

# **STUDENT DATA BASE MANAGEMENT SYSTEM BY PYHTON**

A PROJECT REPORT

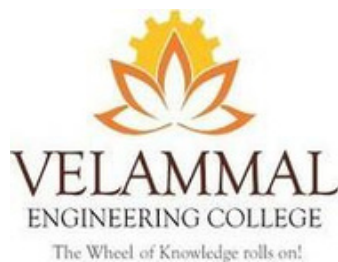
*Submitted by*

**SIVARAJ T**

**REG(113224071091)**

*In the partial fulfillment of the award of the degree of*

**BACHELOR OF TECHNOLOGY  
IN  
INFORMATION TECHNOLOGY**



**INFORMATION TECHNOLOGY**

**VELAMMAL ENGINEERING COLLEGE  
CHENNAI**



**ANNA UNIVERSITY, CHENNAI - 600 025**

**JANUARY 2025**

**VELAMMAL ENGINEERING COLLEGE  
DEPARTMENT OF INFORMATION  
TECHNOLOGY  
BONAFIDE CERTIFICATE**

Certified that this project report titled “**STUDENT DATA BASE MANAGEMENT SYSTEM**” is the bonafide work of **Mr.SIVARAJ T( 113224071091 )** who carried out the Mini project work under my supervision.

**SIGNATURE**

**INTERNAL GUIDE,**

**Mrs.RANGINI M**

**Assistant Professor,**

**INFORMATION TECHNOLOGY**

**Velammal Engineering College,**

**Chennai - 600066.**

**SIGNATURE**

**HEAD OF THE DEPARTMENT,**

**Dr.JEEVAA KATHIRAVAN,**

**Professor and Head,**

**INFORMATION TECHNOLOGY,**

**Velammal Engineering College,**

**Chennai - 600066.**

# **MINI PROJECT EXAMINATION**

The MINI PROJECT Examination of this project work “**STUDENT DATA BASE MANAGEMENT SYSTEM**” is a bonafide record of project done at the department of Information Technology, Velammal Engineering College during the academic year 2024-2025 by

**SIVARAJ T**

**( 113224071091)**

Of first year Bachelor of Technology in information technology  
submitted for the university examination held on.....

**INTERNAL EXAMINER**

## **ACKNOWLEDGEMENT**

Behind every achievement lies an unfathomable sea of gratitude to those who achieved it, without whom it would ever have come into existence. To them we express our words of gratitude. We give all the glory and thanks to our almighty GOD for showering upon, the necessary wisdom and grace for accomplishing this project. We express our gratitude and thanks to Our Parents first for giving health and sound mind for completing this project. First of all, we would like to express our deep gratitude to our beloved and respectable Chairman Thiru M.V. Muthuramalingam and our Chief Executive Officer Thiru M.V.M. Velmurugan for their kind encouragement. We express our deep gratitude to Dr. S. Satish Velammal Engineering College for his helpful attitude towards this project. We wish to express our sincere thanks and gratitude to Dr.Jeevaa kathiravan, Professor and Head of the Department, Department of Information Technology for motivating and encouraging in every part of our project. We express our sincere gratitude to the Mrs.RANGINI M Assistant Professor, Department of Information Technology for their invaluable guidance in shaping of this project without them this project would not have been possible. Our thanks to all other Faculty and Non-Teaching staff members of our department for their support and peers for having stood by me and helped me to complete this project.

# AGENDA

S.NO	TITILE	PAGE.NO
1	ABSTRACT	1
2	INTRODUCTION	2
3	PROJECT OUTLINE	4
4	TOOLS AND PLATFORMS	5
5	ALGORITHM	7
6	PSEUDO CODE	9
7	SOURCE CODE	13
8	OUTPUT	17
9	CONCLUSION	21

# **ABSTRACT**

This program is designed to manage a student database system, allowing users to perform various operations such as adding, viewing, removing, and updating student records. The system stores student information like ID, name, department, section, year, and year of joining. It provides the following functionalities:

1. Add Student: Prompts the user to input details for a new student, which are then stored in the system. The program ensures that duplicate student IDs are not added.
2. View Student: Allows the user to search for and view a student's details by their unique ID.
3. Remove Student: Facilitates the removal of a student's record from the system based on their ID.
4. View All Students: Displays a list of all students stored in the system.
5. Update Data in Text File: Saves the current state of the student data to a text file, enabling persistent storage.
6. View Data from Text File: Displays the content of the text file where student data has been stored. The program includes a simple menu system that allows users to interact with the database. Error handling is incorporated to ensure smooth operation and prevent the system from crashing unexpectedly. This tool can be useful for managing student records in educational institutions or any environment requiring the organization and tracking of student data.

