Hostel Management System

Project Report

---

1. Cover Page

Hostel Management System

College Project Report

Submitted by: Bhavishya Gupta

Course: BTech. CSE Core

College: VIT Bhopal

Date: 19th November, 2025

---

2. Introduction

The Hostel Management System is a simple Python-based application designed to help hostel administrators efficiently manage student accommodation details, including registration, room allocation, and fee status. This system reduces manual errors and streamlines administrative tasks.

---

3. Problem Statement

Managing student records manually in a hostel environment is time-consuming and error-prone. This project aims to automate the record-keeping process by providing a system that stores student information securely and allows easy addition, viewing, and removal of student data.

---

4. Functional Requirements

- Add new student records with registration number, name, branch, room number, and fee status.
- View the list of all students currently registered.
- Remove a student record by registration number.
- View statistics about total students, fee payments completed, and pending fees.
- Persistent storage of data in CSV files.

---

5. Non-functional Requirements

- The system should be easy to use via a command-line interface.
- Data storage must be persistent across sessions using CSV files.
- The system must handle errors gracefully (e.g., duplicate entries, missing records).
- The software should be lightweight and require no additional dependencies beyond standard Python libraries.

---

6. System Architecture

The system is structured around two main components:

- **Student Class:** Represents individual student data.
- **HostelManager Class:** Manages student records, handles file operations, and provides the user interface through CLI menus.

The application interacts with the file system for persistent CSV storage and provides a menu-driven interface for user interaction.

---

7. Design Diagrams

Use Case Diagram

- Actors: Hostel Administrator
- Use Cases: Add Student, View Students, Remove Student, View Statistics, Exit

Workflow Diagram

- Start → Display Menu → User Chooses Option → Perform Action → Loop Back or Exit

Sequence Diagram

- User → HostelManager: select option
- HostelManager → Student: add/view/remove student
- HostelManager → CSV File: read/write data

Class Diagram

- Classes:
    - Student: Attributes (reg, name, branch, room, feestat)
    - HostelManager: Methods (addstudent, viewall, removestudent, stats, save, load)

ER Diagram

Not applicable (CSV file-based storage without relational database).

---

8. Design Decisions & Rationale

- **CSV File Storage:** Chosen for simplicity and ease of use without requiring a database setup.
- **Command-line Interface:** Enables wide compatibility without GUI dependencies.
- **Data Validation:** Ensures no duplicate student registration numbers are allowed.
- **Modular Design:** Separation of concerns between data model (Student class) and management logic (HostelManager class).

---

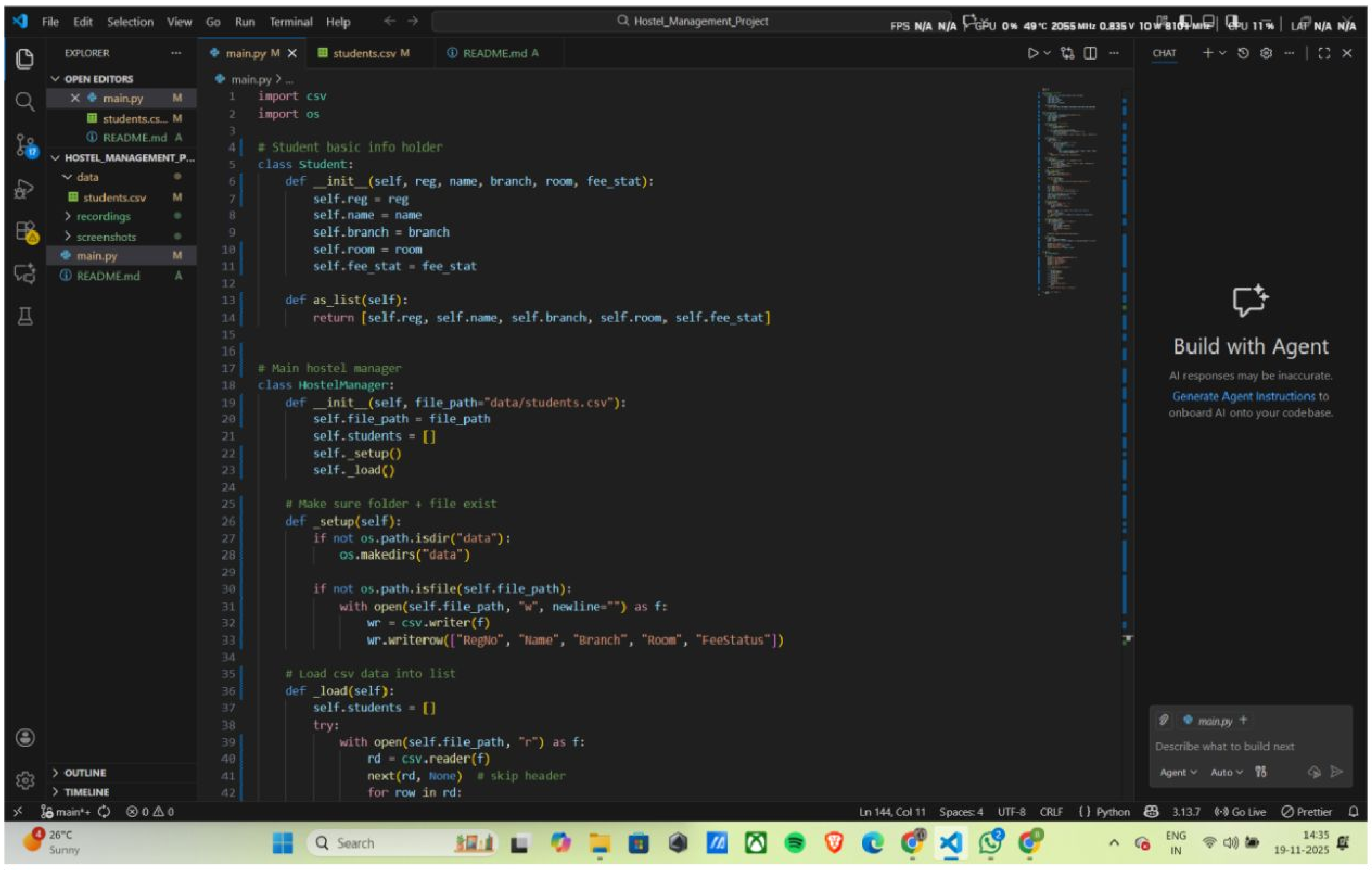
9. Implementation Details

The project is implemented in Python using standard libraries (csv and os).

