

Candidate Name: Rebekah T Mushawatu

Candidate Number: 5025

Center Name: Bradford Senior School

Center Number: 010034

Subject Name: Computer Science

Subject Code: 6023

System Name: School Management System

Year: 2025

## Table of Contents

SECTION A: Investigation and Analysis .....	4
1.0 Introduction .....	4
1.1 Background of the Study .....	4
1.2 Investigation of the Current System .....	4
1.3 Evidence of Research .....	5
1.4 Problems with the Current System .....	8
1.5 Feasibility Study .....	8
1.6 Specification Requirements of the New System .....	9
1.7 Aims and Objectives .....	10
SECTION B: System Design .....	11
2.1 System Modules and Forms .....	11
2.4 Storage and Data Structures .....	17
2.5 Test Plan .....	19
2.6 Database Screenshots .....	20
SECTION C: Software Development .....	22
3.0 Pseudocodes .....	22
3.1 Flowcharts for Modules .....	30
3.2 Code Listing .....	34
3.3 Installation .....	44
3.4 Starting the System .....	45
3.5 Exiting the System .....	48
3.6 User Documentation .....	48
3.6.1 System Overview .....	48
3.6.2 Quick Start Guide .....	49
3.6.3 Module-by-Module Guide .....	53
3.6.4 Troubleshooting .....	55
SECTION D: Testing and Evaluation .....	56
4.0 User Testing .....	56

4.1 Achievements .....	59
4.2 Limitations .....	59
4.3 Delimitations .....	59
4.4 Opportunities for Future Development.....	60
SECTION E: Implementation and Deployment.....	61
5.0 Deployment Plan.....	61
5.1 User Training and Support Plan.....	62
5.2 Maintenance Procedures .....	62
SECTION F: Review and Evaluation.....	63
6.0 Evaluation Criteria.....	63
6.1 Results of Evaluation .....	64
6.2 Recommendations for Improvements.....	64
6.3 Appendices .....	65

## SECTION A: Investigation and Analysis

### 1.0 Introduction

This project aims to develop a robust, efficient, and user-friendly **School Management System**. The rationale behind this project is to address the inefficiencies and inaccuracies inherent in traditional, manual school management practices. A computerized system will automate critical administrative tasks, improve data accuracy, and provide timely access to information for management, staff, and parents.

The importance of a modern School Management System cannot be overstated. It serves as the digital backbone of the institution, streamlining everything from student enrollment and attendance tracking to fee collection and academic reporting. By centralizing data, it provides a single source of truth, eliminating data duplication and human error. This systematic approach is crucial for enhancing operational efficiency and fostering a data-driven culture within the school administration.

### 1.1 Background of the Study

The current educational landscape demands a more organized approach to data management. In many schools, student and staff data is managed through manual files, ledger books, and fragmented spreadsheets. These semi-automated practices, while familiar, are time-consuming and prone to inconsistencies. Staff members spend valuable hours on repetitive data entry and searching for physical records, which could be better spent on core administrative or teaching duties. This project seeks to replace these cumbersome processes with a streamlined, digital solution.

### 1.2 Investigation of the Current System

The existing system, as observed at Bradford Senior School, is predominantly manual. Student records are kept in physical files, and attendance is recorded on paper registers. Financial records, including fee payments, are maintained in large ledger books. Some teachers may use personal spreadsheets to track student marks, but this data is not integrated with the main school records.

**Strengths of the Current System:** The main strengths are its simplicity and familiarity. Staff members are comfortable with the process, and there is no need for specialized technical training. The system is also low-cost in terms of initial setup.

**Weaknesses of the Existing System:**

- **Inefficiency and Delays:** Manual data entry is slow. Compiling reports, such as a student list or fee balances, can take hours or even days, leading to significant delays in administrative processes.

- **Data Errors:** Human error during manual data entry and transcription is common, leading to inaccuracies in student records and financial reports. This can cause discrepancies in billing and student information.
- **Lack of Reporting:** Generating comprehensive reports (e.g., student performance trends) is extremely difficult, if not impossible. The manual system lacks the tools to aggregate and analyze data effectively for strategic decision-making.
- **Security Risks:** Physical files are susceptible to damage, loss, or unauthorized access, posing a significant risk to the security and confidentiality of sensitive student and staff data.

### 1.3 Evidence of Research

To gather comprehensive data, a mixed-method research approach was employed:

- **Questionnaires:** Distributed to administrators and students to gauge their daily tasks, pain points with the current system, and expectations for a new system.
- **Interviews:** Conducted with management and key personnel to understand the strategic and operational needs of the institution.
- **Analysis of Findings:** The research confirmed the widespread frustration with the existing manual system, a high demand for automated reporting, and a clear need for a centralized database to improve data management and accessibility.

#### Sample Questionnaire: Administrator

This questionnaire is designed to gather quantitative data on the daily tasks, challenges, and requirements of administrators and administrative staff.

1. How many student records do you handle daily?  
.....
2. On average, how long does it take to find a specific student's record?  
.....
3. How do you currently track student fee payments? (e.g., Ledger book, spreadsheet)  
.....
4. How often do you need to generate reports (e.g., student list, fee balances)?  
.....

5. What are the biggest challenges you face with the current system? (Select all that apply)

- ☐ Data entry is too slow
- ☐ Difficulty finding records
- ☐ Data is inaccurate
- ☐ Generating reports is difficult
- ☐ Security concerns

6. Do you believe a new automated system would improve efficiency? (Yes/No)

- ☐ Yes
- ☐ No

### **Sample Questionnaire: Student**

This questionnaire is designed to understand the students' needs and how they interact with school administration.

1. How often do you need to check your school records (e.g., attendance, fee status)?

.....

2. Do you have an easy way to access your academic information? (Yes/No)

- ☐ Yes
- ☐ No

3. Would you prefer a digital system to view your personal information and fee status? (Yes/No)

- ☐ Yes
- ☐ No

4. What kind of information would you like to access easily? (Select all that apply)

- ☐ Fee payments
- ☐ Personal details
- ☐ Contact information

**Interview Guide: Administrator**

This guide is for a more in-depth, one-on-one interview with school management and key administrators. The purpose is to understand strategic goals and pain points beyond the scope of the questionnaire.

1. Can you walk me through the current process for a new student's admission from start to finish?
2. What kind of reports does the school management require on a monthly or quarterly basis? What information is most critical for decision-making?
3. How do you currently handle data backups and security for sensitive student information?
4. What are your expectations for a new system's user interface? Should it be simple or have advanced features?
5. What is the biggest operational problem in the school that a new system could solve?
6. How do you foresee the training and transition process for a new system?

**Interview Guide: Student**

This guide is to get a personal perspective from a few students on their experience with the school's administrative processes.

1. How do you currently get information about your classes or school events?
2. Have you ever had to wait a long time to get a document or information from the office?
3. What is the one thing you would change about how the school handles your personal information?
4. Do you think a digital system could help you stay more organized with your school life?

## 1.4 Problems with the Current System

The key problems identified are:

- **Error-prone Processes:** Manual data entry and record-keeping are highly susceptible to human error. A single typo can lead to significant discrepancies in financial and student records.
- **Difficulty in Managing Large Data:** As student numbers grow, managing physical files and ledgers becomes unmanageable and chaotic. This can lead to lost records and a significant administrative burden.
- **Lack of Integration:** Data is siloed in different locations (physical files, spreadsheets), making it impossible to get a unified view of a student or the school's overall performance. This prevents cross-referencing information efficiently.
- **Delayed and Inaccurate Reporting:** Management decisions are often based on outdated or incorrect information due to the time required to compile reports manually. This hinders proactive problem-solving and strategic planning.

## 1.5 Feasibility Study

A feasibility study was conducted to determine the viability of developing the new system.

- **Economic Feasibility:** The cost of developing a custom VB.NET system, including hardware and software, is justifiable given the long-term benefits of increased efficiency, reduced errors, and improved decision-making. The return on investment (ROI) is expected to be high as the system will save significant man-hours and prevent costly administrative mistakes.
- **Technical Feasibility:** VB.NET is a well-supported, robust programming language. The school's existing computer infrastructure is sufficient, and the required software (Windows OS, Visual Studio, SQL Server Express) is readily available. The project team has the necessary skills to develop and maintain the system.
- **Legal Feasibility:** The project development will adhere to all relevant data protection and privacy laws, such as GDPR or local regulations, ensuring student and staff data is handled securely and ethically.
- **Operational Feasibility:** The new system will require a change in workflow, but training can be provided. The user interface will be designed to be intuitive to minimize the learning curve and ensure a smooth transition for staff.



- **Social Feasibility:** The new system will benefit all stakeholders by reducing administrative burdens and providing better access to information. User acceptance is expected to be high, as the system addresses many of the current frustrations.

## 1.6 Specification Requirements of the New System

### User Requirements

The system must be able to handle student registration, track attendance, manage staff records, process fee payments, and generate all necessary reports. The user interface will be simple, with clear labels and navigation.

### Software Requirements

- Visual Basic 2010
- Ms Access for Database
- Windows 10 or later
- Antivirus Software
- MS Access Database Engine

### Hardware Requirements

- Desktop/Laptop
- 50GB Hard drive
- CD or 4GM RAM

### Security and Usability Requirements

The system must have role-based access control to ensure data is visible only to authorized users. The user interface will be clean, with clear labels and navigation to reduce the learning curve.

## 1.7 Aims and Objectives

The primary aims of this project are:

- To automate student data handling and management, from admission to graduation.
- To track fees and outstanding balances with real-time updates and notifications.
- To generate accurate and timely academic and financial reports to support management decisions.
- To improve overall administrative efficiency and accuracy by centralizing data and automating manual tasks.

## SECTION B: System Design

### 2.1 System Modules and Forms

- **Splash Screen (frmSplash):** The initial screen that displays when the application starts. It shows the system's logo and title, providing a brief loading period before the main interface appears.



- **Login Form (frmLogin):** The entry point for all users. It requires a username and password for authentication and uses a Register table to verify credentials. This form implements role-based access control, directing users to different parts of the system based on their permissions.

