

AI ASSIGNMENT-6.5

Name : M.Raja

Hall No. : 2303A52277

Batch : 36

Task Description #1 (AI-Based Code Completion for Conditional

Eligibility Check)

Task: Use an AI tool to generate eligibility logic.

Prompt:

“Generate Python code to check voting eligibility based on age and citizenship.”

Expected Output:

- AI-generated conditional logic.
- Correct eligibility decisions.
- Explanation of conditions.

The screenshot shows the VS Code interface with the Explorer sidebar open, displaying files like AI, .vscode, Al.py, ass-4.1, tpy, and task 3.py. The Al.py file is selected and shown in the main editor area. The code generated by AI is as follows:

```
##Generate a Python program that checks voting eligibility using conditional statements
def check_voting_eligibility(age):
    if age < 0:
        return "Invalid age. Age cannot be negative."
    elif age < 18:
        return "You are not eligible to vote."
    else:
        return "You are eligible to vote."
# Example usage
age = int(input("Enter your age: "))
result = check_voting_eligibility(age)
print(result)
```

To the right of the editor, there is a sidebar with the heading "Build with Agent". It includes a note that "AI responses may be inaccurate.", a "Generate Agent" button, and instructions to "Instructions to onboard AI onto your codebase".

Output:

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows a folder named "AI" containing ".vscode", "AI.py", "ass-4.1", "tpy", and "task 3.py".
- Code Editor:** Displays the file "AI.py" with the following code:

```
1  ##Generate a Python program that checks voting eligibility using conditional statements
2  def check_voting_eligibility(age):
3      if age < 0:
4          return "Invalid age. Age cannot be negative."
5      elif age < 18:
```
- Terminal:** Shows the output of running the code:

```
You are eligible to vote.
PS C:\Users\reddy\OneDrive\Desktop\AI> & c:\Users\reddy\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/reddy/OneDrive/Desktop/AI/AI.py
Enter your age: 19
You are eligible to vote.
PS C:\Users\reddy\OneDrive\Desktop\AI>
```
- AI Assistant Panel:** On the right, there's a panel titled "Build with Agent" with instructions to onboard AI onto the codebase.

Task Description #2(AI-Based Code Completion for Loop-Based

String Processing)

Task: Use an AI tool to process strings using loops.

Prompt:

"Generate Python code to count vowels and consonants in a string using a loop."

Expected Output:

- AI-generated string processing logic.
- Correct counts.
- Output verification.

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows a folder named "AI" containing ".vscode", "AI.py", "ass-4.1", "tpy", and "task 3.py".
- Code Editor:** Displays the file "AI.py" with the following code:

```
13  ##Generate Python code to count the number of vowels and consonants in a given string using
14  ##The program should take a string as input from the user and display the total count of vo
15  def count_vowels_and_consonants(input_string):
16      vowels = "aeiouAEIOU"
17      vowel_count = 0
18      consonant_count = 0
19
20      for char in input_string:
21          if char.isalpha(): # Check if the character is a letter
22              if char in vowels:
23                  vowel_count += 1
24              else:
25                  consonant_count += 1
26
27      return vowel_count, consonant_count
28  # Example usage
29  input_string = input("Enter a string: ")
30  vowels, consonants = count_vowels_and_consonants(input_string)
31  print(f"Number of vowels: {vowels}")
32  print(f"Number of consonants: {consonants}")
33
```
- Terminal:** Shows the output of running the code:

```
Number of consonants: 5
PS C:\Users\reddy\OneDrive\Desktop\AI>
```
- AI Assistant Panel:** On the right, there's a panel titled "Build with Agent" with instructions to onboard AI onto the codebase.

Output:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows files: .vscode, AI.py, ass-4.1, t.py, and task 3.py.
- Code Editor (Top Center):** Displays Python code for counting vowels and consonants. The code uses AI-generated documentation strings and imports.
- Terminal (Bottom):** Shows the output of running the script with user input "Ashwitha".
- Right Sidebar:** Features a "Build with Agent" section with buttons for "powershell" and "AI responses may be inaccurate". It also includes "Generate Agent", "Instructions to onboard", and "AI onto your codebase".
- Bottom Status Bar:** Shows file path (Ln 33, Col 1), code editor settings (Spaces: 4, CRLF, Python), and default interpreter path (3.13.9.final0).

Task Description #3 (AI-Assisted Code Completion Reflection)

Task)

Task: Use an AI tool to generate a complete program using classes, loops, and conditionals.

Prompt:

“Generate a Python program for a library management system using classes, loops, and conditional statements.”