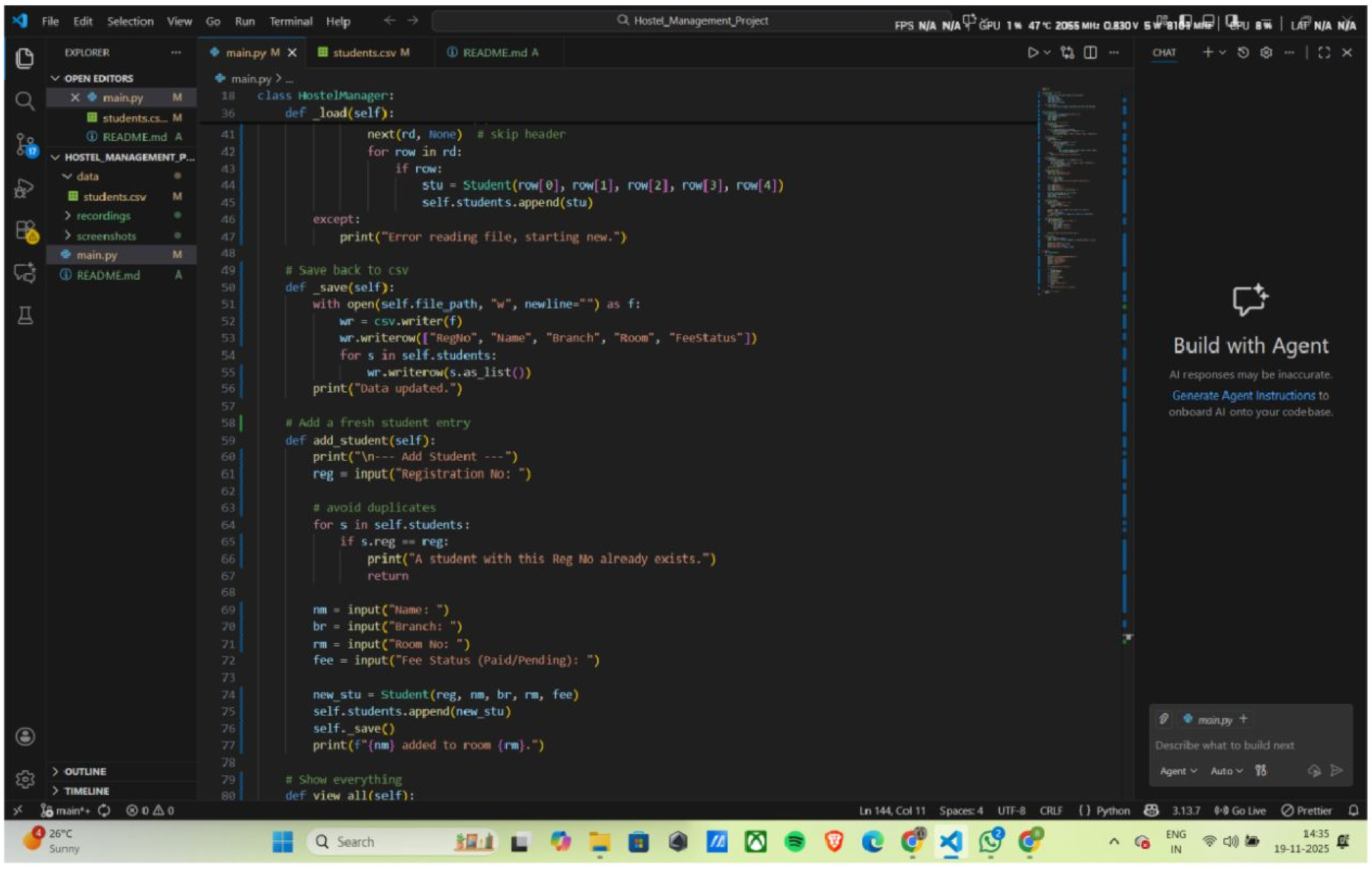
- The Student class encapsulates student attributes and provides a method to convert data into list form for CSV writing.
- The HostelManager class manages CRUD operations, handles file reading/writing, and provides the user menu for interaction.
- Data persistence is handled by reading from and writing to data/students.csv.
- The program ensures the data directory and CSV file exist, creating them if necessary.