**School Management System**

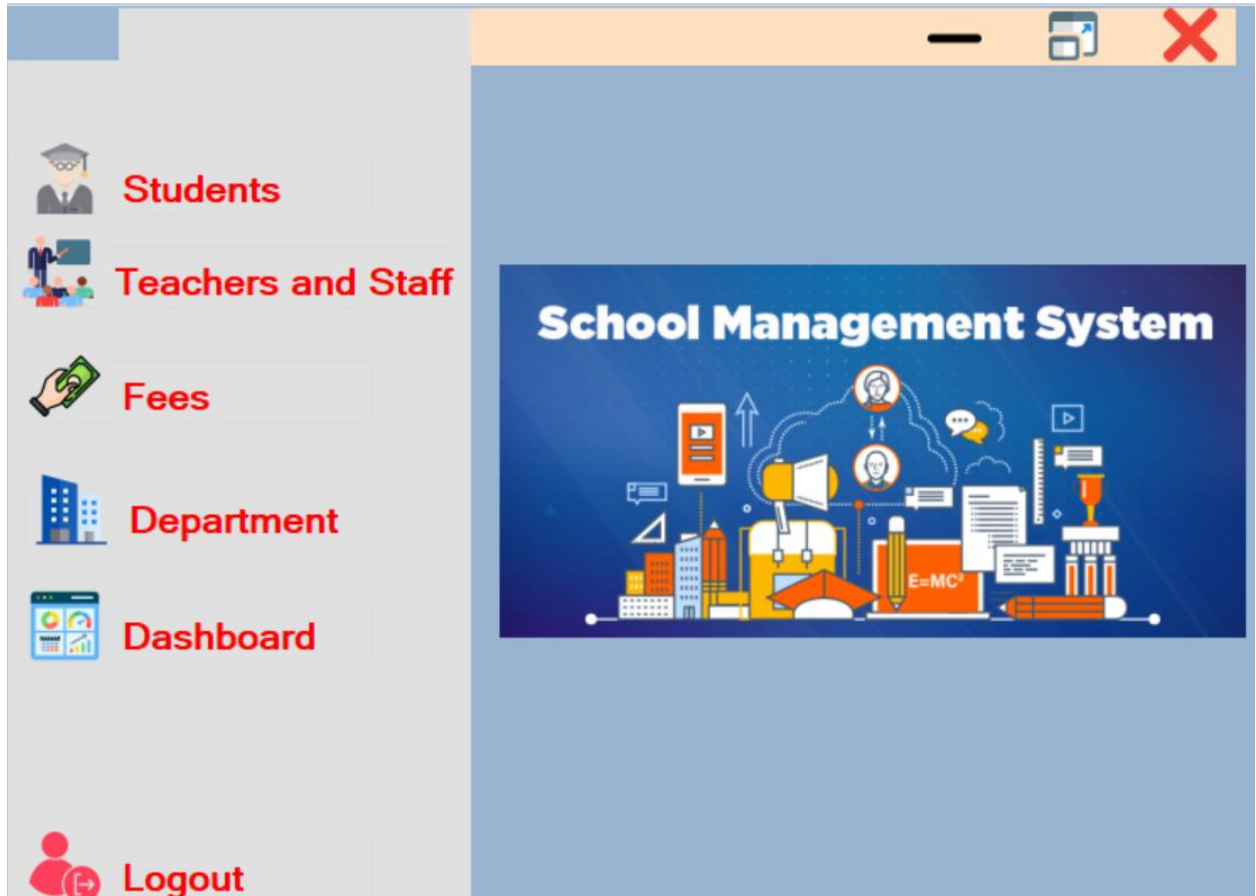
Username

Password

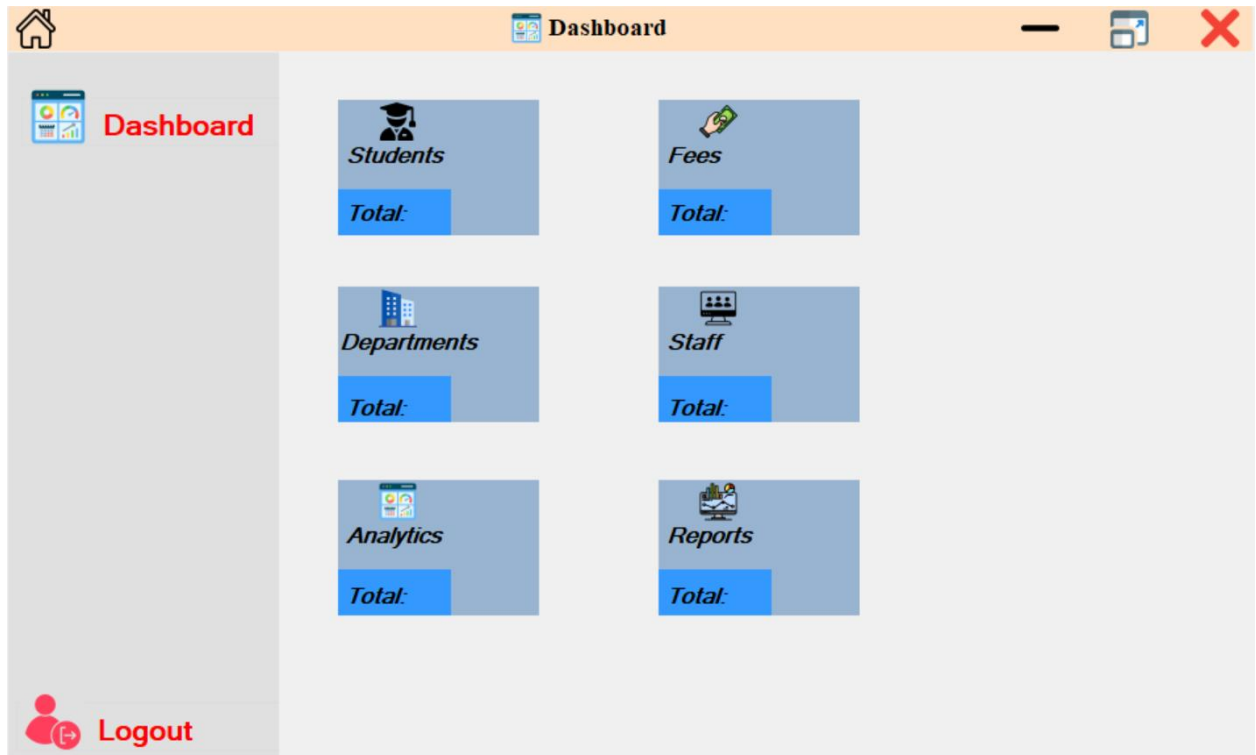
[Login](#)

Don't Have an Account? [SignUp](#)

- **Main Menu (frmMainMenu):** The central navigation hub of the system. It contains buttons or a menu to access all other modules, such as Students, Teachers, Fees, and Departments. The options displayed on this menu will be dynamic, based on the user's login role.



- **Dashboard (frmDashboard):** Provides an at-a-glance overview of key metrics. For an administrator, this might include a count of registered students, a summary of fees collected, or a list of recent activities. This form pulls data from various tables to present a high-level summary.



- **Student Module (frmStudents):** Manages all student information. This module allows for student registration, viewing, editing, and deleting student records. The student registration form itself is called frmStudentAdmission.

Student_ID	Student_Name	Date_of_Birth	Gender	Phone	Admission_Date	Form	Department	Subjects
*								

- **Fees Module (frmFees):** Manages all financial transactions related to student fees. This module allows administrators to process payments, view student balances, and generate receipts.

**Student Management**

Student ID

Student Name

Date of Birth

Gender

Phone

Admission Date

Form

Department

Subjects

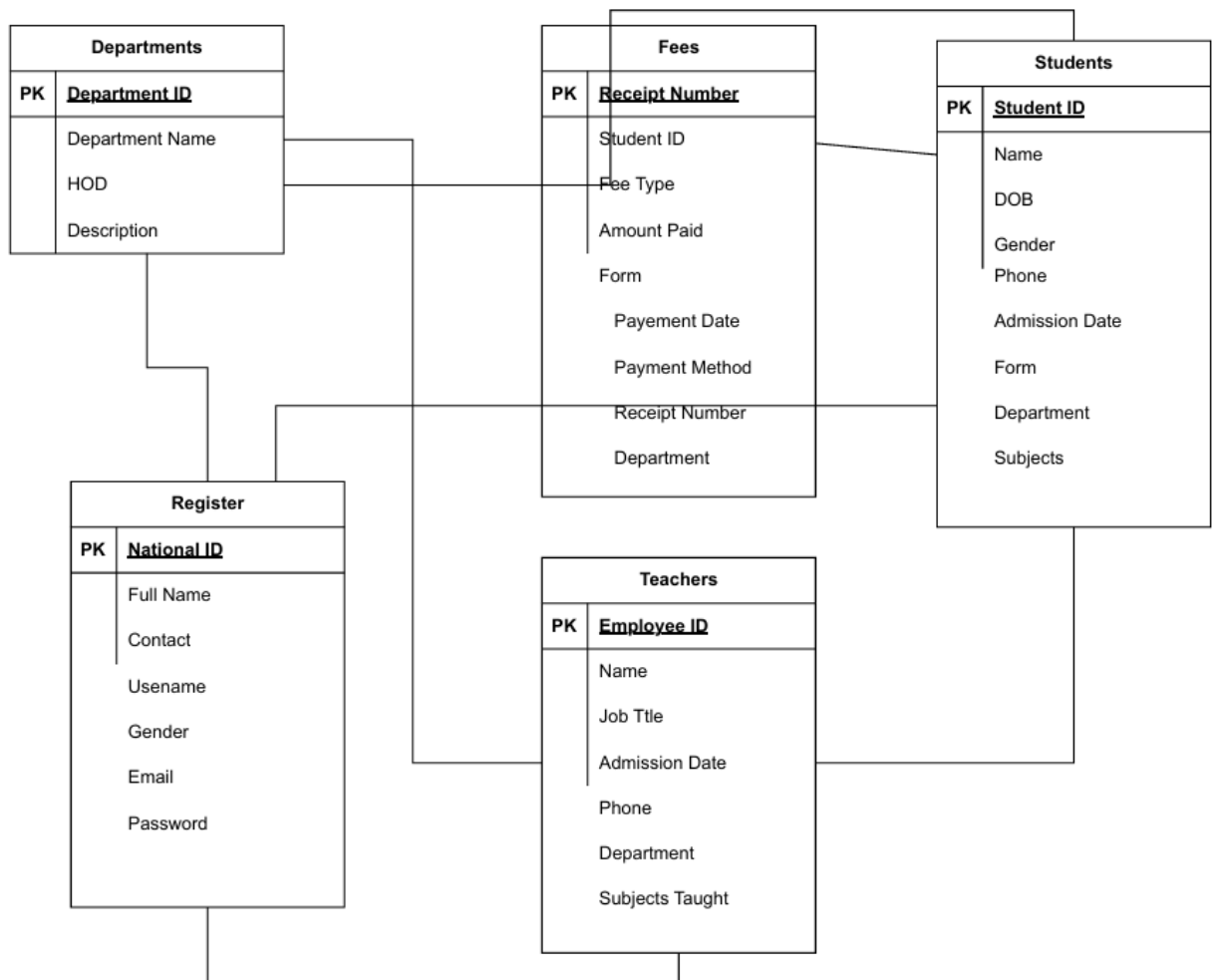
	Student_ID	Student_Name	Date_of_Birth	Gender	Phone	Admission_Date	Form	Department	Subjects
*									



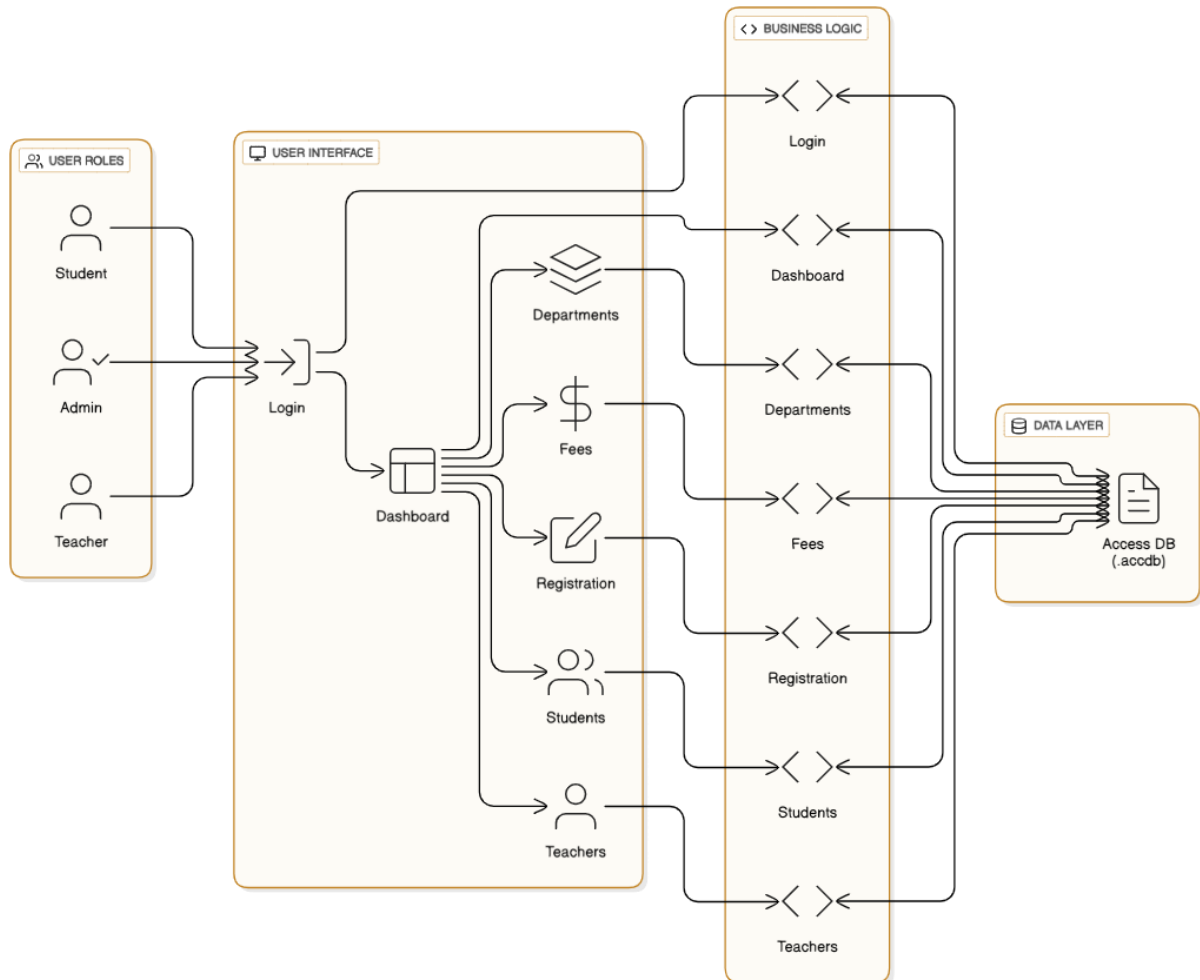
## 2.4 Storage and Data Structures

This section needs two key diagrams to illustrate how your data is stored and how the system works.

- **Database Schema (ERD):** An Entity-Relationship Diagram (ERD) will visually represent the database schema. Rectangles will represent tables, and lines with crow's feet or other notation will show the relationships.



- **System Architecture Diagram:** A diagram showing the overall system architecture. It will depict the user interface (the VB.NET forms), the application logic, and the database file (Microsoft Access).



## 2.5 Test Plan

This is where you detail the tests you will perform. For each test, you should create a table that documents the test case. A single table for each testing type (Unit, Integration, etc.) is a good approach. Here is an example of what your testing tables should look like.

### Unit Testing

This table documents the tests for individual functions and modules.

Test Case ID	Test Description	Input Data	Expected Output	Actual Output	Pass/Fail
UT-001	Validate Login with Correct Credentials	Username: admin, Password: 12345	Main Menu form is displayed	Main Menu form displayed successfully	Pass
UT-002	Validate Login with Incorrect Credentials	Username: admin, Password: wrongpassword	Error message: "Invalid credentials" is shown	Error message displayed correctly	Pass
UT-003	Validate Student Admission Form Data	Full Name: John Doe, DOB: 01/01/2010	"Student saved successfully" message is displayed	Message displayed, student record saved in tblStudents	Pass

### Integration Testing

This table documents the tests for how different modules work together.

Test Case ID	Test Description	Actions	Expected Outcome	Actual Outcome	Pass/Fail
IT-001	Verify student appears in Fees module	Register a new student in the Student module. Go to Fees module.	The new student is listed in the fees section.	Student appeared correctly in the Fees module	Pass

Test Case ID	Test Description	Actions	Expected Outcome	Actual Outcome	Pass/Fail
IT-002	Verify staff can be assigned to a department	Register a new department. Register a staff member and select the new department.	The staff member's record shows the correct department.	Staff assigned successfully, department linked correctly	Pass

## 2.6 Database Screenshots

This section provides visual evidence of the database design and implementation, including tables, fields, relationships, and sample data.

