1. Project Description

Title: Student Record Management System
Description:
A Python-based system that manages student records using file handling. It allows users to add, view, search, update, and delete records, showcasing practical applications of CSV file operations and Python programming.

2. Key Features

- **File Handling:** Persistent storage of student data using CSV files.
- **Menu-Driven Interface:** User-friendly interface for seamless interaction.
- **Data Operations:** Includes CRUD functionalities:
  - Create new student records.
  - Read/view all records.
  - Update existing records.
  - Delete records.
- **Error Handling:** Manages file not found and invalid data scenarios gracefully.

3. Technologies Used

List the technologies and tools used:

- Programming Language: Python
- Modules: csv
- File Format: CSV

4. Code Snippets or Highlights

Include snippets of important code sections, like adding or updating records. For example:

Code:

```
def add_student_record(filename):
    with open(filename, mode='a', newline='') as file:
        writer = csv.writer(file)
        student_id = input("Enter Student ID: ")
        name = input("Enter Name: ")
        marks = input("Enter Marks: ")
        writer.writerow([student_id, name, marks])
        print("Record added successfully!"
```

student Record Management System.py

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

RAM Disk

Gemini

File Handling Using Python

[ ] Start coding or generate with AI.

```python
# Student Record Management System

import csv

def display_menu():
    print("\n=== Enhanced Student Record Management System ===")
    print("1. Add Student Record")
    print("2. View All Records")
    print("3. Search for a Student")
    print("4. Update a Record")
    print("5. Delete a Record")
    print("6. Exit")
```

1m 3s completed at 10:12 AM

student Record Management System.py

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

RAM Disk

Gemini

```python
        marks = input("Enter Marks: ")
        writer.writerow([student_id, name, marks])
        print("Record added successfully!")

def view_all_records(filename):
    try:
        with open(filename, mode='r') as file:
            reader = csv.reader(file)
            print("\n=== All Student Records ===")
            print(f"{'ID':<10}{'Name':<20}{'Marks':<10}")
            print("-" * 40)
            for row in reader:
                if row:  # Skip empty rows
                    print(f"{row[0]:<10}{row[1]:<20}{row[2]:<10}")
    except FileNotFoundError:
        print("No records found. Add a student first.")

def search_student(filename):
    search_id = input("Enter Student ID to search: ")
    try:
        with open(filename, mode='r') as file:
```

1m 3s completed at 10:12 AM

student Record Management System.py

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

RAM Disk

Gemini

```python
    except FileNotFoundError:
        print("No records found. Add a student first.")

def update_record(filename):
    search_id = input("Enter Student ID to update: ")
    records = []
    found = False
    try:
        with open(filename, mode='r') as file:
            reader = csv.reader(file)
            records = list(reader)
        for record in records:
            if record and record[0] == search_id:
                print(f"Current Record: ID: {record[0]}, Name: {record[1]}, Marks: {record[2]}")
                record[1] = input("Enter new Name: ")
                record[2] = input("Enter new Marks: ")
                found = True
        if found:
            with open(filename, mode='w', newline='') as file:
                writer = csv.writer(file)
```

1m 3s completed at 10:12 AM

student Record Management System.py

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

RAM Disk

Gemini

```
def delete_record(filename):
    search_id = input("Enter Student ID to delete: ")
    records = []
    found = False
    try:
        with open(filename, mode='r') as file:
            reader = csv.reader(file)
            records = list(reader)
        updated_records = [record for record in records if record and record[0] != search_id]
        if len(updated_records) < len(records):
            found = True
        with open(filename, mode='w', newline='') as file:
            writer = csv.writer(file)
            writer.writerows(updated_records)
        if found:
            print("Record deleted successfully!")
        else:
            print("Student not found.")
    except FileNotFoundError:
        print("No records found. Add a student first.")
```

✓ 1m 3s completed at 10:12 AM

student Record Management System.py ☆

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

RAM Disk

Gemini

```
        choice = input("Enter your choice: ")
        if choice == '1':
            add_student_record(filename)
        elif choice == '2':
            view_all_records(filename)
        elif choice == '3':
            search_student(filename)
        elif choice == '4':
            update_record(filename)
        elif choice == '5':
            delete_record(filename)
        elif choice == '6':
            print("Exiting the program. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

✓ 1m 3s completed at 10:12 AM

student Record Management System.py ☆

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

RAM Disk

Gemini

```
Enter your choice: 6
Exiting the program. Goodbye!
```