---

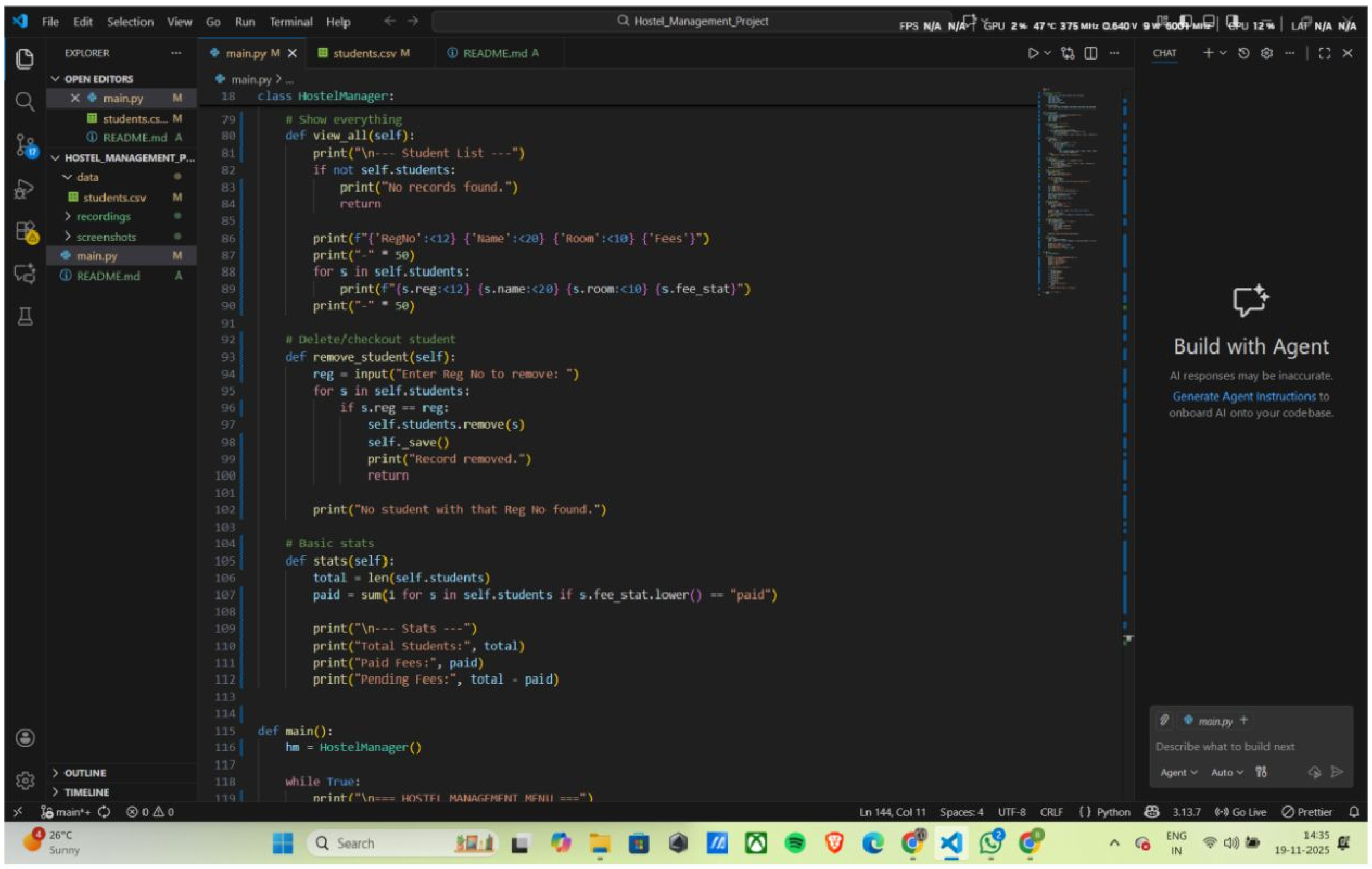
10. Screenshots / Results



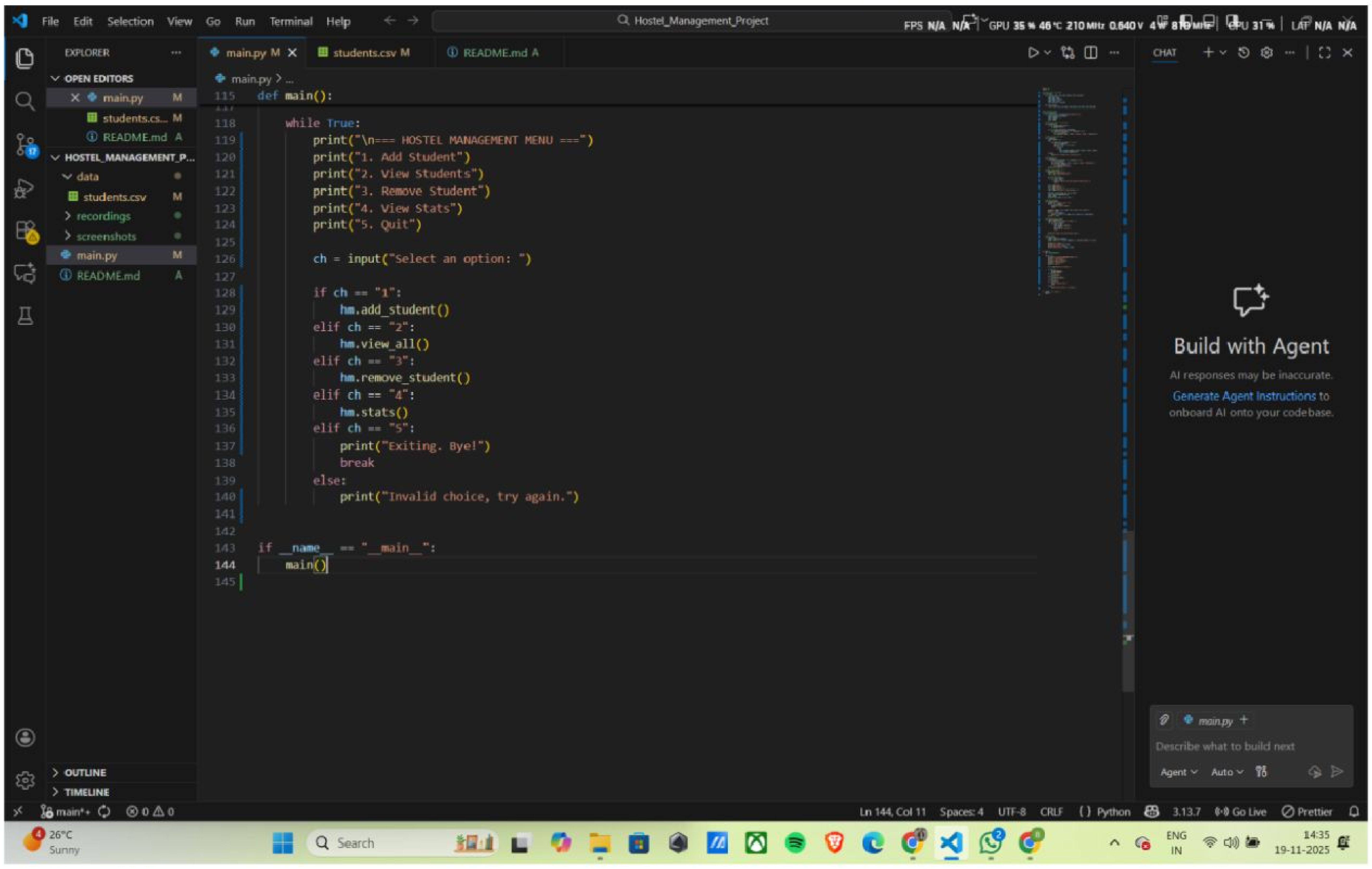
```python
import csv
import os

# Student basic info holder
class Student:
    def __init__(self, reg, name, branch, room, fee_stat):
        self.reg = reg
        self.name = name
        self.branch = branch
        self.room = room
        self.fee_stat = fee_stat

    def as_list(self):
        return [self.reg, self.name, self.branch, self.room, self.fee_stat]


# Main hostel manager
class HostelManager:
    def __init__(self, file_path="data/students.csv"):
        self.file_path = file_path
        self.students = []
        self._setup()
        self._load()

    # Make sure folder + file exist
    def _setup(self):
        if not os.path.isdir("data"):
            os.makedirs("data")

        if not os.path.isfile(self.file_path):
            with open(self.file_path, "w", newline="") as f:
                wr = csv.writer(f)
                wr.writerow(["RegNo", "Name", "Branch", "Room", "FeeStatus"])

    # Load csv data into list
    def _load(self):
        self.students = []
        try:
            with open(self.file_path, "r") as f:
                rd = csv.reader(f)
                next(rd, None)  # skip header
                for row in rd:
```