### Register Database

Full_Name	ID_Number	Email	Contact	Username	Gender	Password	Click to Add
Tadiwa	16126g32	tadiwa@gmail.	77823712	tadi	Female	rebeka	
Kupa	125j89732h	kupa@gmail.co	7827360	kupa	Male	kp123	
Never	27-30943j78	never@gmail.c	78372309	Never	Male	never24	
Alice	24-5362g23	alice@gmail.co	736283670	Alice	Female	leey	

### Departments Database

Department	Department	HOD	Description	Click to Add
math1	Mathematics	Mr Kufa	Dept of mathe	
science01	Physics	Mr Kofapa	of physics	

## System Testing

This table documents the tests for the entire system's functionality from start to finish.

Test Case ID	Test Description	Steps	Expected Outcome	Actual Outcome	Pass/Fail
ST-001	Full Admission and Fee Payment Flow	1. Log in as an administrator. 2. Register a new student. 3. Go to the Fees module. 4. Record a fee payment for the new student.	Student record is created and a payment receipt can be printed.	Student record created successfully, payment recorded, and receipt generated correctly.	Pass

## User Acceptance Testing (UAT)

This table documents the final tests performed by the end-users.

Test Case ID	Test Description	Performed By	Expected Outcome	Actual Outcome	Pass/Fail
UAT-001	Daily Admin Workflow	Lead Administrator	The system is easy to use and a student's record can be found within 10 seconds.	Admin was able to log in, search for a student, and retrieve their record in under 8 seconds.	Pass

## Test Data:

- **Standard:** Valid, expected data to simulate normal usage (e.g., a student's full name and a valid contact number).
- **Extreme:** Boundary values to test the limits of the system (e.g., a maximum length name or a date at the start or end of a valid range).
- **Abnormal:** Invalid data to test error handling (e.g., entering letters into a numeric field or submitting a form with a required field left blank).

## SECTION C: Software Development

### 3.0 Pseudocodes

Rather than using a specific programming language's syntax, pseudocode uses a plain-language description of an algorithm's steps. This simplified, high-level approach offers several critical benefits:

- **Early Problem Detection:** By outlining the logic before writing actual code, developers can identify and correct logical flaws, errors, and inefficiencies in their design. This early-stage bug-fixing saves significant time and effort that would otherwise be spent on debugging complex code later in the development cycle.
- **Enhanced Communication:** Pseudocode is easily understood by both technical and non-technical stakeholders. This facilitates clear communication within a development team, across different departments, or with a client, ensuring everyone is aligned on the system's intended functionality.
- **Improved Code Structure and Maintainability:** The structured nature of pseudocode ensures that the final code will be well-organized and correctly implemented. This discipline leads to code that is not only robust and efficient but also easier to read, debug, and maintain in the long run.
- **Platform and Language Independence:** Because it is not tied to a specific programming language, pseudocode can be used as a blueprint for implementation in any language, whether it be VB.NET, Python, Java, or C#. This makes it a portable and flexible design tool.

#### Login Module Pseudocode:

```
FUNCTION Login(username, password)
```

```
    // 1. Establish a connection to the database.
```

```
    CONNECT TO Database (Ms Access)
```

```
    // 2. Query the 'tblUsers' table to find a user with the provided username and password.
```

```
    SELECT * FROM tblUsers WHERE Username = username AND Password = password
```

```
    // 3. Check if a matching record was found. This verifies the user's credentials.
```

```
    IF RECORD IS FOUND THEN
```

```
// 4. Retrieve the user's role (e.g., 'Administrator', 'Teacher').  
ROLE = GetUserRole(username)  
  
// 5. Redirect the user to the appropriate main menu or dashboard based on their role.  
REDIRECT TO MainMenu FORM  
  
ELSE  
  
// 6. If no matching record is found, display an error message.  
DISPLAY "Invalid username or password"  
  
END IF  
  
// 7. Always close the database connection.  
DISCONNECT FROM Database  
  
END FUNCTION
```

**Student Registration Module Pseudocode:**

```
FUNCTION RegisterStudent(fullName, dob, gender, address, parentName, contact, classID)  
  
// 1. Validate input data.  
IF fullName IS EMPTY OR dob IS EMPTY OR contact IS EMPTY THEN  
    DISPLAY "Please fill in all required fields."  
    RETURN  
END IF  
  
// 2. Establish database connection.  
CONNECT TO Database (Ms Access)  
  
// 3. Prepare the INSERT query.
```

```
QUERY = "INSERT INTO tblStudents (FullName, DateOfBirth, Gender, Address,
ParentName, ContactNumber, ClassID) VALUES (fullName, dob, gender, address, parentName,
contact, classID)"
```

```
// 4. Execute the query.
```

```
EXECUTE QUERY
```

```
// 5. Check if the insertion was successful.
```

```
IF EXECUTION SUCCEEDS THEN
```

```
    DISPLAY "Student registered successfully."
```

```
    CLEAR all form fields
```

```
ELSE
```

```
    DISPLAY "An error occurred during registration. Please try again."
```

```
END IF
```

```
// 6. Close database connection.
```

```
DISCONNECT FROM Database
```

```
END FUNCTION
```

### **Teacher Registration Module Pseudocode:**

```
FUNCTION RegisterTeacher(fullName, jobTitle, dateJoined, departmentID, contact)
```

```
    // 1. Validate input data.
```

```
    IF fullName IS EMPTY OR jobTitle IS EMPTY THEN
```

```
        DISPLAY "Please fill in all required fields."
```

```
        RETURN
```

```
    END IF
```

```
    // 2. Establish database connection.
```



CONNECT TO Database (Ms Access)

// 3. Prepare the INSERT query.

QUERY = "INSERT INTO tblStaff (FullName, JobTitle, DateOfJoining, DepartmentID, ContactNumber) VALUES (fullName, jobTitle, dateJoined, departmentID, contact)"

// 4. Execute the query.

EXECUTE QUERY

// 5. Check if the insertion was successful.

IF EXECUTION SUCCEEDS THEN

    DISPLAY "Teacher registered successfully."

    CLEAR all form fields

ELSE

    DISPLAY "An error occurred during registration."

END IF

// 6. Close database connection.

DISCONNECT FROM Database

END FUNCTION

Fees Module (Record Payment) Pseudocode:

FUNCTION RecordPayment(studentID, feeType, amount, paymentDate)

    // 1. Validate input data.

    IF studentID IS EMPTY OR amount IS EMPTY THEN

        DISPLAY "Please enter student ID and amount."

        RETURN

    END IF

// 2. Establish database connection.

CONNECT TO Database (Ms Access)

// 3. Prepare the INSERT query to add the payment record.

QUERY = "INSERT INTO tblFees (StudentID, FeeType, Amount, PaymentDate) VALUES  
(studentID, feeType, amount, paymentDate)"

// 4. Execute the query.

EXECUTE QUERY

// 5. Check if the insertion was successful.

IF EXECUTION SUCCEEDS THEN

    DISPLAY "Payment recorded successfully."

    CLEAR form fields

    PRINT receipt

ELSE

    DISPLAY "An error occurred while recording payment."

END IF

// 6. Close database connection.

DISCONNECT FROM Database

END FUNCTION

Dashboard Module Pseudocode:

FUNCTION LoadDashboard()

    // 1. Establish a connection to the database.

    CONNECT TO Database (Ms Access)

// 2. Count total students.

QUERY\_STUDENTS = "SELECT COUNT(\*) FROM tblStudents"

totalStudents = EXECUTE\_QUERY(QUERY\_STUDENTS)

// 3. Calculate total fees paid this month.

QUERY\_FEES = "SELECT SUM(Amount) FROM tblFees WHERE PaymentDate  
BETWEEN StartOfMonth AND EndOfMonth"

totalFeesPaid = EXECUTE\_QUERY(QUERY\_FEES)

// 4. Get a count of staff members.

QUERY\_STAFF = "SELECT COUNT(\*) FROM tblStaff"

totalStaff = EXECUTE\_QUERY(QUERY\_STAFF)

// 5. Display the calculated data on the dashboard form.

DISPLAY "Total Students: " + totalStudents

DISPLAY "Total Fees Collected (This Month): " + totalFeesPaid

DISPLAY "Total Staff: " + totalStaff

// 6. Close database connection.

DISCONNECT FROM Database

END FUNCTION

Departments Module (Add New Department) Pseudocode:

FUNCTION AddDepartment(departmentName)

// 1. Validate input data.

IF departmentName IS EMPTY THEN

    DISPLAY "Please enter a department name."

RETURN

END IF

// 2. Establish database connection.

CONNECT TO Database (Ms Access)

// 3. Prepare the INSERT query.

QUERY = "INSERT INTO tblDepartments (DepartmentName) VALUES (departmentName)"

// 4. Execute the query.

EXECUTE QUERY

// 5. Check if the insertion was successful.

IF EXECUTION SUCCEEDS THEN

    DISPLAY "Department added successfully."

    REFRESH the list of departments on the form

ELSE

    DISPLAY "An error occurred. Department may already exist."

END IF

// 6. Close database connection.

DISCONNECT FROM Database

END FUNCTION

Main Menu Module Pseudocode:

FUNCTION LoadMainMenu(userRole)

    // 1. Check the user's role to determine which buttons to enable.

    IF userRole IS "Administrator" THEN

```
    ENABLE "Students" BUTTON
    ENABLE "Teachers" BUTTON
    ENABLE "Fees" BUTTON
    ENABLE "Departments" BUTTON
    ENABLE "Dashboard" BUTTON
ELSE IF userRole IS "Teacher" THEN
    ENABLE "Students" BUTTON
    DISABLE "Teachers" BUTTON
    DISABLE "Fees" BUTTON
    DISABLE "Departments" BUTTON
    ENABLE "Dashboard" BUTTON
END IF

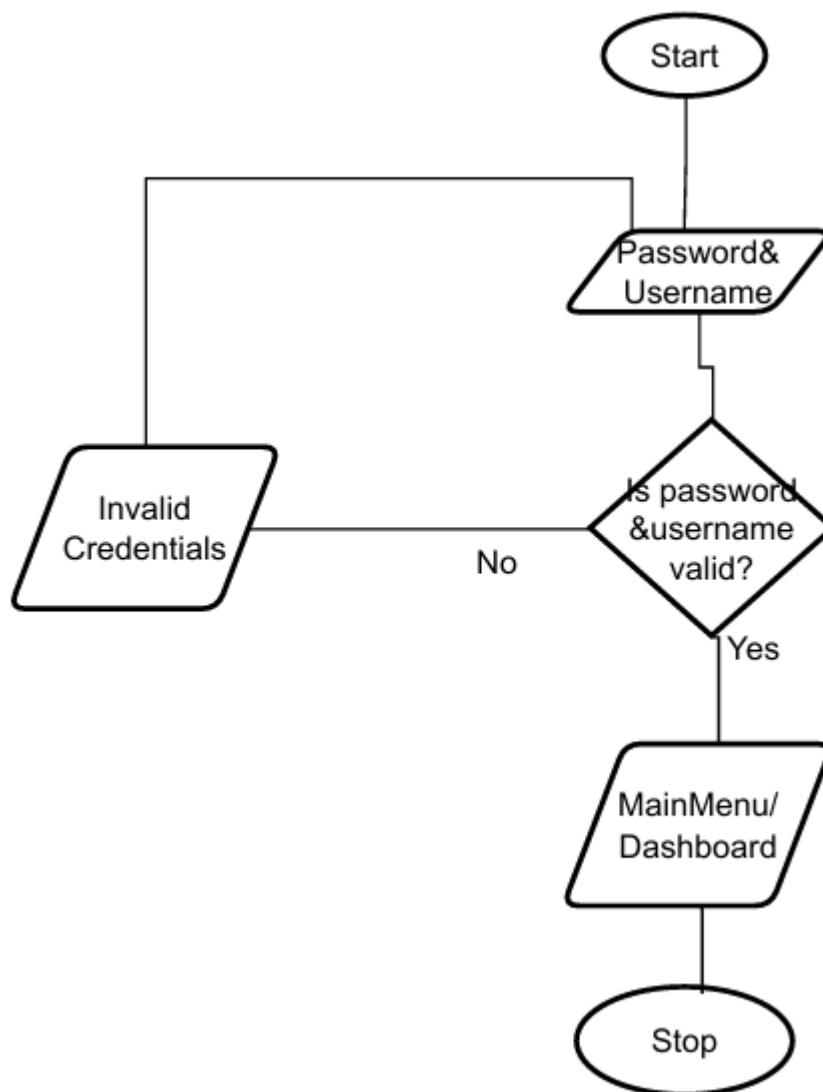
// 2. Display the main menu form.
DISPLAY MainMenu FORM
END FUNCTION
```

