security and retrieval speed.

1. Multi-user Support

Adding simultaneous access support with proper role-based permissions (Admin, Teacher, Student) would make the system scalable in multi-user environments such as college labs or networks.

1. Assignment Notifications

The system could include automatic reminders or notifications for students regarding assignment deadlines and submission statuses via email or SMS.

1. Report Generation.

The system could provide automated generations of academic reports, attendance summaries, and grade charts in PDF or printable formats for both students and administrators.

# Conclusion

In conclusion, the development of Student Database Management System in C language has attained a robust, effective, and user-friendly interface that effectively manages the student's academic information, teachers' information, and assignments with enhanced security and convenience. The modular nature of the system meets the specific needs of administrators, teachers, and students by providing distinct features such as record maintenance, assignment tracking, and secure password management to ensure administrative activities are smooth and overall data accuracy is increased. By automating such routine processes like adding, editing, viewing, deleting records and assignments, and managing user authentications, the system reduces manual mistakes and administrative burden while maximizing open communication within the academic community. The availability of persistent data storage ensures that all such critical information is always stored and retrieved across sessions, ensuring long-term data integrity. Furthermore, the project was implemented on schedule with proper documentation, laying a strong groundwork for future enhancements such as additional user roles, advanced reporting, and integration with other institutional systems. Overall, this system is an asset that can significantly improve academic management processes, improve decision-making, and make a positive impact on the institution's educational mission.

# Reference

Divya.T and Lalitha Lakshmi.D. (2022, April 3). *STUDENT DATABASE MANAGEMENT SYSTEM IN C++*. Retrieved from Sathyabama: https://sist.sathyabama.ac.in/sist\_naac/documents/1.3.4/1922-b.sc-cs-batchno-12.pdf

Acharya, K. (2023, December 10). *Chapter 1 INTRODUCTION OF STUDENT INFORMATION MANAGEMENT SYSTEM*. Retrieved from ResearchGate: https://www.researchgate.net/publication/370593698\_Chapter\_1\_INTRODUCTION\_OF\_STUDENT\_INFORMATION\_MANAGEMENT\_SYSTEM

Acharya, K. (2024, October 31). *Student Information Management System Project Report II*. Retrieved from researchgate: https://www.researchgate.net/publication/385440976\_Student\_Information\_Management\_System\_Project\_Report\_II

Ajay Shankar Bidyarthy And Abhishek Kumar. (2012, November 13). *Student Database Management System*. Retrieved from academia: https://d1wqtxts1xzle7.cloudfront.net/54663279/-Student-DB-Mgmt-System-libre.pdf?1507532781=&response-content-disposition=inline%3B+filename%3DStudent\_Database\_Management\_System.pdf&Expires=1751361028&Signature=UQjX4WqFwWdrrINDMp9PUF27vWANL62pf2Y1ifLCnxsr

Divya Shree Yanamandra, Dr. G.N.R, Praasd. (2022, July 7). *A FEASIBILITY STUDY FOR THE STUDENT PERFORMANCE PREDICTION USING MACHINE LEARNING*. Retrieved from ResearchGate: https://www.researchgate.net/publication/362393382\_A\_FEASIBILITY\_STUDY\_FOR\_THE\_STUDENT\_PERFORMANCE\_PREDICTION\_USING\_MACHINE\_LEARNING

Huijin Huang and Benyuan Li . (2024, May 24). *Design and implementation of student management system of integrated programmable device programming system*. Retrieved from nature.com: https://www.nature.com/articles/s41598-024-62844-z

Jaryal, V. S. (2022, May 27). *Student Management System* . Retrieved from LPU: http://ir.juit.ac.in:8080/jspui/bitstream/123456789/3798/1/Student%20Management%20System.pdf

Luettavissa. (2015, March 3). *Waterfall model*. Retrieved from topperworld.in: http://www. waterfall-model. com

R. Kumar, A. Chandana Priya, U. Ramya Sree, S. Eswar Sai, S. Sravanth Kumar, K. Naveen. (2023, January 2). *Student Database Management System*. Retrieved from ijarsct: https://ijarsct.co.in/Paper8020.pdf

S.R.Bharamagoudar, Geeta R.B and S.G.Totad. (2013, June 6). *Web Based Student Information Management*. Retrieved from academia.edu: https://d1wqtxts1xzle7.cloudfront.net/56215851/4-shobha\_bharamaoudar-WEB\_BASED\_STUDENT\_INFORMATION-libre.pdf?1522667755=&response-content-disposition=inline%3B+filename%3DWeb\_Based\_Student\_Information\_Management.pdf&Expires=1751188235&Signature=DCfu2TH7HA

# Annex

Title: Admin Login Page.

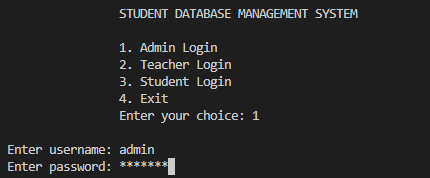


Figure A 1

Title: Admin Dashboard.

A screenshot of a computer

AI-generated content may be incorrect.

Figure A 2

Title: Add Student.

A screen shot of a computer

AI-generated content may be incorrect.

Figure A 3

Title: View All Student.

A black screen with white text

AI-generated content may be incorrect.

Figure A 4

Title: Add Teacher.

A screen shot of a computer

AI-generated content may be incorrect.

Figure A 5

Title: Teacher Dashboard.

A screenshot of a computer

AI-generated content may be incorrect.

Figure A 6

Title: Add Assignment.

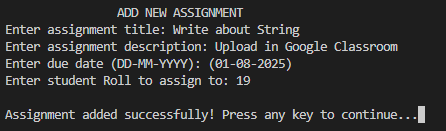


Figure A 7

Title: View Assignment.

A screenshot of a computer

AI-generated content may be incorrect.

Figure A 8

Title: Change Password.

A screen shot of a computer screen

AI-generated content may be incorrect.

Figure A 9

Title: Student Dashboard.

A screen shot of a computer

AI-generated content may be incorrect.

Figure A 10

Title: Mark Assignment as Done.

A screenshot of a computer error

AI-generated content may be incorrect.

Figure A 11

Title: Delete Student.

A computer screen with white text

AI-generated content may be incorrect.

Figure A 12

Title: Edit Student.

A screen shot of a computer

AI-generated content may be incorrect.

Figure A 13

Title: View Own Details.

A screenshot of a computer

AI-generated content may be incorrect.

Figure A 14

Title: Record Attendance

A screen shot of a black screen

AI-generated content may be incorrect.

Figure A 15