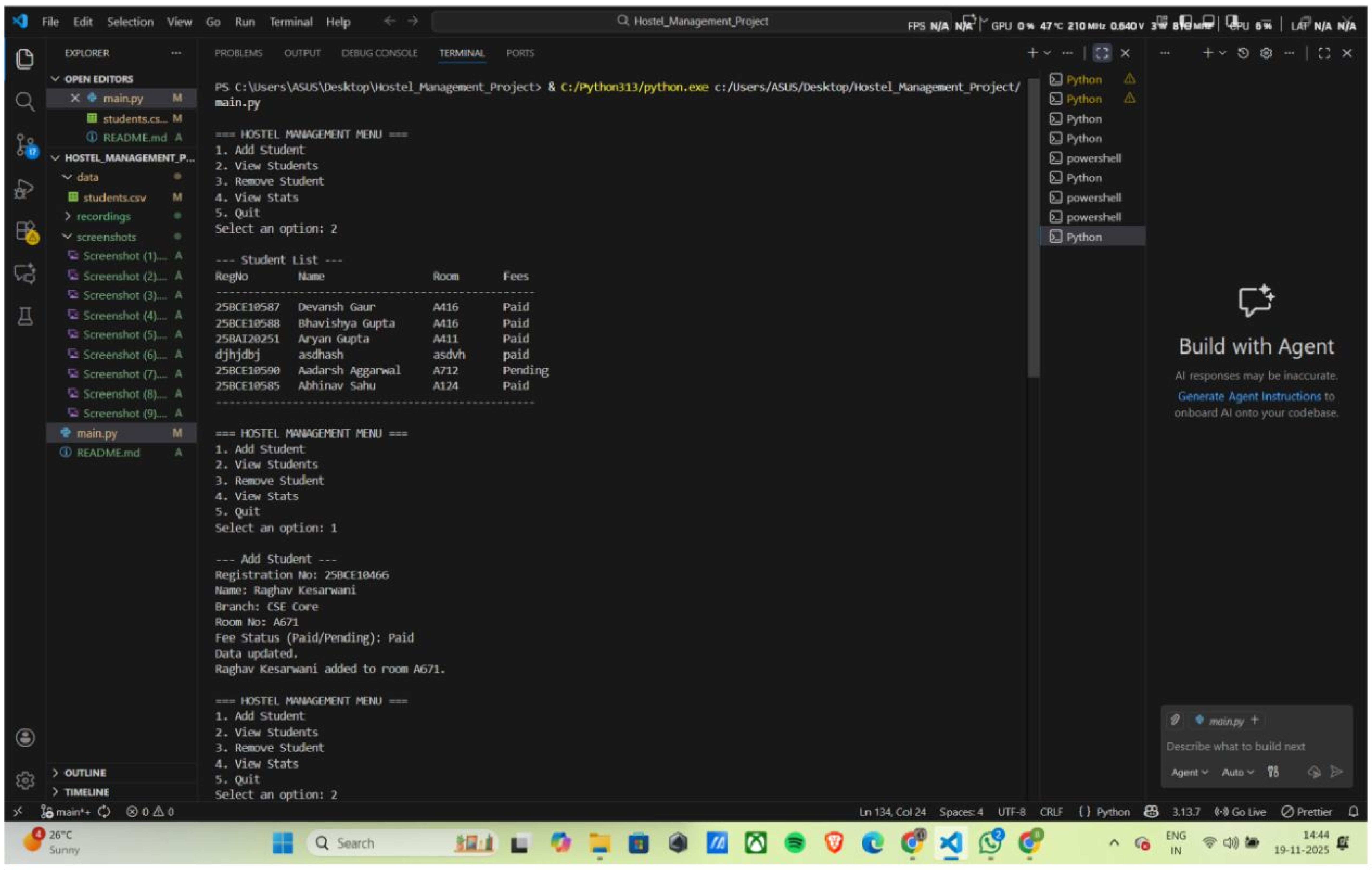
```python
class HostelManager:
    def _load(self):
                next(rd, None)  # skip header
                for row in rd:
                    if row:
                        stu = Student(row[0], row[1], row[2], row[3], row[4])
                        self.students.append(stu)
        except:
            print("Error reading file, starting new.")

    # Save back to csv
    def _save(self):
        with open(self.file_path, "w", newline="") as f:
            wr = csv.writer(f)
            wr.writerow(["RegNo", "Name", "Branch", "Room", "FeeStatus"])
            for s in self.students:
                wr.writerow(s.as_list())
        print("Data updated.")

    # Add a fresh student entry
    def add_student(self):
        print("\n--- Add Student ---")
        reg = input("Registration No: ")

        # avoid duplicates
        for s in self.students:
            if s.reg == reg:
                print("A student with this Reg No already exists.")
                return

        nm = input("Name: ")
        br = input("Branch: ")
        rm = input("Room No: ")
        fee = input("Fee Status (Paid/Pending): ")

        new_stu = Student(reg, nm, br, rm, fee)
        self.students.append(new_stu)
        self._save()
        print(f"{nm} added to room {rm}.")

    # Show everything
    def view_all(self):
```


Hostel_Management_Project
main.py M
students.csv M
README.md A
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Describe what to build next
Ln 144, Col 11 Spaces: 4 UTF-8 CRLF Python 3.13.7