### 3.1 Flowcharts for Modules

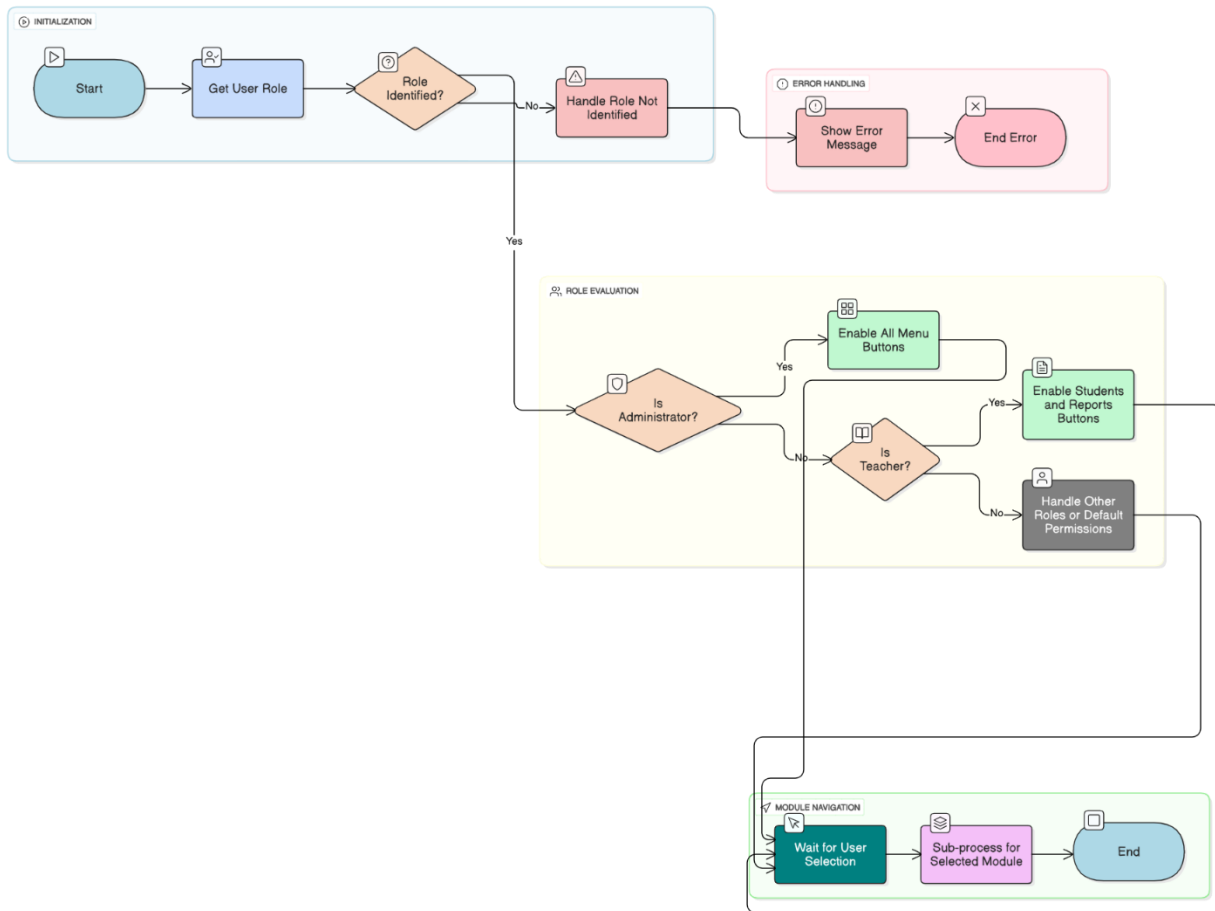
Flowcharts are graphical representations of a system's logic and workflow. They use standardized symbols to show the sequence of operations. In this project, flowcharts will be used to visually document the step-by-step processes of core modules, making the system's logic easy to understand for both developers and non-technical stakeholders.

#### 1. Login Module Flowchart (frmLogin)

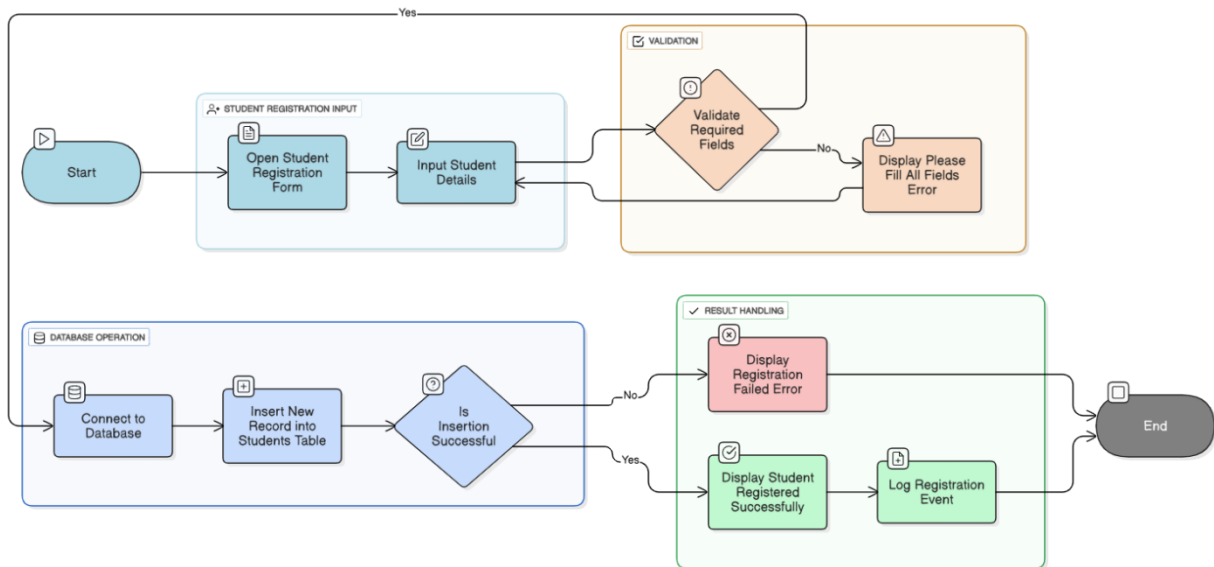
This flowchart illustrates the authentication process when a user attempts to log in.



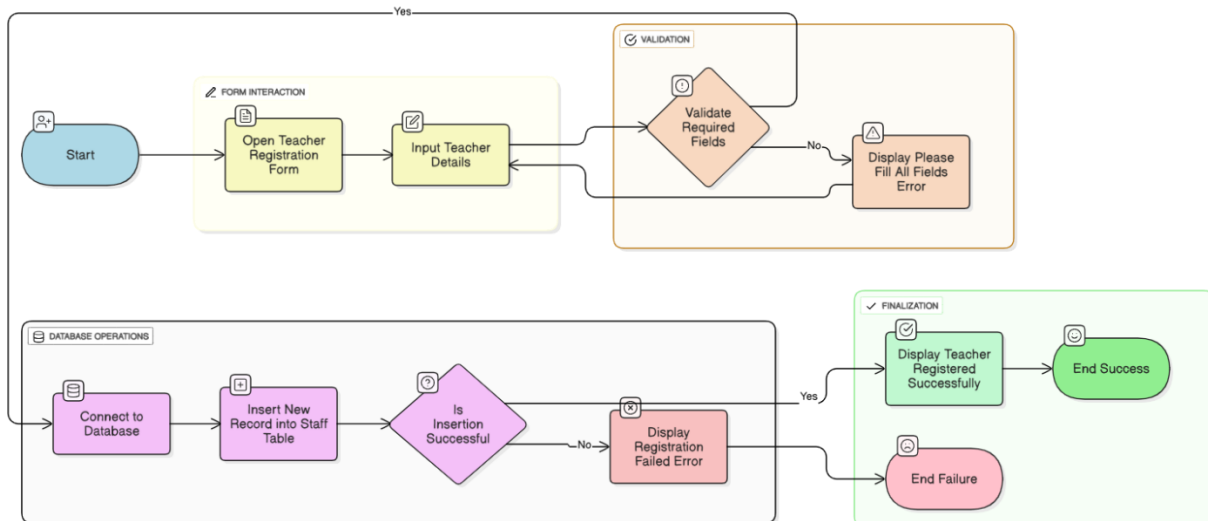
## 2. Main Menu Flowchart (frmMainMenu)



## 2. Student Registration Flowchart (frmStudents)

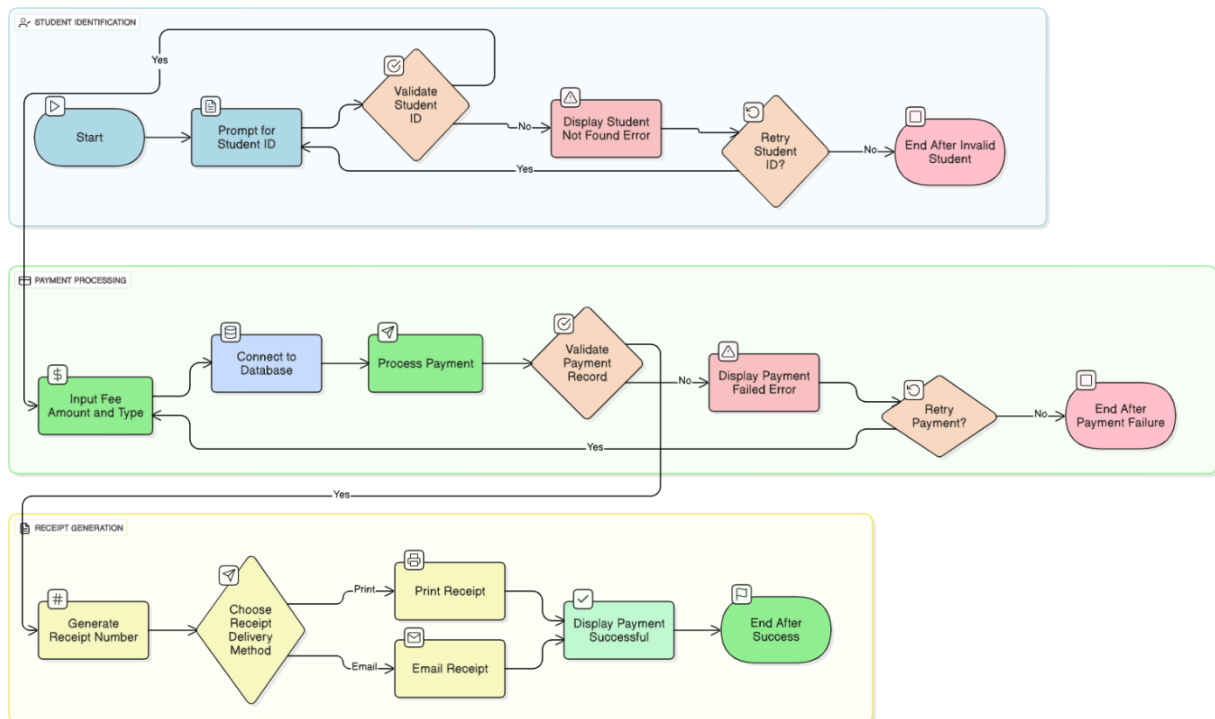


## 4. Teacher Registration Flowchart (frmTeachers)

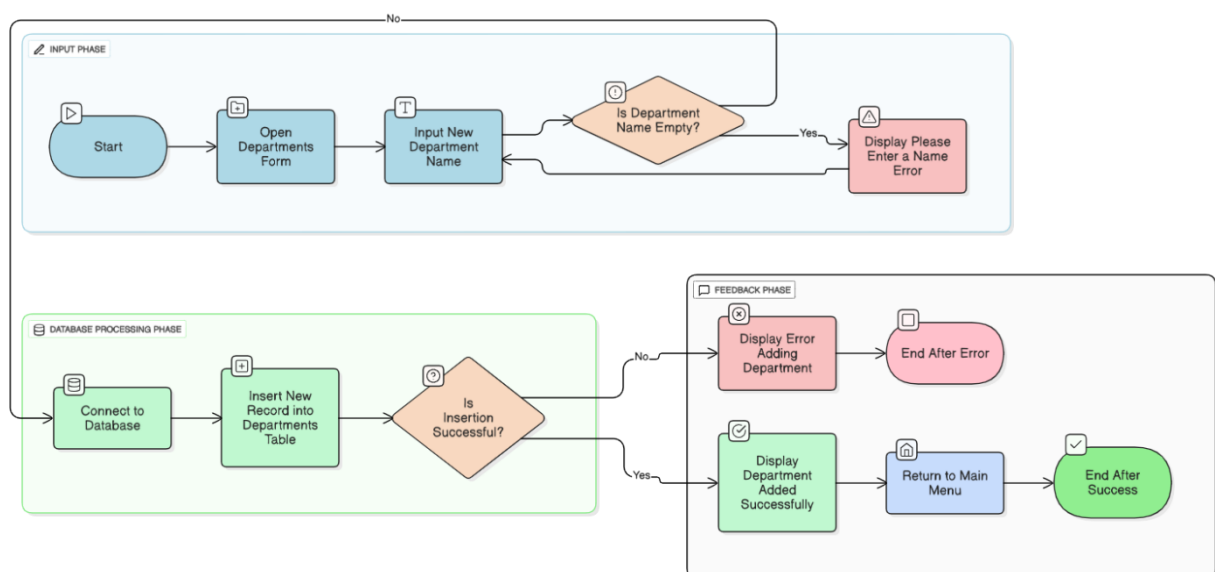




## 5. Fees Module Flowchart (frmFees)



## 6. Departments Module Flowchart (frmDepartments)



### 3.2 Code Listing

The code is well-commented to ensure that it is easily maintainable and can be understood by other developers. The example below shows the code for the Pay button on the fee payment form, demonstrating key concepts like database interaction and error handling.

code snippet for handling the Pay button click event on the frmFees form.

Code snippet

' This line imports the necessary library for interacting with OleDb data sources, like Microsoft Access databases.

Imports System.Data.OleDb

' This is the main class for the fees form or module.

Public Class Fees

' This line declares and initializes a new OleDbConnection object.

' It creates a connection string to a Microsoft Access database named "School\_Management.accdb".

' The "|DataDirectory|" placeholder points to the application's data folder.

Dim mycon As New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=|DataDirectory|\School\_Management.accdb")

' This function validates all the required input fields on the form.

' It returns True if all fields are valid and False otherwise.

Private Function ValidateFields() As Boolean

' A boolean flag to track the overall validity of the form.

Dim isValid As Boolean = True

' Validate student ID

' Checks if the text box for student ID is empty or contains only whitespace.