Expected Output:

- Complete AI-generated program.
 - Review of AI suggestions quality.
 - Short reflection on AI-assisted coding experience.

The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left lists files: AI, .vscode, Al.py, ass-4.1, tpy, and task 3.py. The Al.py file is open in the main editor area. The code defines a Book class with methods for initialization, string representation, adding books to a library, displaying books, and checking out books. A status message is printed for each book's availability.

```
#Generate a complete Python program for a simple Library Management System using classes, Al.py
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.is_available = True

    def __str__(self):
        status = "Available" if self.is_available else "Checked Out"
        return f"{self.title} by {self.author} - {status}"

class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book):
        self.books.append(book)
        print(f"Book '{book.title}' added to the library.")

    def display_books(self):
        if not self.books:
            print("No books in the library.")
            return
        print("Books in the library:")
        for book in self.books:
            print(book)

    def check_out_book(self, title):
        for book in self.books:
            if book.title == title:
                if book.is_available:
                    book.is_available = False
                    print(f"You have checked out '{book.title}'.")
                    return
                else:
                    print(f"Sorry, '{book.title}' is currently checked out.")
                    return
        print(f"Book '{title}' not found in the library.")

def return_book(self, title):
    for book in self.books:
        if book.title == title:
            if not book.is_available:
                book.is_available = True
                print(f"You have returned '{book.title}'.")
                return
            else:
                print(f"'{book.title}' was not checked out.")
                return
    print(f"Book '{title}' not found in the library.")

def main():
    library = Library()
    while True:
        print("\nLibrary Management System")
        print("1. Add Book")
        print("2. Display Books")
        print("3. Check Out Book")

        choice = input("Enter your choice (1/2/3): ")
        if choice == "1":
            title = input("Enter book title: ")
            author = input("Enter book author: ")
            book = Book(title, author)
            library.add_book(book)
```

The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left lists files: AI, .vscode, Al.py, ass-4.1, tpy, and task 3.py. The Al.py file is open in the main editor area. The code now includes a main function that initializes a Library object and enters a loop. Inside the loop, it prints the library management system menu and prompts the user for a choice. The logic for adding books, displaying books, and checking out books remains the same as in the previous screenshot.

```
def main():
    library = Library()
    while True:
        print("\nLibrary Management System")
        print("1. Add Book")
        print("2. Display Books")
        print("3. Check Out Book")

        choice = input("Enter your choice (1/2/3): ")
        if choice == "1":
            title = input("Enter book title: ")
            author = input("Enter book author: ")
            book = Book(title, author)
            library.add_book(book)
```

The screenshot shows two instances of the Visual Studio Code (VS Code) interface, both displaying the same Python code for a Library Management System. The code is named `Al.py` and is located in a folder structure that includes `.vscode`, `ass-4.1`, `tpy`, and `task 3.py`.

```
81     |     print(f"Book '{title}' not found in the library.")
82 def main():
83     library = Library()
84     while True:
85         print("\nLibrary Management System")
86         print("1. Add Book")
87         print("2. Display Books")
88         print("3. Check Out Book")
89         print("4. Return Book")
90         print("5. Exit")
91         choice = input("Enter your choice (1-5): ")
92
93         if choice == '1':
94             title = input("Enter book title: ")
95             author = input("Enter book author: ")
96             book = Book(title, author)
97             library.add_book(book)
98         elif choice == '2':
99             library.display_books()
100        elif choice == '3':
101            title = input("Enter the title of the book to check out: ")
102            library.check_out_book(title)
103        elif choice == '4':
104            title = input("Enter the title of the book to return: ")
105            library.return_book(title)
106        elif choice == '5':
107            print("Exiting the Library Management System. Goodbye!")
108            break
109        else:
110            print("Invalid choice. Please try again.")
111 if __name__ == "__main__":
112     main()
```

The right-hand pane of each VS Code window displays an AI assistant interface titled "Build with Agent". It includes a message box, a status bar indicating "AI responses may be inaccurate", and a "Generate Agent" button. The bottom status bar of both windows shows the date and time as "23-01-2026" and "11:38".

Output:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the AI folder, including AI.py, ass-4.1, tpy, and task 3.py.
- Code Editor:** Displays the AI.py script. The code includes a main() function that handles user input for a library management system. The AI tool is suggesting completions for variables like book title and author.
- Terminal:** Shows the execution of the script and its output. The user adds a book titled "Python Basics" by "john".
- AI Panel:** On the right, there's an "AI" panel with a "Build with Agent" section. It includes a note about AI responses being inaccurate, a "Generate Agent" button, and instructions to onboard AI onto the codebase.
- Bottom Status Bar:** Provides information like the current file (AI.py), line and column (Ln 113, Col 5), encoding (UTF-8), and date (23-01-2026).