# **INTRODUCTION**

## **Student Data Base Management System By Python:**

The Student Database Management Program is a simple yet efficient application designed to manage and organize student records. It provides an interactive interface for users to add, view, update, and remove student information in a structured and user-friendly manner. The program allows users to input and track key details about each student, such as their ID, name, department, section, year of study, and year of joining.

### **Objective:**

The primary objectives of the Student Database Management Program are:

1. **Efficient Student Record Management:** To provide a system for managing student records that allows for the easy addition, viewing, updating, and removal of student data, ensuring an organized database.
2. **Data Persistence:** To enable the saving of student data to a text file, ensuring that the information is preserved between sessions, and can be accessed or updated as needed.
3. **Simplify Data Retrieval:** To offer a simple method for users to search for and retrieve student records by their unique ID, allowing quick access to individual student information.
4. **Avoid Duplicate Entries:** To ensure the integrity of the student database by preventing the entry of duplicate student IDs, thus maintaining uniqueness and avoiding confusion.
5. **User-Friendly Interface:** To provide an intuitive, menu-driven

interface that makes it easy for users (such as school administrators or staff) to navigate through the different options and interact with the system without requiring technical expertise. 6. Data Deletion: To allow for the efficient removal of student records from the database when necessary, keeping the data up to date and relevant. 7. Error Handling and Stability: To ensure the program runs smoothly without crashing, providing clear messages in case of unexpected errors or invalid user inputs, improving overall system reliability.

By achieving these objectives, the program aims to streamline the process of managing student records, reduce administrative workload, and improve accuracy in tracking student data.

## **Scope:**

The scope of this program is to efficiently manage student records by allowing users to add, view, update, and remove student data, while providing persistent storage in a text file. It is designed for educational institutions and organizations needing a simple, text- based solution for student data management.



# **PROJECT OUTLINE**

## **I. Project Overview**

- Title: Student Database Management System
- Description: Manage student data
- Goals: User-friendly, secure, efficient

## **II. System Requirements**

- Functional: User registration, student data management, search, reporting
- Non-Functional: Security, scalability, usability, maintainability

## **III. System Design**

- Architecture: Front-end, back-end, database
- Components: User interface, database management, server-side scripting

## **IV. Implementation**

- Programming Languages: Python, SQL, HTML, CSS, JavaScript
- Development Frameworks: Flask, Django, Bootstrap, React

## **V. Testing, Deployment, Maintenance**

- Testing: Unit, integration, system, acceptance
- Deployment: Cloud, on-premise

# **TOOLS AND PLATFORMS**

## **I .Programming Languages**

1. Python: For backend development and scripting.
2. SQL: For database management and querying.

## **II .Database Management Systems**

1. MySQL: A popular open-source relational database management system.
2. SQLite: A lightweight, self-contained relational database management system.

## **III .Frontend Development**

1. HTML/CSS: For creating user interfaces and styling.
2. JavaScript: For client-side scripting and dynamic functionality.

## **IV .Integrated Development Environments (IDEs)**

1. PyCharm: A popular IDE for Python development.
2. Visual Studio Code: A lightweight, versatile IDE for multiple programming languages.

## **V .Text Editors**

1. Sublime Text: A feature-rich text editor for coding.
2. Atom: A customizable, open-source text editor.

## VI .Version Control Systems

1. Git: A popular version control system for tracking
2. GitHub: A web-based platform for hosting and collaborating on Git repositories.

## VII .Operating Systems

1. Windows: A popular operating system for development and deployment.
2. Linux: A versatile, open-source operating system for development and deployment.
3. macOS: A proprietary operating system for development and deployment.