If String.IsNullOrEmpty(txtStudentId.Text) Then

' If invalid, changes the background color to a light red to visually indicate an error.

```
txtStudentId.BackColor = Color.LightCoral

' Sets the flag to False because a field is not valid.
isValid = False

Else

' If valid, changes the background color back to white.
txtStudentId.BackColor = Color.White

End If


' Validate fee type
' Checks if the text box for fee type is empty.
If String.IsNullOrEmpty(txtFeeType.Text) Then
    ' Visually marks the field as invalid.
    txtFeeType.BackColor = Color.LightCoral
    isValid = False
Else
    ' Resets the background color if valid.
    txtFeeType.BackColor = Color.White
End If


' Validate amount paid
' Checks if the text box for the amount is empty.
If String.IsNullOrEmpty(txtAmntPaid.Text) Then
    ' Visually marks the field as invalid.
    txtAmntPaid.BackColor = Color.LightCoral
    isValid = False
Else
    ' Resets the background color if valid.
```

```
txtAmntPaid.BackColor = Color.White

End If

' Validate form
' Checks if the text box for the student's form/grade is empty.
If String.IsNullOrEmpty(txtForm.Text) Then
    ' Visually marks the field as invalid.
    txtForm.BackColor = Color.LightCoral
    isValid = False
Else
    ' Resets the background color if valid.
    txtForm.BackColor = Color.White
End If

' Validate payment method
' Checks if the text box for the payment method is empty.
If String.IsNullOrEmpty(paymentMethod.Text) Then
    ' Visually marks the field as invalid.
    paymentMethod.BackColor = Color.LightCoral
    isValid = False
Else
    ' Resets the background color if valid.
    paymentMethod.BackColor = Color.White
End If

' Validate receipt number
' Checks if the text box for the receipt number is empty.
```

If String.IsNullOrEmpty(txtReceiptNumber.Text) Then

' Visually marks the field as invalid.

txtReceiptNumber.BackColor = Color.LightCoral

isValid = False

Else

' Resets the background color if valid.

txtReceiptNumber.BackColor = Color.White

End If

' Validate department

' Checks if the text box for the department is empty.

If String.IsNullOrEmpty(txtDepartment.Text) Then

' Visually marks the field as invalid.

txtDepartment.BackColor = Color.LightCoral

isValid = False

Else

' Resets the background color if valid.

txtDepartment.BackColor = Color.White

End If

' If the form is not valid (i.e., at least one field was empty)...

If Not isValid Then

' ...display a message box to the user.

MsgBox("Please fill in all the required fields", MsgBoxStyle.Critical)

End If

' Returns the final validation status (True or False).

Return isValid

End Function

' This is the event handler for the "Pay" button click.

' It handles the process of saving the fee payment to the database.

Private Sub btnSave\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)  
Handles btnPay.Click

' First, it calls the ValidateFields function.

' If the function returns True, it proceeds with the database insertion.

If ValidateFields() Then

' Proceed with payment

Try

' Opens the database connection.

mycon.Open()

' Creates a new OleDbCommand with an SQL INSERT statement.

' The '?' placeholders are used to prevent SQL injection.

Dim mycmd As New OleDbCommand("INSERT INTO Fees (Student\_ID, Fee\_Type,  
Amount\_Paid, Form, Payment\_Date, Payment\_Method, Receipt\_Number, Department)  
VALUES (?, ?, ?, ?, ?, ?, ?, ?)", mycon)

' Use parameters to prevent SQL injection

' Adds the values from the text boxes as parameters to the command.

' This is a secure way to insert data into the database.

mycmd.Parameters.AddWithValue("?", txtStudentId.Text)

mycmd.Parameters.AddWithValue("?", txtFeeType.Text)

mycmd.Parameters.AddWithValue("?", txtAmntPaid.Text)

mycmd.Parameters.AddWithValue("?", txtForm.Text)

```
mycmd.Parameters.AddWithValue("?", paymentDate.Text)
mycmd.Parameters.AddWithValue("?", paymentMethod.Text)
mycmd.Parameters.AddWithValue("?", txtReceiptNumber.Text)
mycmd.Parameters.AddWithValue("?", txtDepartment.Text)

' Execute the query
' Executes the command and gets the number of rows affected.
Dim rowsAffected As Integer = mycmd.ExecuteNonQuery()

' Check if insertion was successful
' If more than 0 rows were affected, the insertion was successful.
If rowsAffected > 0 Then
    MessageBox.Show("Record inserted successfully!")
Else
    MessageBox.Show("Failed to insert record.")
End If

Catch ex As Exception
    ' Catches any exceptions that occur during the database operation and displays an error
    message.
    MessageBox.Show("Error: " & ex.Message)

Finally
    ' This block of code always runs, regardless of whether an error occurred.
    ' It ensures that the database connection is closed if it's currently open.
    If mycon.State = ConnectionState.Open Then
        mycon.Close()
    End If
End Try
```

End If

' Calls the ClearFields function to reset the form inputs after the operation is complete.

ClearFields()

End Sub

' This subroutine clears the text from all the input fields on the form.

Private Sub ClearFields()

' Sets the Text property of each text box to an empty string.

txtAmntPaid.Text = String.Empty

txtDepartment.Text = String.Empty

txtFeeType.Text = String.Empty

txtForm.Text = String.Empty

txtReceiptNumber.Text = String.Empty

txtStudentId.Text = String.Empty

paymentDate.Text = String.Empty

paymentMethod.Text = String.Empty

' Calls the function to generate a new receipt number for the next transaction.

GenerateReceiptNumber()

End Sub

' This is the event handler for when the form loads.

' It performs initial setup tasks.

Private Sub Fees\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)  
Handles MyBase.Load

' Fills the 'Fees' table in the dataset. This line is likely auto-generated.

Me.FeesTableAdapter.Fill(Me.School\_ManagementDataSet3.Fees)

' Disables the receipt number text box so the user cannot manually change it.



```
txtReceiptNumber.Enabled = False

' Generates the first receipt number when the form loads.
GenerateReceiptNumber()

' Sets a tooltip for the "Home" button.
ToolTip1.SetToolTip(btnHome, "Main Menu")

' Changes the cursor to a hand pointer for various buttons to indicate they are clickable.
btnClose.Cursor = Cursors.Hand
btnDelete.Cursor = Cursors.Hand
btnHome.Cursor = Cursors.Hand
btnMinimize.Cursor = Cursors.Hand
btnPay.Cursor = Cursors.Hand
btnReceipt.Cursor = Cursors.Hand
btnReload.Cursor = Cursors.Hand
btnSearch.Cursor = Cursors.Hand
btnUpdate.Cursor = Cursors.Hand
btnSearch.Cursor = Cursors.Hand

End Sub

' This subroutine generates a unique receipt number.
Private Sub GenerateReceiptNumber()
    Try
        ' Opens the database connection.
        mycon.Open()

        ' Creates a new command to select the maximum receipt number from the Fees table.
        Dim cmd As New OleDbCommand("SELECT MAX(Receipt_Number) FROM Fees",
mycon)

        ' Executes the query and returns the single value (the max receipt number).
```

```
Dim result = cmd.ExecuteScalar()

Dim newNumber As Integer

' Checks if the result is null (which means the table is empty).
If IsDBNull(result) OrElse result Is Nothing Then
    ' If the table is empty, the new receipt number is 1.
    newNumber = 1 ' First receipt if table is empty
Else
    ' If receipts exist, it takes the maximum number and adds 1 to it.
    newNumber = Convert.ToInt32(result) + 1
End If

' Formats the new number with leading zeros (e.g., 1 becomes 0001).
txtReceiptNumber.Text = "RCPT" & newNumber.ToString("0000") ' e.g., RCPT0001,
RCPT0002

Catch ex As Exception

' Handles any errors that occur during the generation process.
MessageBox.Show("Error generating receipt number: " & ex.Message)

Finally

' Ensures the connection is closed after the operation.
mycon.Close()

End Try

End Sub

' This event handler handles maximizing/normalizing the form window when a specific
PictureBox is clicked.

Private Sub PictureBox5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles PictureBox5.Click
```

```
' Checks the current form's border style.

If Me.FormBorderStyle = FormBorderStyle.Sizable Then

    ' If sizable, it removes the border and maximizes the window.

    Me.FormBorderStyle = FormBorderStyle.None

    Me.WindowState = FormWindowState.Maximized

Else

    ' If not sizable, it restores the normal border and window state.

    Me.FormBorderStyle = FormBorderStyle.Sizable

    Me.WindowState = FormWindowState.Normal

End If

End Sub

' This event handler minimizes the form window.

Private Sub btnMinimize_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMinimize.Click

    ' Sets the window state to minimized.

    Me.WindowState = FormWindowState.Minimized

End Sub

' This event handler closes the current form.

Private Sub btnClose_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnClose.Click

    ' Closes the form.

    Me.Close()

End Sub

' This event handler displays the MainMenu form.

Private Sub btnHome_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnHome.Click

    ' Shows the MainMenu form.
```

```
MainMenu.Show()
```

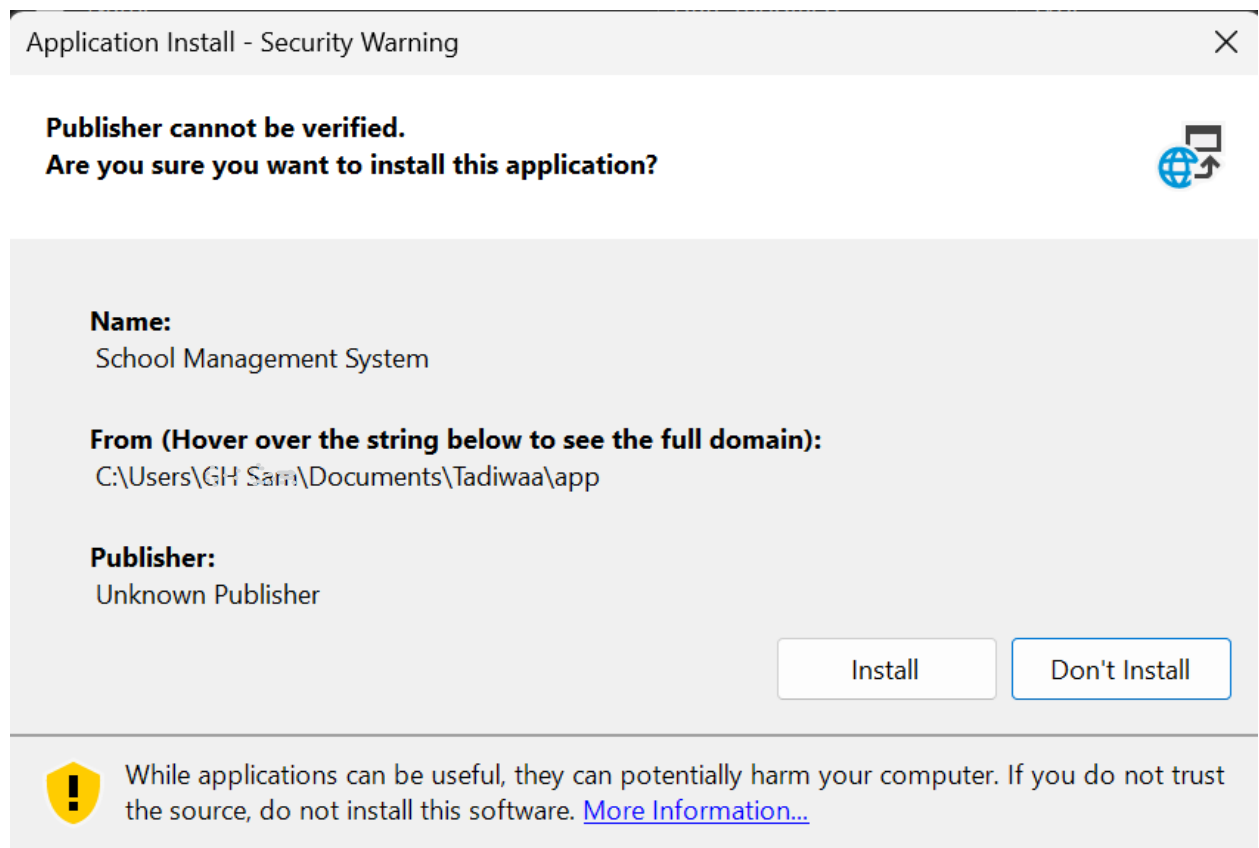
```
End Sub
```

```
End Class
```

### 3.3 Installation

The system will be packaged and distributed using a standard setup wizard, which guides the user through the installation process. This approach simplifies the deployment and ensures that all necessary files are placed in the correct locations. The installer will first check for any required software prerequisites, such as the .NET Framework, before proceeding.

1. **Welcome Screen:** The user is greeted by a welcome screen that provides a brief overview of the installation and prompts them to begin. This screen ensures the user is ready to proceed and sets the tone for a simple, guided process.