Task Description #4 (AI-Assisted Code Completion for Class-

Based Attendance System)

Task: Use an AI tool to generate an attendance management class.

Prompt: “Generate a Python class to mark and display student attendance using loops.”

Expected Output:

- AI-generated attendance logic.
- Correct display of attendance.

The image displays two vertically stacked screenshots of the Visual Studio Code (VS Code) interface, showing the same Python code in two separate files: `AI.py` and `ass-4.1`.

Top Editor (AI.py):

```
113     ##Generate a Python class to mark and display student attendance using loops.
114     ##The program should allow marking students as present or absent and display the attendance
115     class Student:
116         def __init__(self, name):
117             self.name = name
118             self.attendance = []
119
120         def mark_present(self):
121             self.attendance.append("Present")
122
123         def mark_absent(self):
124             self.attendance.append("Absent")
125
126         def display_attendance(self):
127             print(f"Attendance for {self.name}:")
128             for i, status in enumerate(self.attendance, start=1):
129                 print(f"Day {i}: {status}")
130
131
132     class AttendanceSystem:
133         def __init__(self):
134             self.students = []
135
136         def add_student(self, student):
137             self.students.append(student)
138
139         def display_all_attendance(self):
140             for student in self.students:
141                 student.display_attendance()
```

Bottom Editor (ass-4.1):

```
143
144
145     def main():
146         attendance_system = AttendanceSystem()
147
148         # Input validation
149         num_students = int(input("Enter the number of students: "))
150
151         for _ in range(num_students):
152             name = input("Enter student name: ")
153             student = Student(name)
154             attendance_system.add_student(student)
155
156         num_days = int(input("Enter the number of days to mark attendance: "))
157
158         for day in range(1, num_days + 1):
159             print(f"\nMarking attendance for Day {day}")
160             for student in attendance_system.students:
161                 while True:
162                     status = input(f"Is {student.name} present? (y/n): ").strip().lower()
163                     if status == 'y':
164                         student.mark_present()
165                         break
166                     elif status == 'n':
167                         student.mark_absent()
168                         break
169                     else:
170                         print("Invalid input! Please enter 'y' or 'n'.")
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the AI folder: AI.py, ass-4.1, tpy, and task 3.py.
- Code Editor:** The AI.py file is open, displaying Python code for an attendance system. The code includes functions for marking present and absent students, and a main loop for user input.
- AI Panel:** A sidebar on the right is titled "Build with Agent". It includes a message stating "AI responses may be inaccurate.", a "Generate Agent" button, and "Instructions to onboard AI onto your codebase".
- Bottom Status Bar:** Shows the current line (Ln 179, Col 1), spaces used (Spaces: 4), encoding (UTF-8), and line endings (CRLF). It also displays the Python interpreter path (defaultInterpreterPath: 3.13.9.final.0), Go Live buttons, and system status (ENG IN, 11:51, 23-01-2026).

Output:

The screenshot shows two instances of the Visual Studio Code (VS Code) interface. Both instances have the same workspace open, featuring an Explorer sidebar on the left containing files like 'AI', '.vscode', 'Al.py', 'ass-4.1', 'tpy', and 'task 3.py'. The main editor area displays a Python script named 'Al.py' with the following code and output:

```
145 def main():
    PS C:\Users\reddy\OneDrive\Desktop\AI> & c:\Users\reddy\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users\reddy\OneDrive\Desktop/AI/AI.py
    Enter the number of students: 4
    Enter student name: Ashwutha
    Enter student name: Harshini
    Enter student name: Akshitha
    Enter student name: varshitha
    Enter the number of days to mark attendance: 4

    Marking attendance for Day 1
    Is Ashwutha present? (y/n): y
    Is Harshini present? (y/n): y
    Is Akshitha present? (y/n): y
    Is varshitha present? (y/n): y

    Marking attendance for Day 2
    Is Ashwutha present? (y/n): n
    Is Harshini present? (y/n): y
    Is Akshitha present? (y/n): y
    Is varshitha present? (y/n): y

    Marking attendance for Day 3
    Is Ashwutha present? (y/n): y
    Is Harshini present? (y/n): n
    Is Akshitha present? (y/n): n
    Is varshitha present? (y/n): y