## VII .Other Tools

1. pip: A package installer for Python.
2. virtualenv: A tool for creating isolated Python environments.
3. SQLite Studio: A graphical user interface for SQLite database management.
4. MySQL Workbench: A graphical user interface for MySQL database management.

# ALGORITHM

Step 1: Display the main menu with options to Add Student, View Student, Remove Student, View All Students Data, Update Students Data to Text File, View Students Data in Text File, and Step 2: Ask the user to enter their choice (1-7).

Step 3: Add Student

- Ask the user to enter the student's ID.
- Check if a student with the same ID already exists in the database.
- If the ID is unique, ask the user to enter the student's name, department, section, year, and year of joining.
- Create a new student dictionary with the entered data.
- Add the student dictionary to the students list.
- Display a success message.

Step 4: View Student

- Ask the user to enter the student's ID.
- Search for the student with the entered ID in the students list.
- If found, display the student's details.
- If not found, display an error message.

Step 5: Remove Student

- Ask the user to enter the student's ID. Search for the student with the entered ID in the students list.
- If found, remove the student from the list.
- Display a success message. ☒ If not found, display an error message.

#### Step 6: View All Students Data

- Iterate through the students list.
- Display each student's details.

#### Step 7: Update Students Data to Text File

- Open a text file named "DATA.txt" in append
- Convert the students list to a string and write it
- Close the file.
- Display a success message.

#### Step 8: View Students Data in Text File

- Open the "DATA.txt" file in read mode.
- Read the contents of the file.
- Display the students data.
- Close the file.
- Display a goodbye message.
- Exit the program.

# PSEUDO CODE

## Main Menu PROCEDURE

mainMenu()

DISPLAY "Student Database Menu:"

DISPLAY "1. Add Student"

DISPLAY "2. View Student"

DISPLAY "3. Remove Student"

DISPLAY "4. View All Students Data"

DISPLAY "5. Update Students Data to Text

DISPLAY "6. View Students Data in Text File"

DISPLAY "7. Exit"

INPUT choice

RETURN choice

## Add Student PROCEDURE

addStudent()

INPUT studentID

IF studentID EXISTS IN students THEN

DISPLAY "A student with this ID already

INPUT name, department, section, year,  
joiningYear

CREATE student RECORD WITH studentID,

```
name, department, section, year, joiningYear
    ADD student TO students DISPLAY "Student
    added successfully."
END IF
```

## View Student PROCEDURE

```
viewStudent()
    INPUT studentID
    FOR EACH student IN students DO
        IF studentID MATCHES student ID THEN
            DISPLAY student DETAILS
        RETURN
    END IF
END FOR
DISPLAY "Student not found."
```

## Remove Student PROCEDURE

```
removeStudent()
    INPUT studentID
    FOR EACH student IN students DO
        IF studentID MATCHES student ID THEN
            REMOVE student FROM students
            DISPLAY "Student removed successfully."
        RETURN
    END IF
END FOR
```

```
DISPLAY "Student not found."
```

## View All Students Data

```
PROCEDURE viewAllStudents()  
  FOR EACH student IN students DO  
    DISPLAY student DETAILS  
  END FOR
```

## Update Students Data to Text File

```
PROCEDURE updateTextFile()  
  OPEN FILE "DATA.txt" IN APPEND MODE  
  WRITE students TO FILE  
  CLOSE FILE  
  DISPLAY "Data updated successfully."
```

## View Students Data in Text File

```
PROCEDURE viewTextFile()  
  OPEN FILE "DATA.txt" IN READ MODE  
  READ FILE CONTENTS  
  DISPLAY FILE CONTENTS  
  CLOSE FILE
```

## Main Program

```
PROCEDURE main()
```



```
WHILE TRUE DO
    choice = mainMenu() IF
    choice == 1 THEN
        addStudent()
    ELSE IF choice == 2 THEN
        viewStudent()
    ELSE IF choice == 3 THEN
        removeStudent()
    ELSE IF choice == 4 THEN
        viewAllStudents()
    ELSE IF choice == 5 THEN
        updateTextFile()
    ELSE IF choice == 6 THEN
        viewTextFile()
    ELSE IF choice == 7 THEN
        DISPLAY "Exiting program."
        BREAK
    END IF
END WHILE
```