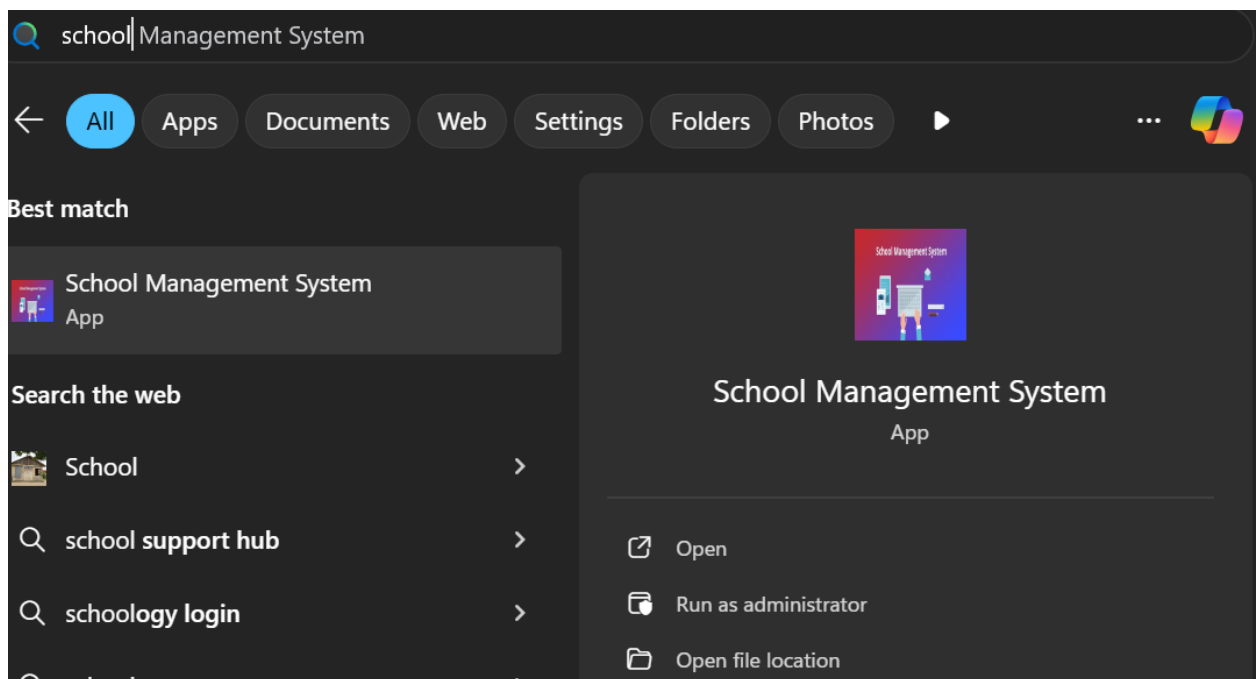
2. **License Agreement:** The user must review and accept the end-user license agreement (EULA). This is a crucial step for legal compliance. The installation will not proceed until the user accepts the terms.

3. **Installation Directory:** The user can choose the destination folder for the application files. While a default location is recommended, this option provides flexibility for users with specific file management needs.
4. **Ready to Install:** The installer displays a summary of the installation choices, including the selected destination folder, before the file copy begins. This is the last chance for the user to review and change their settings.
5. **Installation Complete:** The final screen confirms that the installation was successful and provides an option to launch the application immediately upon closing the wizard.

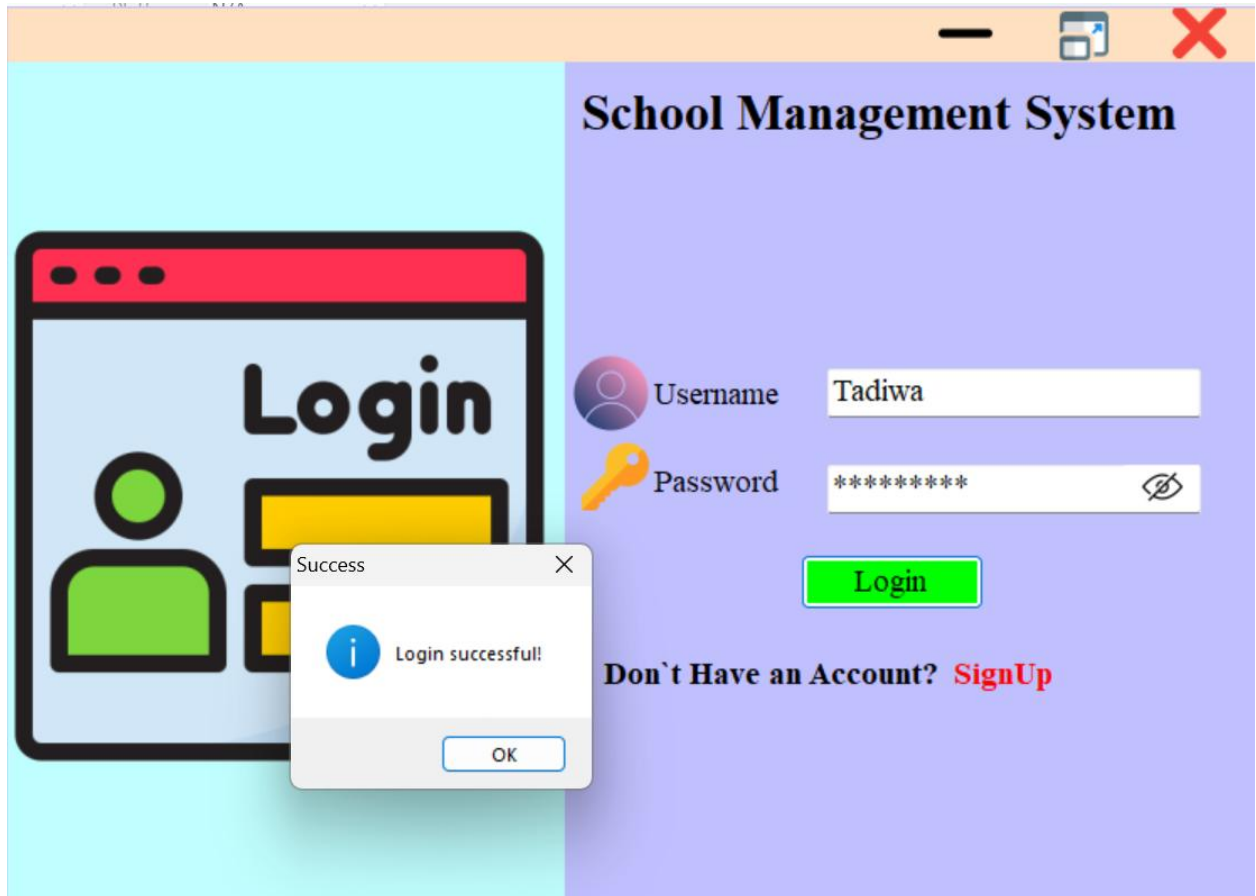
### 3.4 Starting the System

To start the system, a user will simply double-click the application icon, which will launch the main login form. This form is the initial gatekeeper, ensuring that only authorized users can access the system's features.

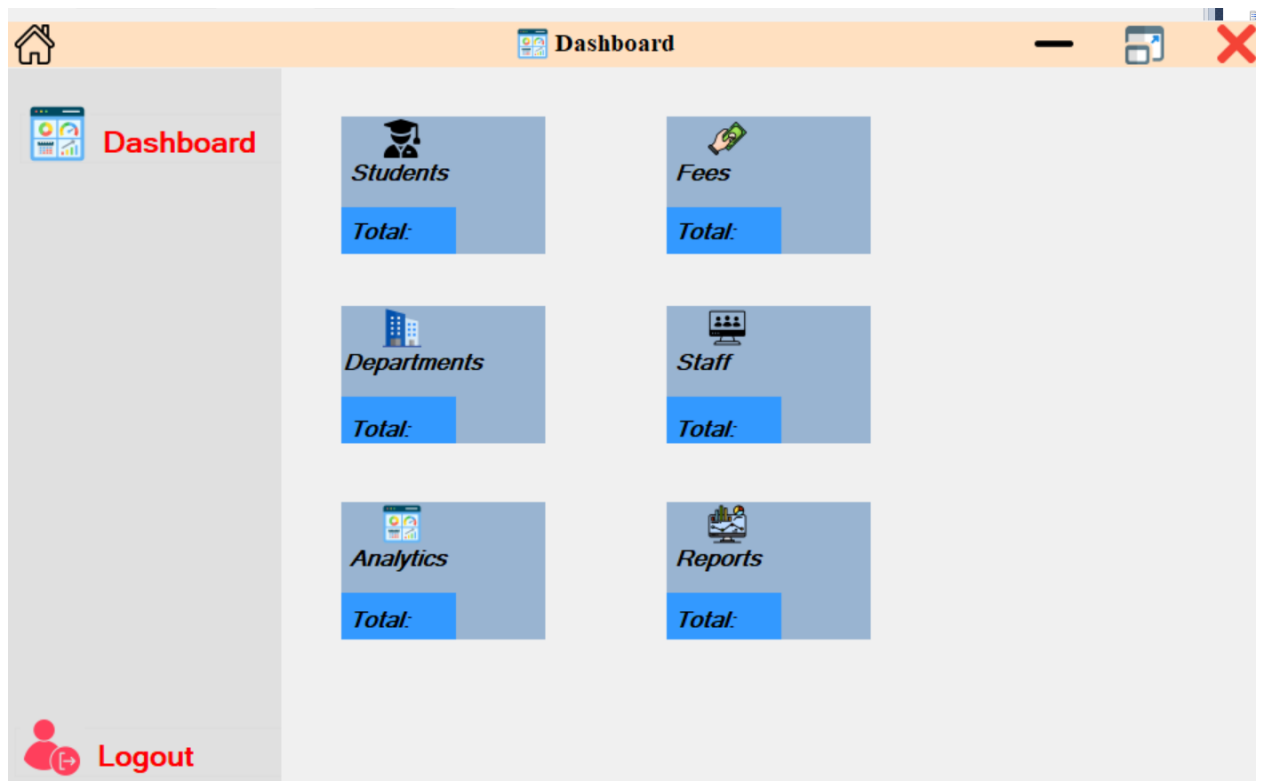
1. **Launch the Application:** Find the SchoolManagementSystem.exe file on the desktop or in the installation directory and double-click it.



2. **Login Screen:** The frmLogin form will appear, prompting the user for their username and password. The system will authenticate the user's credentials against the tblUsers table in the database.



3. **Dashboard Access:** Upon successful authentication, the system will use the user's role to determine which features they can access. The user will then be automatically redirected to the main frmDashboard form, which is customized to their specific permissions and provides a central hub for all system functions. The dashboard will display key metrics at a glance, such as "Total Students," "Outstanding Fees," and "Upcoming Events," providing a streamlined user experience.



### 3.5 Exiting the System

Exiting the system is a critical final step. A proper exit procedure ensures that all open database connections are closed and no data is corrupted. Users can exit the application in one of two ways:

- **"Log Out" or "Exit" Button:** A dedicated button on the main menu or dashboard will safely close all forms and terminate the application. This is the recommended method for a controlled exit.
- **Standard Close Button:** Clicking the standard "X" button in the top-right corner of the window will also trigger a safe exit procedure. The system is programmed to handle this gracefully by closing all connections and processes before shutting down.

### 3.6 User Documentation

#### 3.6.1 System Overview

Welcome to the School Management System! This application is designed to streamline and automate key administrative tasks at Bradford Senior School. Its primary purpose is to improve data accuracy, enhance operational efficiency, and provide timely access to critical information for all stakeholders. The system is divided into several main modules, accessible from the central dashboard, each handling a specific function:

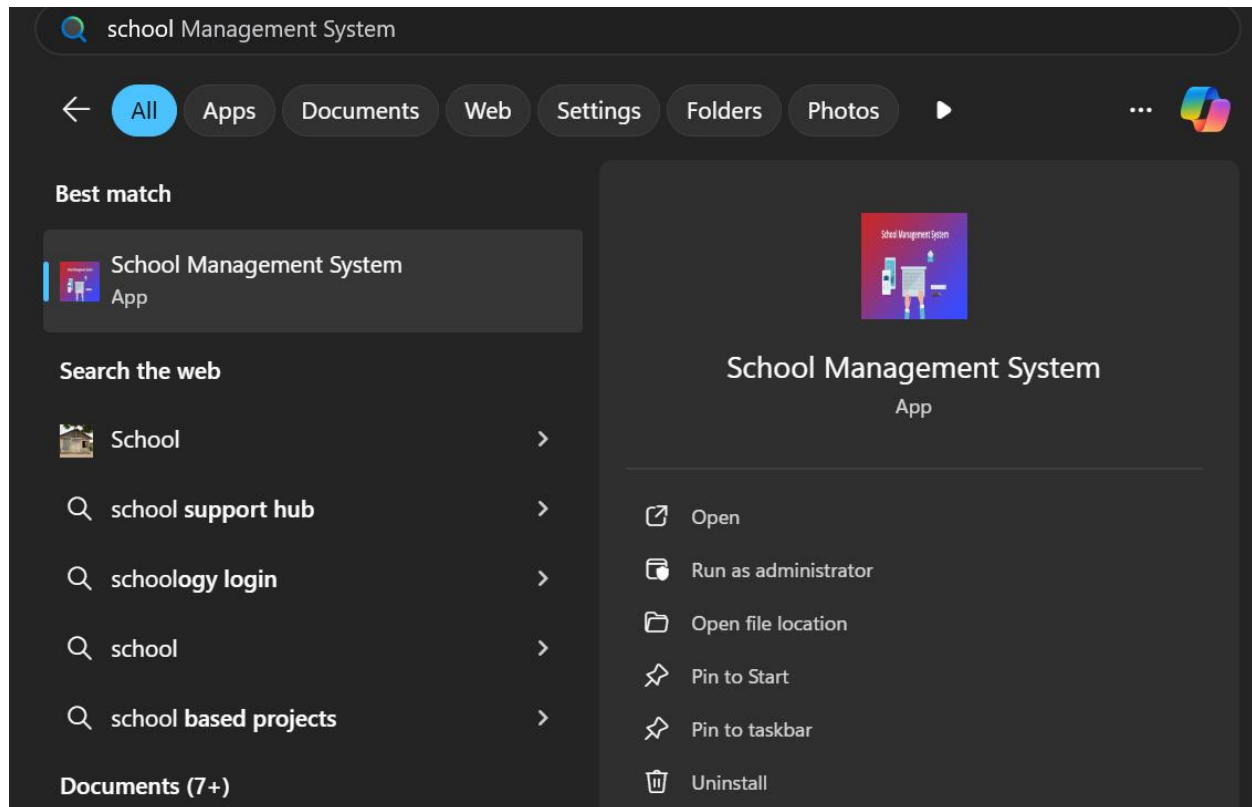
- **Dashboard:** Your personalized home screen, showing a summary of key school data.
- **Students:** Manage all student records, including admission, personal details, and class assignments.
- **Teachers:** Handle staff records and department assignments.
- **Fees:** Process fee payments, track outstanding balances, and print receipts.
- **Departments:** Maintain a list of all school departments.
- **Reports:** Generate academic, financial, and administrative reports.