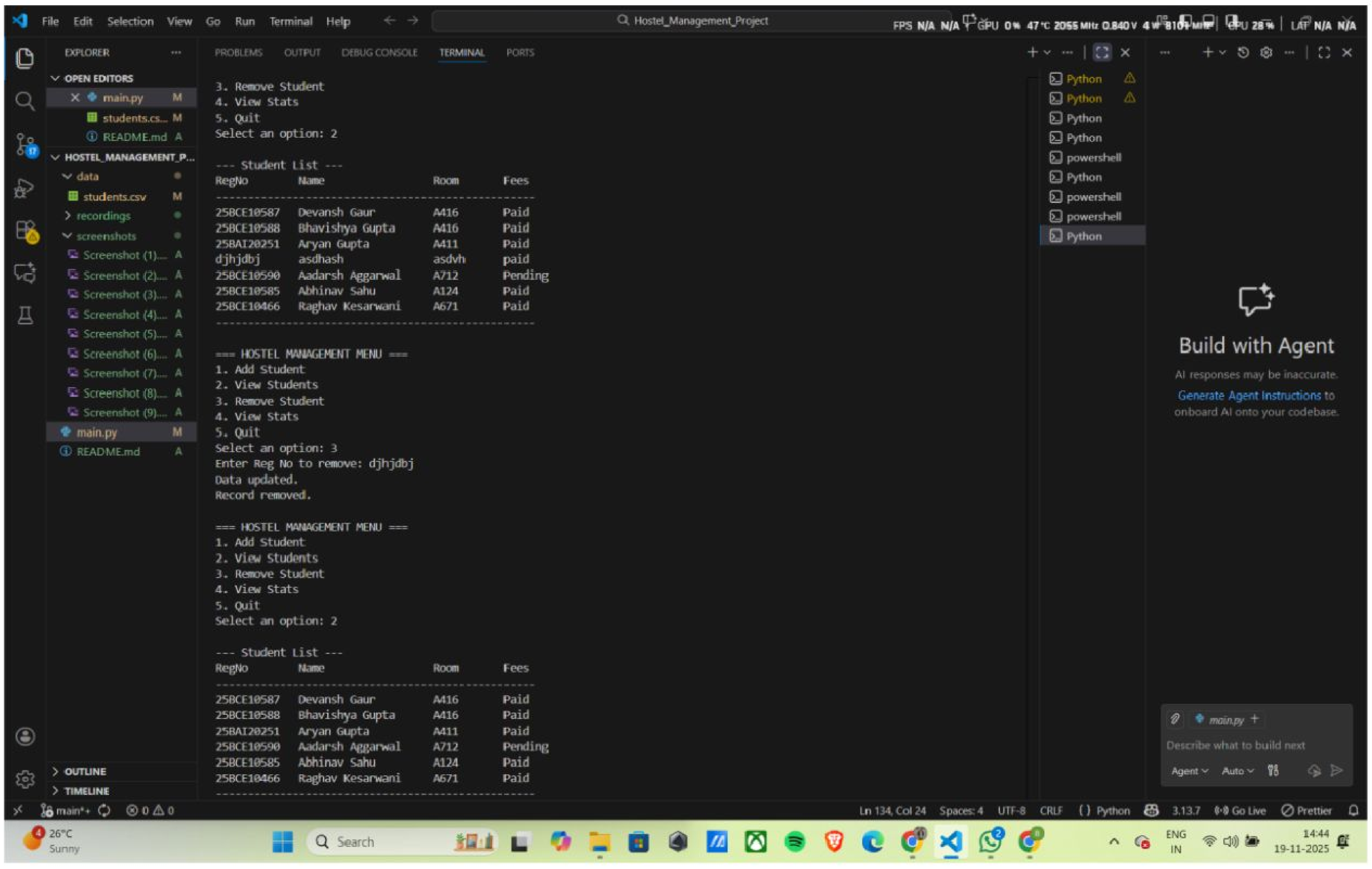
```python
class HostelManager:
    # Show everything
    def view_all(self):
        print("\n--- Student List ---")
        if not self.students:
            print("No records found.")
            return

        print(f"{'RegNo':<12} {'Name':<20} {'Room':<10} {'Fees'}")
        print("-" * 50)
        for s in self.students:
            print(f"{s.reg:<12} {s.name:<20} {s.room:<10} {s.fee_stat}")
        print("-" * 50)

    # Delete/checkout student
    def remove_student(self):
        reg = input("Enter Reg No to remove: ")
        for s in self.students:
            if s.reg == reg:
                self.students.remove(s)
                self._save()
                print("Record removed.")
                return

        print("No student with that Reg No found.")

    # Basic stats
    def stats(self):
        total = len(self.students)
        paid = sum(1 for s in self.students if s.fee_stat.lower() == "paid")

        print("\n--- Stats ---")
        print("Total Students:", total)
        print("Paid Fees:", paid)
        print("Pending Fees:", total - paid)


def main():
    hm = HostelManager()

    while True:
        print("\n=== HOSTEL MANAGEMENT MENU ===")
```


Hostel_Management_Project
main.py M
students.csv M
README.md A
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Describe what to build next
Ln 144, Col 11 Spaces: 4 UTF-8 CRLF Python 3.13.7