    Marking attendance for Day 4
    Is Ashwutha present? (y/n): y
    Is Harshini present? (y/n): y
    Is Akshitha present? (y/n): y
    Is varshitha present? (y/n): y
```

The status bar at the bottom of both windows indicates the file is Python, the interpreter path is '3.13.9.final.0', and the date and time are '23-01-2026 11:51'. A floating 'Build with Agent' panel is visible on the right side of both windows, providing options to 'Generate Agent' and 'Instructions to onboard AI onto your codebase'. The interface is dark-themed.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files like `AI.py`, `ass-4.1`, `t.py`, and `task 3.py`.
- Code Editor:** Displays a Python script (`AI.py`) with AI-generated code completion. The code prints attendance records for two students: Akshitha and varshitha over four days.
- Terminal:** Shows the command line prompt `PS C:\Users\reddy\OneDrive\Desktop\AI>`.
- Right Panel:** Features an "AI" sidebar with a "Build with Agent" section, a note about AI responses being inaccurate, and a "Generate Agent" button. It also includes a text input field for instructions and a "Describe what to build next" button.
- Bottom Bar:** Includes standard OS icons for search, file operations, and system status.

Task Description #5 (AI-Based Code Completion for Conditional Menu Navigation)

Task: Use an AI tool to complete a navigation menu.

Prompt: “Generate a Python program using loops and conditionals

to simulate an ATM menu.”

Expected Output:

- AI-generated menu logic.
- Correct option handling.
- Output verification.

The screenshot shows two instances of the Visual Studio Code (VS Code) interface, demonstrating the integration of AI features. Both instances have a dark theme.

Top Instance:

- Explorer View:** Shows files in the workspace, including `.vscode`, `AI`, `ass-4.1`, `Al.py`, `tpy`, and `task 3.py`.
- Code Editor:** Displays the `Al.py` file containing Python code for an ATM menu system. The code includes functions for withdraw, exit, deposit, and withdraw. A cursor is positioned at the end of the `atm_menu` function definition.
- Right Panel:** Features an "AI" icon with a speech bubble, a "Build with Agent" button, and a message stating "AI responses may be inaccurate." Below it are buttons for "Generate Agent" and "Instructions to onboard AI onto your codebase".
- Bottom Status Bar:** Shows the current line (Ln 222, Col 5), spaces (Spaces: 4), encoding (UTF-8), and interpreter path (defaultInterpretorPath: 3.13.9.final.0). It also includes "Go Live" and "ENG IN" status indicators.

Bottom Instance:

- Explorer View:** Similar to the top instance, showing the same workspace files.
- Code Editor:** Displays the `Al.py` file with the same code as the top instance. The cursor is now at the beginning of the `if __name__ == '__main__':` block.
- Bottom Status Bar:** Shows the current line (Ln 222, Col 5), spaces (Spaces: 4), encoding (UTF-8), and interpreter path (defaultInterpretorPath: 3.13.9.final.0). It also includes "Go Live" and "ENG IN" status indicators.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with an AI integration. The main area displays Python code for an ATM menu system. The sidebar includes an 'EXPLORER' view showing files like 'AI.py', '.vscode', 'ass-4.1', 't.py', and 'task 3.py'. A central panel shows the code for 'AI.py' with line numbers 187 to 222. The right side features an AI interface with a message input field, a 'Build with Agent' button, and instructions about AI responses being inaccurate. The bottom status bar shows file details like 'Ln 222, Col 5' and 'defaultInterpretePath: 3.13.9.final.0', along with system icons and a date/time stamp '23-01-2026 11:55'.

```
187     def atm_menu():
188         balance = 1000
189         print("Welcome to the ATM!")
190         while True:
191             print("Please select an option:")
192             print("1. Deposit")
193             print("2. Withdraw")
194             print("3. Check Balance")
195             print("4. Exit")
196             choice = input("Enter your choice (1-4): ")
197             if choice == '1':
198                 amount = float(input("Enter amount to deposit: "))
199                 balance += amount
200                 print(f"${amount:.2f} deposited successfully.")
201             elif choice == '2':
202                 amount = float(input("Enter amount to withdraw: "))
203                 if 0 < amount <= balance:
204                     balance -= amount
205                     print(f"${amount:.2f} withdrawn successfully.")
206                 else:
207                     print("Invalid amount. Please enter a positive value within your balance.")
208             elif choice == '3':
209                 print(f"Your current balance is ${balance:.2f}.")
210             elif choice == '4':
211                 print("Thank you for using the ATM. Goodbye!")
212                 break
213             else:
214                 print("Invalid choice. Please select a valid option.")
215         if __name__ == "__main__":
216             atm_menu()
```