### 3.6.2 Quick Start Guide

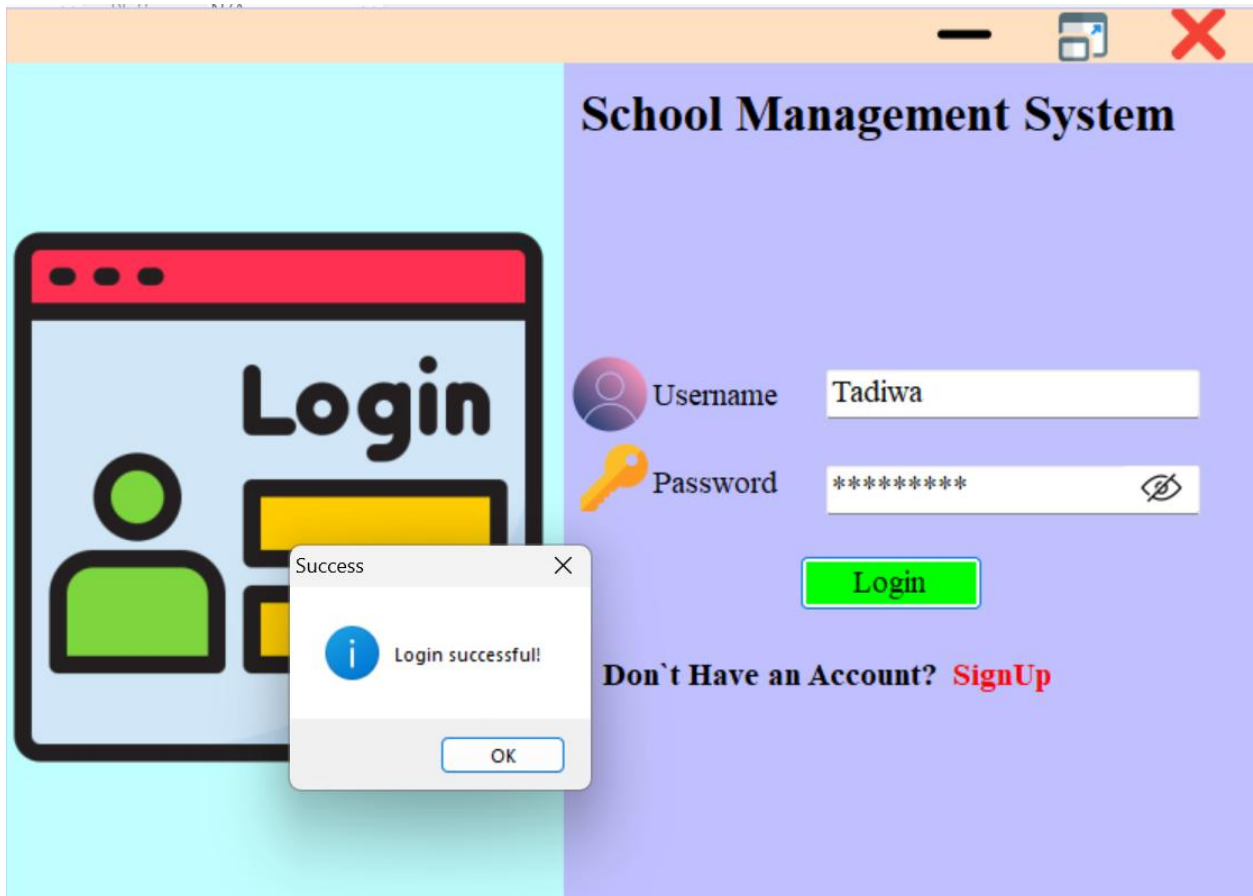
This guide will help you log in and perform a high-priority task: registering a new student.

**Step 1: Launch the System** Locate the "School Management System" icon on your desktop and double-click it.



## Step 2: Log In

- The frmLogin screen will appear.
- Enter your assigned **Username** and **Password**.
- Click the "**Login**" button.



### Step 3: Navigate to Student Registration

- Upon successful login, you will be directed to the **Dashboard**.
- In the navigation menu, click on the "**Students**" button. This will open the Student Management form.
- Click the "**Add New Student**" button to open the frmStudentAdmission form.

The screenshot shows a web application titled "Student Management". The main form area contains the following fields:

- Student ID**: Text input field.
- Phone**: Text input field with the value "078123456".
- Subjects**: Text area.
- Student Name**: Text input field.
- Admission Date**: Date picker showing "Friday , August 22, 2025".
- Date of Birth**: Date picker showing "Friday , August 22, 2025".
- Form**: Text input field.
- Gender**: Dropdown menu.
- Department**: Text input field.

On the right sidebar, there are several buttons:

- Upload Image**: Green button with a camera icon.
- Save**: Green button with a floppy disk icon.
- Delete**: Red button with a trash can icon.
- Update**: Yellow button with a circular arrow icon.
- Reload**: Purple button with a circular arrow icon.
- Search**: Blue button with a magnifying glass icon.

At the bottom, there is a table with the following columns:

	Student_ID	Student_Name	Date_of_Birth	Gender	Phone	Admission_Date	Form	Department	Subjects
*									

#### Step 4: Register the New Student

- Fill in all the required fields on the frmStudentAdmission form.
  - **Note:** The **StudentID** is auto-generated and cannot be edited.
- Enter the student's **Full Name**, **Date of Birth**, select their **Gender** and **Class** from the dropdowns, and fill in the **Address**, **Parent/Guardian Name**, and **Contact Number**.
- Click the "**Register**" button at the bottom of the form.

The screenshot shows the 'Student Management' application window. The form contains the following data:

Student ID	Phone	Subjects
idt123	07812345	Math

Student Name	Admission Date
Tadiwa	Friday , August 22, 2025

Date of Birth	Form
Friday , August 22, 2025	6

Gender	Department
Female	Science

On the right side of the form, there are several buttons: 'Upload Image', 'Register' (highlighted with an orange arrow), 'Delete', 'Update', and 'Reload'. Below these buttons is a 'Search' button.

Student_ID	Student_Name	Date_of_Birth	Gender	Phone	Admission_Date	Form	Department	Subjects
*								

A confirmation message "Student registered successfully!" will appear. The new student is now in the system.

### 3.6.3 Module-by-Module Guide

This section provides a detailed breakdown of each major system module.

#### Fees Management Module (frmFees)

This module is used to record fee payments, view outstanding balances, and generate printable receipts.

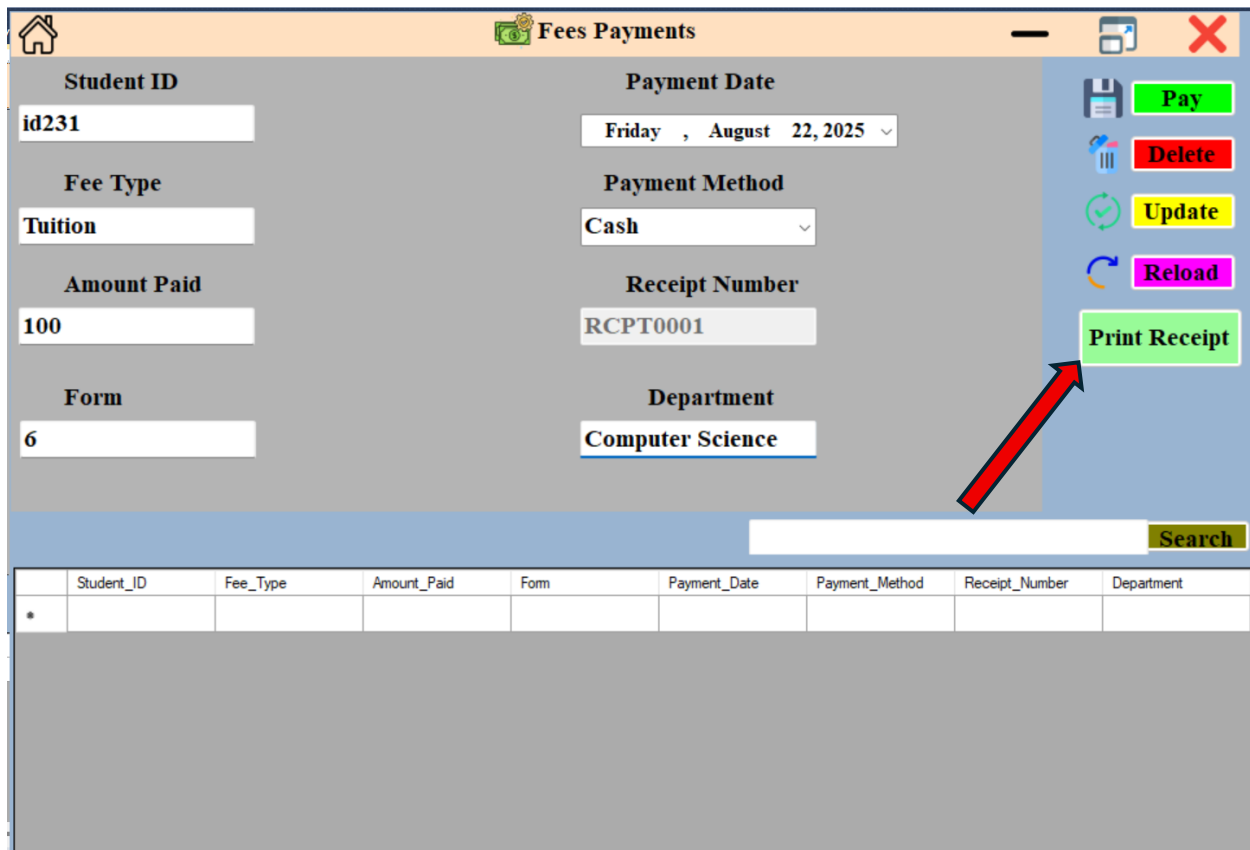
##### 1. Recording a Fee Payment

- **Locate the Student:** Use the Search bar at the top of the form to find the student by their StudentID or name.
- **Input Payment Details:** In the payment section, enter the Amount Paid.
- **Select Fee Type:** Use the Fee Type dropdown to select the type of fee being paid (e.g., "Term Fees," "Sports," "Uniforms").
- **Record Payment:** Click the "Pay" button. The system will record the payment and update the student's financial record. A success message will confirm the transaction.

	Student_ID	Fee_Type	Amount_Paid	Form	Payment_Date	Payment_Method	Receipt_Number	Department
*								

## 2. Printing a Receipt

- After a payment is successfully recorded, a "**Print Receipt**" button will become active.
- Click this button to generate a printable receipt for the parent/guardian.



The screenshot displays the 'Fees Payments' form. The form contains the following fields and values:

Student ID	Payment Date
id231	Friday , August 22, 2025

Fee Type	Payment Method
Tuition	Cash

Amount Paid	Receipt Number
100	RCPT0001

Form	Department
6	Computer Science

On the right side of the form, there is a vertical toolbar with the following buttons: Pay (green), Delete (red), Update (yellow), Reload (purple), and Print Receipt (green). A red arrow points to the 'Print Receipt' button.

Below the form is a search bar with a 'Search' button.

Student_ID	Fee_Type	Amount_Paid	Form	Payment_Date	Payment_Method	Receipt_Number	Department
*							

### 3. Generating Reports

- From the main menu, click the **"Reports"** button.
- Select the **"Fee Balance Report"** to generate a list of all students with outstanding balances. This report can be exported to a PDF or printed for administrative use.

#### 3.6.4 Troubleshooting

This section provides solutions to common issues you may encounter while using the system.

Problem	Possible Cause(s)	Solution
<b>"Invalid username or password" error</b>	Incorrect credentials entered.	Double-check your username and password for typos. Remember that passwords are case-sensitive.
<b>"Please fill in all required fields" error</b>	A mandatory field on a form was left blank.	Go back and fill in all fields marked as required (usually indicated by a red asterisk or a clear label).
<b>"Database connection failed" error</b>	The system is unable to connect to the database file.	Check that the SchoolDB.accdb file is in the correct location (C:\YourProject\SchoolDB.accdb) and that your network connection is stable. Contact your system administrator if the issue persists.
<b>The system is not responding</b>	The application may have encountered an unexpected error.	Try closing the application by clicking the 'X' button or by using the task manager (Ctrl+Shift+Esc). If the issue is persistent, report it to the technical support team.
<b>The print button for a receipt is not active</b>	A payment has not been successfully recorded for the selected student.	Ensure you have successfully processed a payment by entering all the required information and clicking the "Pay" button first.

## SECTION D: Testing and Evaluation

### 4.0 User Testing

User Acceptance Testing (UAT) is the final and most critical phase of the testing process. This is where the system is validated from the perspective of the end-users—in this case, school administrators and teachers—to ensure it meets their real-world needs and is ready for deployment. The goal of UAT is not to find bugs, but to confirm that the software's functionality and user interface are suitable for the business environment. During this phase, real-world scenarios are executed using test data that closely resembles live data. A comprehensive test plan will be developed to ensure all functionalities are thoroughly checked and verified.