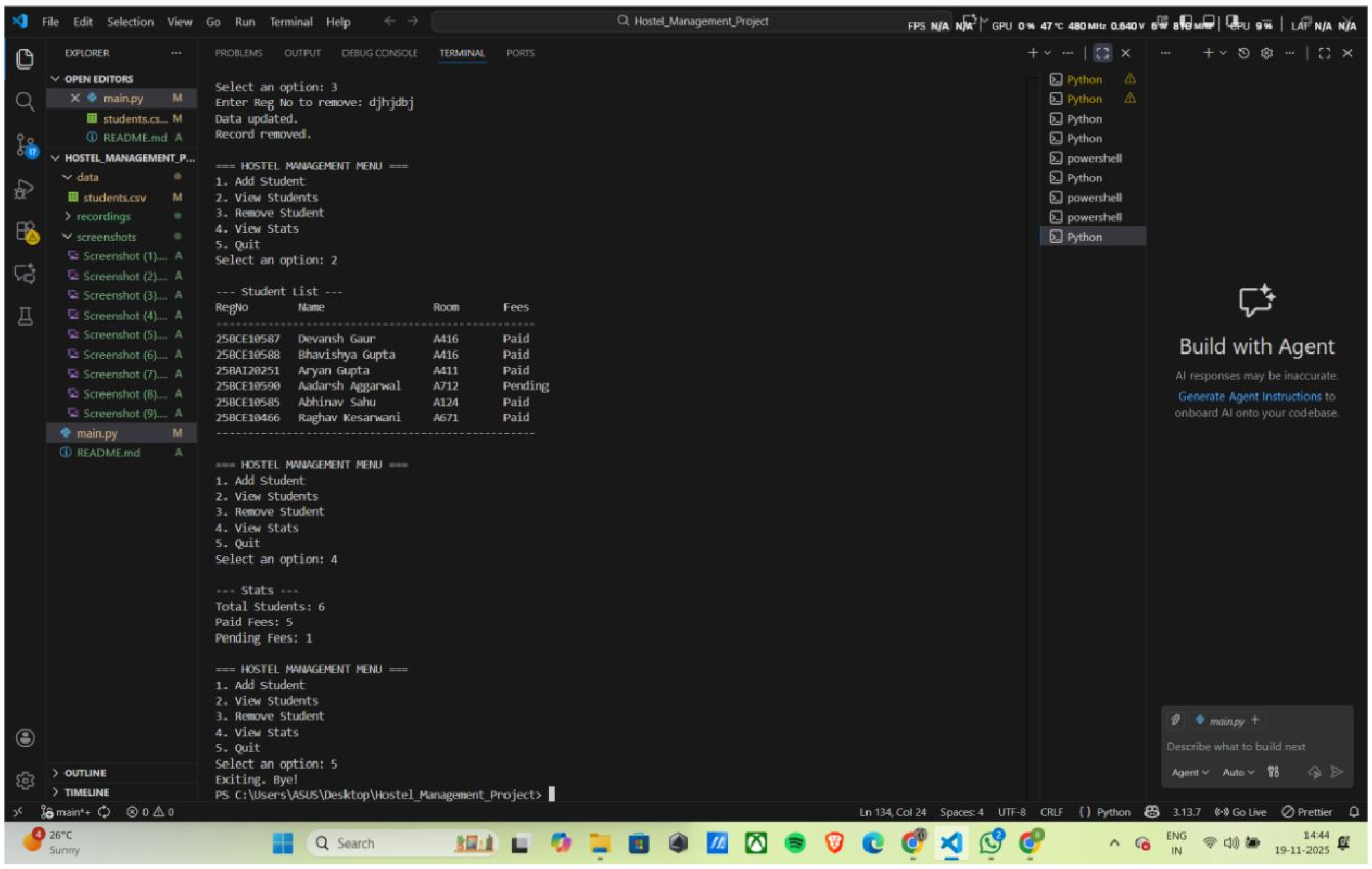
```python
def main():
    while True:
        print("\n=== HOSTEL MANAGEMENT MENU ===")
        print("1. Add Student")
        print("2. View Students")
        print("3. Remove Student")
        print("4. View Stats")
        print("5. Quit")

        ch = input("Select an option: ")

        if ch == "1":
            hm.add_student()
        elif ch == "2":
            hm.view_all()
        elif ch == "3":
            hm.remove_student()
        elif ch == "4":
            hm.stats()
        elif ch == "5":
            print("Exiting. Bye!")
            break
        else:
            print("Invalid choice, try again.")


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



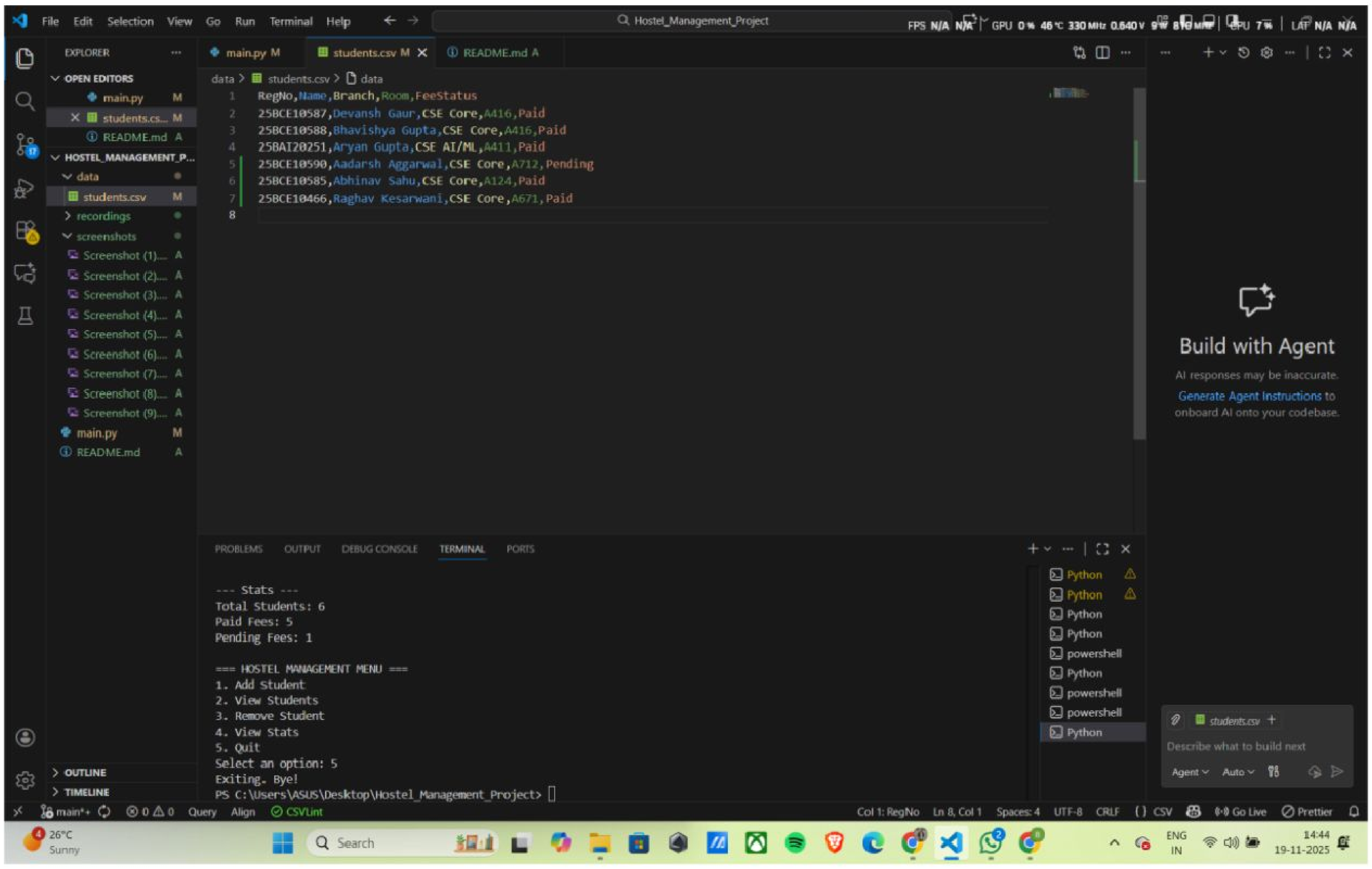
```
PS C:\Users\ASUS\Desktop\Hostel_Management_Project> & C:/Python313/python.exe c:/Users/ASUS/Desktop/Hostel_Management_Project/
main.py

=== HOSTEL MANAGEMENT MENU ===
1. Add Student
2. View Students
3. Remove Student
4. View Stats
5. Quit
Select an option: 2

--- Student List ---
RegNo        Name                Room      Fees
-------------------------------------------------
25BCE10587   Devansh Gaur        A416      Paid
25BCE10588   Bhavishya Gupta     A416      Paid
25BAI20251   Aryan Gupta         A411      Paid
djhjdbj      asdhash             asdvh     paid
25BCE10590   Aadarsh Aggarwal    A712      Pending
25BCE10585   Abhinav Sahu        A124      Paid
-------------------------------------------------

=== HOSTEL MANAGEMENT MENU ===
1. Add Student
2. View Students
3. Remove Student
4. View Stats
5. Quit
Select an option: 1

--- Add Student ---
Registration No: 25BCE10466
Name: Raghav Kesarwani
Branch: CSE Core
Room No: A671
Fee Status (Paid/Pending): Paid
Data updated.
Raghav Kesarwani added to room A671.

=== HOSTEL MANAGEMENT MENU ===
1. Add Student
2. View Students
3. Remove Student
4. View Stats
5. Quit
Select an option: 2
```



```
3. Remove Student
4. View Stats
5. Quit
Select an option: 2

--- Student List ---
RegNo        Name                Room      Fees
-------------------------------------------------
25BCE10587   Devansh Gaur        A416      Paid
25BCE10588   Bhavishya Gupta     A416      Paid
25BAI20251   Aryan Gupta         A411      Paid
djhjdbj      asdhash             asdvh     paid
25BCE10590   Aadarsh Aggarwal    A712      Pending
25BCE10585   Abhinav Sahu        A124      Paid
25BCE10466   Raghav Kesarwani    A671      Paid
-------------------------------------------------

=== HOSTEL MANAGEMENT MENU ===
1. Add Student
2. View Students
3. Remove Student
4. View Stats
5. Quit
Select an option: 3
Enter Reg No to remove: djhjdbj
Data updated.
Record removed.