# SOURCE CODE

```
students = []
def add_student():
    student_id = input("Enter student ID: ")
    if any(student["id"] == student_id for
student in students):
        print("A student with this ID already
exists.\n")
        return

    name = input("Enter student name: ")
    department=input("Enter student
department: ")
    section=input("Enter student section: ")
    year=input("Enter year of studying: ")
    stu_join=input("Enter student joining
year: ")

    student = {"id": student_id, "name":
name,"department":department,"section":s
ection,"year":year,"year of joing":stu_join}
students.append(student)
print(f"Student {name} with ID
{student_id} added successfully.\n")

def view_student():
```

```

student_id = input("Enter the student ID to view details: ")
for student in students:
    if student["id"] == student_id:
        print("\nStudent Details:")
        print(f"ID           : {student['id']}")
        print(f"Name          : {student['name']}")
        print(f"Department     : {student['department']}")
        print(f"Section       : {student['section']}")
        print(f"Year          : {student['year']}")
        print(f"Year of Joining : {student['year of joining']}")
        return
print("Student not found.\n")

def remove_student():
    student_id = input("Enter the student ID to remove: ")
    for student in students:
        if student["id"] == student_id:
            students.remove(student)
            print(f"Student with ID {student_id} has been removed
successfully.\n")
            return
    print("Student not found.\n")

def remove_student():
    student_id = input("Enter the student ID to remove: ")
    for student in students:
        if student["id"] == student_id:
            students.remove(student)

```

```

print(f"Student with ID {student_id} has been removed
successfully.\n")
    return
    print("Student not found.\n")
def alldata():
    for student in students:
        print("\nStudent Details:")
        print(f"ID          : {student['id']}")
        print(f"Name           : {student['name']}")
        print(f"Department      : {student['department']}")
        print(f"Section         : {student['section']}")
        print(f"Year            : {student['year']}")
        print(f"Year of Joining : {student['year of joining']}")

def display_menu():
    """Display the main menu."""
    print("Student Database Menu:")
    print("1. Add Student")
    print("2. View Student")
    print("3. Remove Student")
    print("4. To View All Students Data")
    print("5. To Update the Students Data into Text file")
    print("6. To View Students Data in the Text File")
    print("7. Exit")
def update():
    f=open("DATA.txt","a+")
    b=str(students)
    f.write(b)
    print(f.read())
    f.close()

```

```

print("DATA UPDATED TO TEXT DOCUMENT
SUCCESSFULLY")
def viewdata():
    f=open("DATA.txt","r")
    print(f.read())
    f.close()

def main():
    while True:
        display_menu()
        choice = input("Enter your choice (1-7): ").strip()
        if choice == "1":
            add_student()
        elif choice == "2":
            view_student()
        elif choice == "3":
            remove_student()
        elif choice == "4":
            alldata()
        elif choice == "5":
            update()
        elif choice == "6":
            viewdata()
        elif choice == "7":
            print("Exiting the Student Database. Goodbye!")
            break
        else:
            print("Invalid choice! Please select an option between 1 and 7.\n")

if __name__ == "__main__":
    try:
        main()
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

```

# OUTPUT

Student Database Menu:

1. Add Student
2. View Student
3. Remove Student
4. To View All Students Data
5. To Update the Students Data into Text file
6. To View Students Data in the Text File
7. Exit

Enter your choice (1-7): 1

Enter student ID: 1

Enter student name: SIVARAJ

Enter student department: IT

Enter student section: A

Enter year of studying: 1

Enter student joining year: 2024

Student SIVARAJ with ID 1 added successfully.

Student Database Menu:

1. Add Student
2. View Student
3. Remove Student
4. To View All Students Data
5. To Update the Students Data into Text file
6. To View Students Data in the Text File
7. Exit

Enter your choice (1-7): 1

Enter student ID: 2

Enter student name: KISHORE

Enter student department: IT

Enter student section: A

Enter year of studying: 1

Enter student joining year: 2024

Student KISHORE with ID 2 added successfully

Student Database Menu:

1. Add Student
2. View Student
3. Remove Student
4. To View All Students Data
5. To Update the Students Data into Text file
6. To View Students Data in the Text File
7. Exit

Enter your choice (1-7): 2

Enter the student ID to view details: 1

.