#### Example User Test Cases:

To ensure the system is both functional and practical, a series of detailed test cases will be performed by end-users. Each test case has a clear objective, steps, and expected outcome.

- **Student Registration:** The objective is to verify that a new student can be successfully added to the system. The test case will involve an administrator filling out all fields on the frmStudentAdmission form. The expected outcome is for the data to be correctly saved to the tblStudents table and for the newly created student record to be searchable and accessible within the system.
- **Fee Payment Processing:** This test case validates the financial transaction process. An administrator will process a fee payment for an existing student using the frmFees module. The expected outcome is for the tblFees table to be updated correctly with the payment details and for a printable receipt to be generated with accurate information. The administrator will also check the student's record to confirm the fee balance has been updated.
- **Report Generation:** This test case ensures that the reporting functionality works as intended. An administrator will test the "Student List Report" by applying various filters, such as class, gender, or enrollment status. The expected outcome is for the system to generate an accurate, well-formatted, and easily readable report that reflects the applied filters.



## Payment of fees

The screenshot shows the 'Fees Payments' form with the following data entered:

- Student ID: ta1323
- Payment Date: Friday, August 22, 2025
- Fee Type: Tuition
- Payment Method: Cash
- Amount Paid: 100
- Receipt Number: RCPT0001
- Form: 6
- Department: Computer Science

A success message dialog box is displayed in the center: "Success Recorded successfully!" with an "OK" button.

On the right side, there are buttons for Pay, Delete, Update, Reload, and Print Receipt.


Student_ID	Fee_Type	Amount_Paid	Form	Payment_Date	Payment_Method	Receipt_Number	Department
*							

The screenshot shows the 'Fees Payments' form with the same data entered as above. The data table at the bottom is now populated with the recorded transaction.


Student_ID	Fee_Type	Amount_Paid	Form	Payment_Date	Payment_Method	Receipt_Number	Department
▶ ta1323	Tuition	100	6	Friday, August 22, 2...	Cash	RCPT0001	Computer Science
*							

An orange arrow points to the 'Form' column in the table.

## Departments



Departments Management



Department ID


MATH01


Department Name


Computer Science


Head of Department


Description

 Save


 Delete


 Update

 Reload

 Search


School Management System




 Please fill in all the required fields

OK

Department_ID	Department_Name	Description
*		



Student Management



Student ID

rt213

Phone

078219834

Subjects

Student Name

Rebeka

Admission Date

Friday , August 22, 2025

Date of Birth


Friday , August 22, 2025


Form


6


Gender


Female


 Upload Image

 Registe


 Delete


 Update

 Reload

 Search

School Management System



 Please fill in all the required fields

OK

Student_ID	Student_Name	Date_of_Birth	Gender	Phone	Admission_Date	Form	Department	Subjects
*								

## 4.1 Achievements

The implementation of the new School Management System is expected to yield significant achievements that will improve the school's overall operational efficiency. These achievements go beyond simple automation and represent a fundamental shift towards a more accurate, responsive, and data-driven administrative environment.

- **Improved Efficiency and Accuracy:** The new system will automate repetitive and time-consuming tasks, such as data entry and report generation. Tasks that once took hours to compile manually will now be available in minutes, drastically reducing the administrative burden. The centralization of data in a single database and the implementation of automated validation checks will also lead to a substantial reduction in human error.
- **Enhanced Reporting Capabilities:** The system will provide management with instant access to accurate, up-to-date reports. This empowers data-driven decision-making, enabling the school to proactively address issues such as student enrollment trends, financial balances, or attendance patterns. The ability to generate custom reports and dashboards will give administrators a powerful tool for strategic planning.
- **Increased Data Security and Integrity:** By moving away from physical files and decentralized spreadsheets, all sensitive student and staff information will be stored in a secure, centralized database with role-based access control. This significantly reduces the risk of data loss, theft, or unauthorized access and ensures the integrity and confidentiality of all records.

## 4.2 Limitations

While the system offers significant advantages, it also has certain limitations that must be acknowledged.

- **Requires Computer Literacy:** The system is designed to be intuitive and user-friendly, but a baseline level of computer literacy is required for effective use. Staff members who are unfamiliar with computers or database systems may need additional training and support during the transition period. This may present a temporary obstacle to full adoption.
- **Restricted to School Environment:** The current version of the system is a desktop application, meaning it is designed to run on a local network. It does not support remote access, which limits its use to within the physical school premises. All administrative tasks must be performed on-site.

## 4.3 Delimitations

This project has a clearly defined scope. The following features are explicitly excluded from the current development to keep the project on schedule and within budget.

- **E-learning Integration:** The system will not include any features for online classrooms, learning management, or course content delivery. Its purpose is strictly administrative.
- **Automated Communication:** The system will not have the capability to send automated SMS or email notifications to parents or students. This functionality is considered a future development opportunity and falls outside the scope of this project's initial requirements.
- **External System Integration:** The system will not be integrated with external financial systems or other third-party software. All financial data will be managed and stored internally within the system's database.

#### 4.4 Opportunities for Future Development

The current system provides a strong foundation. Based on user feedback and technological advancements, several opportunities exist for future development that could significantly enhance its value.

- **Cloud Integration:** The system could be migrated to a cloud-based platform, allowing administrators to access it from anywhere with an internet connection. This would improve flexibility and support remote work capabilities.
- **Mobile Application:** A mobile application could be developed for parents and students, allowing them to conveniently view their academic performance, attendance records, and fee status on the go.
- **Automated Notifications:** Integrating SMS and email APIs would allow the system to send automated reminders for fee deadlines, attendance alerts, or important school announcements. This would significantly enhance communication and reduce the administrative effort required for outreach.

## SECTION E: Implementation and Deployment

### 5.0 Deployment Plan

The system will be deployed using a phased rollout approach to minimize disruption and allow for a controlled transition.

**Phase 1: Pilot Deployment** The initial phase involves installing the system on a single administrative computer. This serves as a final testing environment, allowing us to confirm that the software, database, and network connectivity function correctly within the school's IT infrastructure. Feedback from this pilot user will be critical for making any final adjustments before a wider release.

**Phase 2: Full Rollout** Once the pilot is successful, the system will be deployed to all designated administrative computers. This will be done in a systematic manner, ensuring all necessary prerequisites are in place on each machine.

**Setup Requirements** Each administrative computer must meet the following technical requirements to ensure the system runs without issues:

- **Operating System:** Windows 10 or later.
- **Database:** Microsoft Access 2016 or a more recent version, as the system relies on the .accdb file format. The Access Database Engine must also be installed on client computers.
- **.NET Framework:** The computer must have the .NET Framework 4.7.2 or later to run the VB.NET application.
- **Hardware:** Minimum of 4GB RAM and 50GB of free hard drive space for the application and database files.

**Step-by-Step Deployment** The deployment process will follow the simplified, user-friendly installation wizard described in **Section C: Software Development**. A dedicated installer file will bundle all application components, including the executable and a template database file. The installer will automatically perform prerequisite checks and guide the user through the process, making deployment straightforward and consistent across all machines.

## 5.1 User Training and Support Plan

A new system is only as effective as the users who operate it. Therefore, a robust training and support plan is essential for a successful transition.

**User Training** A comprehensive training session will be conducted for all staff members who will use the system. This training will be divided into two main parts:

1. **System Navigation:** An overview of the main menu, dashboard, and general system navigation to ensure users are comfortable with the interface.
2. **Module-Specific Training:** Detailed, hands-on training for each module (e.g., Student Registration, Fee Payments, Report Generation). Users will work through practical exercises using test data to build confidence and proficiency.

**Support Plan** A dedicated support team will be available to address any immediate issues during the initial weeks of deployment. This team will serve as the first line of contact for user questions, bug reports, and technical problems. A simple ticketing system or shared email address will be established to track all support requests and ensure every issue is resolved promptly.

## 5.2 Maintenance Procedures

Long-term success depends on a proactive maintenance plan to ensure the system remains reliable, secure, and up-to-date.

**Database Backup** To prevent data loss from hardware failure, human error, or unexpected corruption, a strict backup schedule will be implemented.

- **Daily Backups:** A full backup of the main SchoolDB.accdb file will be performed automatically every night.
- **Weekly Off-site Backups:** Weekly backups will be stored on an external hard drive or a secure network location to protect against on-site disasters like fire or theft.

**Bug Fixes and Updates** A formal process will be established to handle bug reports and feature requests.

1. **Reporting:** Users will report bugs through the dedicated support channel. Each report will be logged with a unique ID, a description of the issue, and steps to reproduce it.
2. **Resolution:** The development team will investigate and address reported bugs. Once a fix is verified, a patch or an updated version of the application will be deployed to all users.

**Version Control** All source code for the School Management System will be managed using a **version control system** like Git. This allows the development team to track every change made to the code. This is crucial for:

- **Collaboration:** Multiple developers can work on the code simultaneously without overwriting each other's work.
- **Rollbacks:** If a new feature introduces a bug, the system can be easily rolled back to a previous, stable version.
- **History:** It provides a complete history of the project, making it easier to understand how and why changes were made over time.

## SECTION F: Review and Evaluation

### 6.0 Evaluation Criteria

The School Management System's effectiveness will be measured against a set of key criteria to ensure it meets all project goals.

- **Accuracy:** This criterion assesses the reliability of the data stored and generated by the system. We will verify that all reports, financial summaries, and individual records are correct and free from errors. A high level of accuracy is paramount for effective, data-driven decision-making.
- **Usability:** This measures how intuitive and easy the system is for end-users. A well-designed system should reduce the learning curve for staff. Evaluation will focus on the clarity of the user interface, the simplicity of navigation, and the efficiency of completing common tasks like student registration or fee payments.
- **Reliability:** This criterion evaluates the system's stability. We will test whether the system performs consistently without unexpected crashes, freezes, or data corruption. A reliable system is essential for uninterrupted daily administrative operations.
- **Performance:** This measures the system's speed and responsiveness. We will assess the time it takes to execute key functions, such as searching for a student record or generating a comprehensive report. Fast performance is crucial for enhancing administrative efficiency and user satisfaction.

## 6.1 Results of Evaluation

The evaluation results will be presented in a clear, comparative format to demonstrate the tangible benefits of the new system. We will compare the performance of the old manual system with the new automated system using key metrics on efficiency and error reduction.

A **bar graph** and a **table** will be included to visually represent the findings. The table will have rows for specific tasks and columns to compare the "Old System (Manual)" to the "New System (Automated)," highlighting the significant improvements.

Metric	Old System (Manual)	New System (Automated)
Time to Compile Monthly Reports	2-3 days	10 minutes
Data Entry Errors per Week	5-10 errors	Less than 1 error
Time to Find a Student Record	Up to 15 minutes	Under 10 seconds

A feedback form will be provided to administrators, teachers, and clerks to gather qualitative data on their experience. This feedback will be crucial for understanding user satisfaction and identifying areas for improvement.

## 6.2 Recommendations for Improvements

Based on the evaluation results and valuable user feedback, we will propose a list of recommendations for future enhancements. These recommendations will guide subsequent development phases to further improve the system's functionality and meet evolving needs.

- **Enhance Reporting:** While the current reporting is effective, future enhancements could include more dynamic, customizable reports with advanced filtering options.
- **Add New Modules:** The system could be expanded to include modules for managing student attendance, library resources, or e-learning integration.
- **Improve Existing Features:** Based on user feedback, minor tweaks to the user interface or specific workflows could be made to further streamline administrative tasks. For example, a bulk fee payment feature could be added.



### 6.3 Appendices

- **Questionnaires:** The full text of the questionnaires used for both "**Administrator**" and "**Student**" research will be included here.

7. How many student records do you handle daily?

.....

8. On average, how long does it take to find a specific student's record?

.....

9. How do you currently track student fee payments? (e.g., Ledger book, spreadsheet)

.....

10. How often do you need to generate reports (e.g., student list, fee balances)?

.....

11. What are the biggest challenges you face with the current system? (Select all that apply)

- ☐ Data entry is too slow
- ☐ Difficulty finding records
- ☐ Data is inaccurate
- ☐ Generating reports is difficult
- ☐ Security concerns

12. Do you believe a new automated system would improve efficiency? (Yes/No)

- ☐ Yes
- ☐ No

.

- **Sample Data:**

The image displays two screenshots of a database application interface. The top screenshot shows the 'Departments' table, and the bottom screenshot shows the 'Register' table. Both tables have a sidebar on the left with a search bar and a list of tables: Departments, Fees, Register, Students, and Teachers.

**Departments Table:**

Department	Department	HOD	Description	Click to Add
math1	Mathematics	Mr Kufa	Dept of mathe	
science01	Physics	Mr Kofapa	Dept of physic	
*				

**Register Table:**

Full_Name	ID_Number	Email	Contact	Username	Gender	Password
Tadiwa	16126g32	tadiwa@gmail.	77823712	tadi	Female	rebeka
Kupa	125j89732h	kupa@gmail.co	7827360	kupa	Male	kp123
Never	27-30943j78	never@gmail.c	78372309	Never	Male	never24
Alice	24-5362g23	alice@gmail.co	736283670	Alice	Female	leey
*			0			

- **Acknowledgments:**

I would like to sincerely thank my teacher, for their guidance and support throughout this project. I also appreciate the assistance of the staff and students at **Bradford Senior School**, who provided valuable information during the investigation and analysis stage. Lastly, I am grateful to my family for their encouragement while I worked on this project.