=== HOSTEL MANAGEMENT MENU ===
1. Add Student
2. View Students
3. Remove Student
4. View Stats
5. Quit
Select an option: 2

--- Student List ---
RegNo        Name                Room      Fees
-------------------------------------------------
25BCE10587   Devansh Gaur        A416      Paid
25BCE10588   Bhavishya Gupta     A416      Paid
25BAI20251   Aryan Gupta         A411      Paid
25BCE10590   Aadarsh Aggarwal    A712      Pending
25BCE10585   Abhinav Sahu        A124      Paid
25BCE10466   Raghav Kesarwani    A671      Paid
```

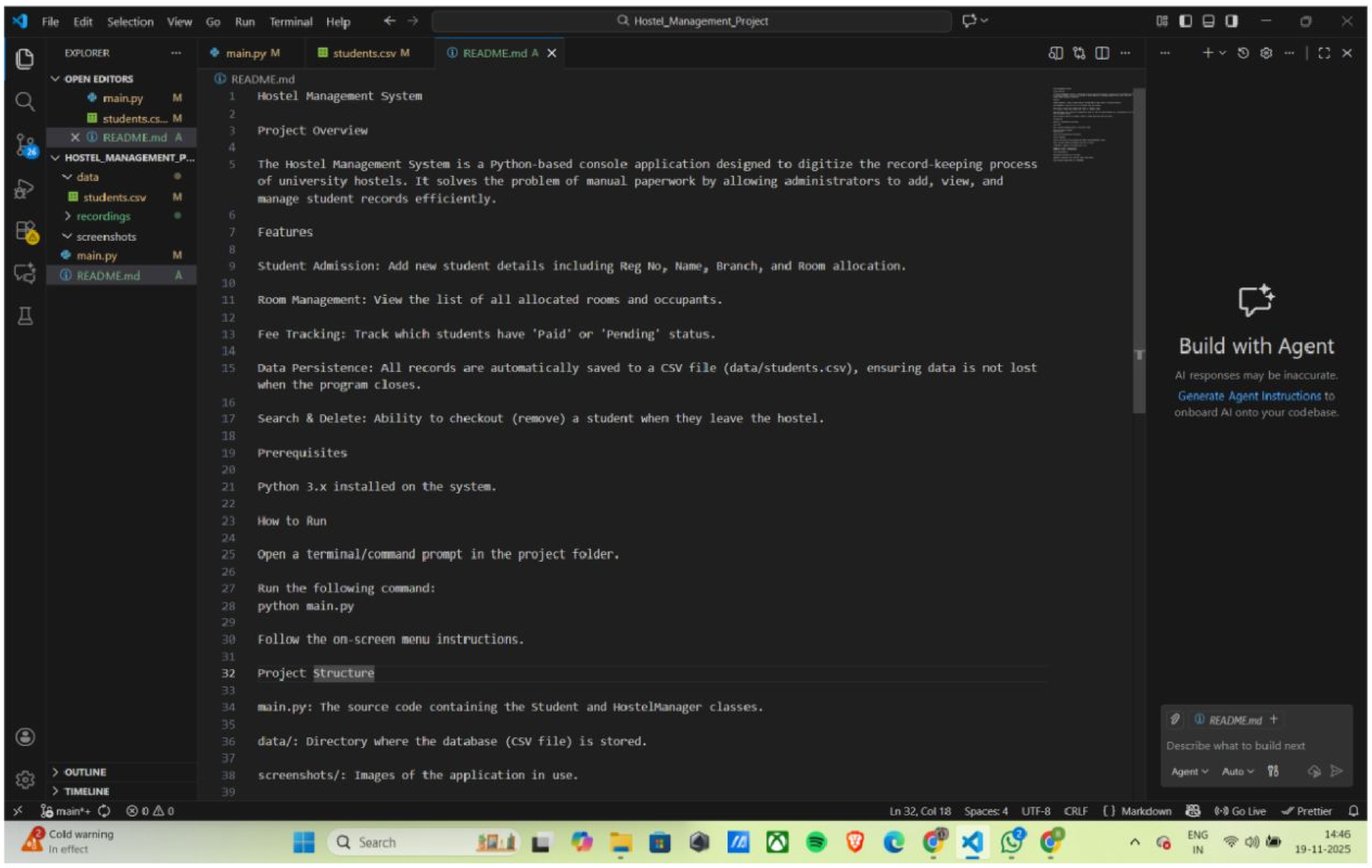


```
Select an option: 3
Enter Reg No to remove: djhjdbj
Data updated.
Record removed.

=== HOSTEL MANAGEMENT MENU ===
1. Add Student
2. View Students
3. Remove Student
4. View Stats
5. Quit
Select an option: 2

--- Student List ---
RegNo       Name               Room      Fees
--------------------------------------------------
25BCE10587  Devansh Gaur       A416      Paid
25BCE10588  Bhavishya Gupta    A416      Paid
25BAI20251  Aryan Gupta        A411      Paid
25BCE10590  Aadarsh Aggarwal   A712      Pending
25BCE10585  Abhinav Sahu       A124      Paid
25BCE10466  Raghav Kesarwani   A671      Paid
--------------------------------------------------

=== HOSTEL MANAGEMENT MENU ===
1. Add Student
2. View Students
3. Remove Student
4. View Stats
5. Quit
Select an option: 4