# OUTPUT

Student Details:

ID : 1

Name : SIVARAJ

Department : IT

Section : A

Year : 1

Year of Joining : 2024

Student Database Menu:

1. Add Student

2. View Student

3. Remove Student

4. To View All Students Data

5. To Update the Students Data into Text file

6. To View Students Data in the Text File

7. Exit

Enter your choice (1-7): 3

Enter the student ID to remove: 9

Student not found.

Student Database Menu:

1. Add Student

2. View Student

3. Remove Student

4. To View All Students Data

5. To Update the Students Data into Text file

6. To View Students Data in the Text File

7. Exit

Enter your choice (1-7): 4

Student Details:

ID : 1

Name : SIVARAJ

Department : IT

Section : A

Year : 1

Year of Joining : 2024

Student Details:

ID : 2

Name : KISHORE

Department : IT

Section : A

Year : 1

Year of Joining : 2024


# OUTPUT

Student Database Menu:

1. Add Student
2. View Student
3. Remove Student
4. To View All Students Data
5. To Update the Students Data into Text file
6. To View Students Data in the Text File
7. Exit

Enter your choice (1-7): 5

DATA UPDATED TO TEXT DOCUMENT SUCCESSFULLY



```
File Edit Format View Help
[{'id': '1', 'name': 'SIVARAJ', 'department': 'IT', 'section': 'A', 'year': '1', 'year of joining': '2024'},
 {'id': '2', 'name': 'KISHORE', 'department': 'IT', 'section': 'A', 'year': '1', 'year of joining': '2024'},
 {'id': '3', 'name': 'JEFFY', 'department': 'IT', 'section': 'A', 'year': '1', 'year of joining': '2024'}]
```



# OUTPUT

Student Database Menu:

1. Add Student
2. View Student
3. Remove Student
4. To View All Students Data
5. To Update the Students Data into Text file
6. To View Students Data in the Text File
7. Exit

Enter your choice (1-7): 6

```
[{'id': '1', 'name': 'SIVARAJ', 'department': 'IT', 'section': 'A', 'year': '1', 'year of joining': '2024'}, {'id': '2', 'name': 'KISHORE', 'department': 'IT', 'section': 'A', 'year': '1', 'year of joining': '2024'}, {'id': '3', 'name': 'JEFFY', 'department': 'IT', 'section': 'A', 'year': '1', 'year of joining': '2024'}]
```

Student Database Menu:

1. Add Student
2. View Student
3. Remove Student
4. To View All Students Data
5. To Update the Students Data into Text file
6. To View Students Data in the Text File
7. Exit

Enter your choice (1-7): 7

Exiting the Student Database. Goodbye!

# CONCLUSION

The Student Database Management System is a simple console-based application designed to manage student records. It allows users to add, view, remove, and update student records, as well as store and retrieve data from a text file.

## Key Features

1. **\*Student Record Management\***: Add, view, remove, and update student records.
2. **\*Data Storage\***: Store student records in a text file.
3. **\*Data Retrieval\***: Retrieve student records from the text file.
4. **\*User-Friendly Interface\***: Simple console-based interface for easy navigation. Benefits
  1. **\*Easy Record Keeping\***: Simplifies student record management for educational institutions.
  2. **\*Data Security\***: Stores data in a text file, providing a basic level of security.
  3. **\*Scalability\***: Can be easily modified to accommodate a large number of student records.

## Limitations

1. **\*Data Validation\***: Does not include robust data validation checks.

2. **\*Security\***: Does not provide advanced security features to protect sensitive data. 3. **\*User Authentication\***: Does not include user authentication mechanisms.

### Future Enhancements

1. **\*Implement Data Validation\***: Add robust data validation checks to ensure data integrity.
2. **\*Enhance Security\***: Implement advanced security features, such as encryption and access controls.
3. **\*Add User Authentication\***: Integrate user authentication mechanisms to ensure only authorized users can access the system. Overall, the

### Student Database Management

System provides a basic yet functional solution for managing student records. With future enhancements, it can become a more robust and secure system for educational institutions.

**\*\*\* THANK YOU \*\*\***