--- Stats ---
Total Students: 6
Paid Fees: 5
Pending Fees: 1

=== HOSTEL MANAGEMENT MENU ===
1. Add Student
2. View Students
3. Remove Student
4. View Stats
5. Quit
Select an option: 5
Exiting. Bye!
PS C:\Users\ASUS\Desktop\Hostel_Management_Project>
```

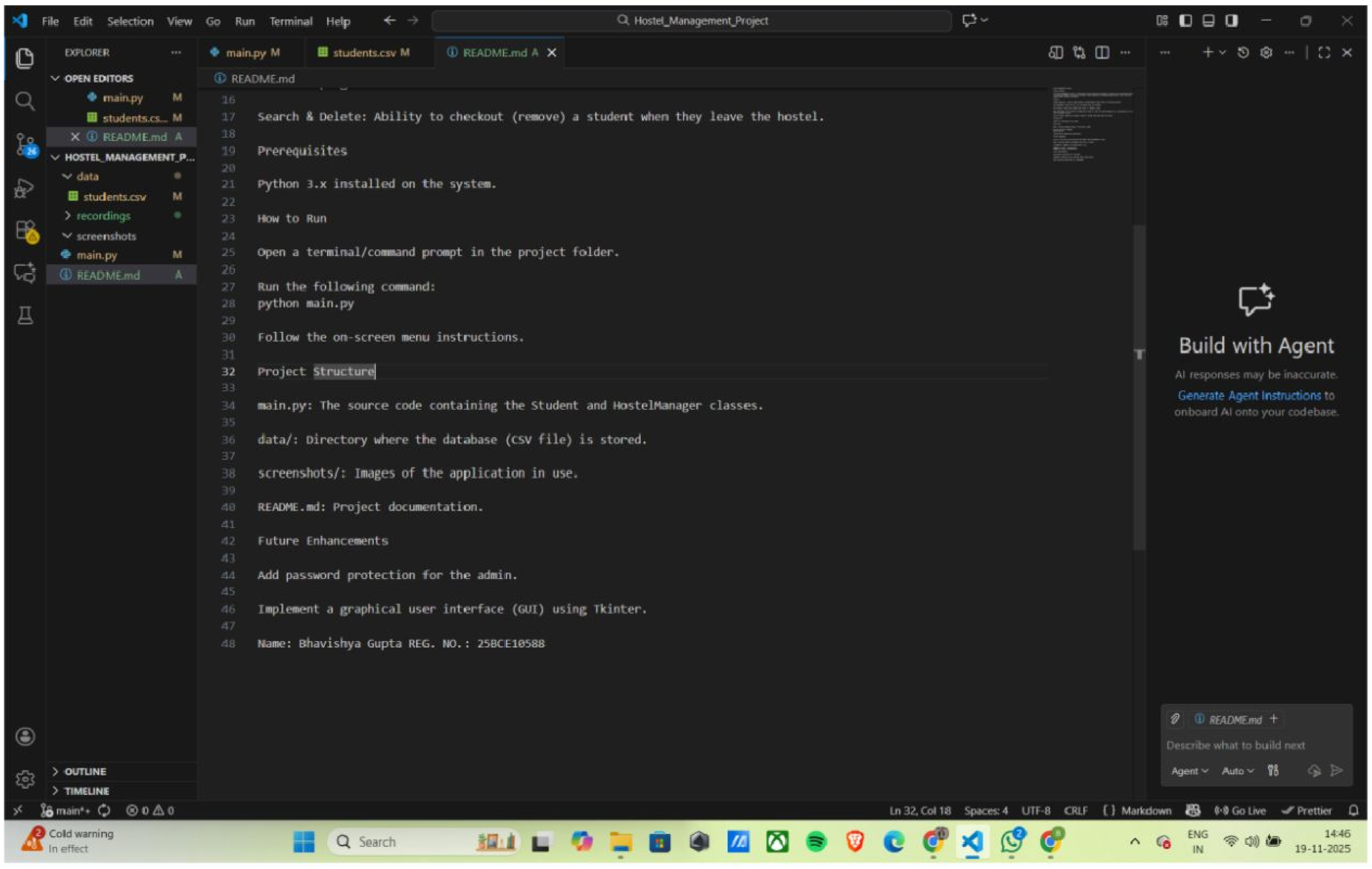


```
RegNo,Name,Branch,Room,FeeStatus
25BCE10587,Devansh Gaur,CSE Core,A416,Paid
25BCE10588,Bhavishya Gupta,CSE Core,A416,Paid
25BAI20251,Aryan Gupta,CSE AI/ML,A411,Paid
25BCE10590,Aadarsh Aggarwal,CSE Core,A712,Pending
25BCE10585,Abhinav Sahu,CSE Core,A124,Paid
25BCE10466,Raghav Kesarwani,CSE Core,A671,Paid
```

```
--- Stats ---
Total Students: 6
Paid Fees: 5
Pending Fees: 1

=== HOSTEL MANAGEMENT MENU ===
1. Add Student
2. View Students
3. Remove Student
4. View Stats
5. Quit
Select an option: 5
Exiting. Bye!
PS C:\Users\ASUS\Desktop\Hostel_Management_Project>
```



```
Hostel Management System

Project Overview

The Hostel Management System is a Python-based console application designed to digitize the record-keeping process
of university hostels. It solves the problem of manual paperwork by allowing administrators to add, view, and
manage student records efficiently.

Features

Student Admission: Add new student details including Reg No, Name, Branch, and Room allocation.

Room Management: View the list of all allocated rooms and occupants.

Fee Tracking: Track which students have 'Paid' or 'Pending' status.

Data Persistence: All records are automatically saved to a CSV file (data/students.csv), ensuring data is not lost
when the program closes.

Search & Delete: Ability to checkout (remove) a student when they leave the hostel.

Prerequisites

Python 3.x installed on the system.

How to Run

Open a terminal/command prompt in the project folder.

Run the following command:
python main.py

Follow the on-screen menu instructions.

Project Structure

main.py: The source code containing the Student and HostelManager classes.

data/: Directory where the database (CSV file) is stored.

screenshots/: Images of the application in use.
```



```
Search & Delete: Ability to checkout (remove) a student when they leave the hostel.

Prerequisites

Python 3.x installed on the system.

How to Run

Open a terminal/command prompt in the project folder.

Run the following command:
python main.py

Follow the on-screen menu instructions.

Project Structure

main.py: The source code containing the Student and HostelManager classes.

data/: Directory where the database (CSV file) is stored.

screenshots/: Images of the application in use.

README.md: Project documentation.

Future Enhancements

Add password protection for the admin.

Implement a graphical user interface (GUI) using Tkinter.

Name: Bhavishya Gupta REG. NO.: 25BCE10588
```

11. Testing Approach

- Manual testing by running the program and interacting with each menu option.
- Tested for duplicate student entries to ensure the system prevents duplicates.
- Tested removal of existing and non-existing records to check error handling.
- Verified data persistence by restarting the program and checking data integrity.

12. Challenges Faced

- Handling file read/write exceptions and ensuring data consistency.
- Managing user input validation in a command-line environment.
- Structuring the code to maintain modularity while keeping it simple.

13. Learnings & Key Takeaways

- Gained practical experience in file handling with Python.
- Understood the importance of modular code design for maintainability.
- Improved skills in CLI-based user interaction design.
- Learned about data validation and error handling in simple systems.

14. Future Enhancements

- Develop a graphical user interface (GUI) for better user experience.
- Integrate a database system (e.g., SQLite) for more scalable data management.
- Add features like fee reminders and automated reports.
- Implement user authentication for secure access.

15. References

- Python Official Documentation: https://docs.python.org/3/
- CSV Module Documentation: https://docs.python.org/3/library/csv.html
- Various online tutorials on Python file handling and CLI applications.