SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment # 6

Name of Student : Ch.Wishram
Enrollment No. : 2303A51532
Batch No. : 22

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."

Prompt:-
write a python program to check whether a person is eligible for voting or not based on age and citizenship. if we given any letter gives as invalid input and do same for citizenship

Code:-

Assignment6.py > ATM

```python
#Task-1
# write a python program to check whether a person is eligible f
# if we given any letter gives as invalid input and do same for
def check_voting_eligibility(age, citizenship):
    if not isinstance(age, int) or age < 0:
        return "Invalid input for age."
    if citizenship.lower() not in ['yes', 'no']:
        return "Invalid input for citizenship."

    if age >= 18 and citizenship.lower() == 'yes':
        return "Eligible to vote."
    else:
        return "Not eligible to vote."
# Example usage
age_input = input("Enter your age: ")
citizenship_input = input("Are you a citizen? (yes/no): ")
try:
    age = int(age_input)
    result = check_voting_eligibility(age, citizenship_input)
    print(result)
except ValueError:
    print("Invalid input for age.")
```

Output:-

```
PS C:\AIAC LAB\Assignments Codes> & C
Enter your age: 45
Are you a citizen? (yes/no): yes
Eligible to vote.
PS C:\AIAC LAB\Assignments Codes> & C
Enter your age: 17
Are you a citizen? (yes/no): yes
Not eligible to vote.
PS C:\AIAC LAB\Assignments Codes>
```

Justification:-

The user requested to complete the code in the middle given the prefix and suffix code snippets. The provided code snippet completes program as described, and the suffix contains additional tasks that are not part of program. The completion adheres to the user's request without altering or removing any existing code.

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."

Prompt:-

write a python program to count vowels and consonants in a given string using loops and conditional statements.

Code:

Assignment6.py > ...

```python
#Task-2
#write a python program to count vowels and consonants in
def count_vowels_consonants(input_string):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

    for char in input_string:
        if char.isalpha():
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1

    return vowel_count, consonant_count
# Example usage
input_str = input("Enter a string: ")
vowels, consonants = count_vowels_consonants(input_str)
print(f'Number of vowels: {vowels}')
print(f'Number of consonants: {consonants}')
```

Output:-

```
Enter a string: Hello I am Phani kumar and I'm currently Watching a series called Stranger Things
Number of vowels: 23
Number of consonants: 44
```

Justification:-
The function `count_vowels_consonants` correctly identifies vowels and consonants in a given string by iterating through each character and checking if it is an alphabetic character and whether it is a vowel or consonant.

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."

Prompt:-
write a python program to a library management system using classes and loops and conditional statements.

Code:

SR UNIVERSITY

-

Assignment6.py > ...

```python
#task-3
# write a python program to a library management system using classe
class Library:
    def __init__(self):
        self.books = []

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

    def remove_book(self, book_name):
        if book_name in self.books:
            self.books.remove(book_name)
            print(f'Book "{book_name}" removed from the library.')
        else:
            print(f'Book "{book_name}" not found in the library.')

    def display_books(self):
        if self.books:
            print("Books in the library:")
            for book in self.books:
                print(f'- {book}')
        else:
            print("No books in the library.")
# Example usage
library = Library()
while True:
    print("\nLibrary Management System")
    print("1. Add Book")
    print("2. Remove Book")
    print("3. Display Books")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")

    if choice == '1':
        book_name = input("Enter the name of the book to add: ")
        library.add_book(book_name)
    elif choice == '2':
        book_name = input("Enter the name of the book to remove: ")
        library.remove_book(book_name)
    elif choice == '3':
        library.display_books()
    elif choice == '4':
        print("Exiting the program.")
        break
    else:
        print("Invalid choice. Please try again.")
```

Output:-

SR UNIVERSITY

```
Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice (1-4): 1
Enter the name of the book to add: AIAC
Book "AIAC" added to the library.

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice (1-4): 1
Enter the name of the book to add: HPC
Book "HPC" added to the library.

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice (1-4): 1
Enter the name of the book to add: CP
Book "CP" added to the library.

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice (1-4): 1
Enter the name of the book to add: JAVA
Book "JAVA" added to the library.
```

```
Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice (1-4): 3
Books in the library:
- AIAC
- HPC
- CP
- JAVA

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice (1-4): 2
Enter the name of the book to remove: JAVA
Book "JAVA" removed from the library.

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice (1-4): 3
Books in the library:
- AIAC
- HPC
- CP

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice (1-4): 4
Exiting the program.
```

Justification:-
The above code implements a simple library management system using a class called `Library`. The class has methods to add, remove, and display books. A loop is used to provide a menu-driven interface for the user to interact with the library system. Conditional statements handle user choices and validate inputs.

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."

Prompt:-
write a python program for class based Attendance system to mark attendance of students and display the attendance percentage using loops. the program should allows to add multiple students and their attendance records class Attendance System.

Code:-

SR UNIVERSITY

```python
#task-4
# write a python program for class based Attendance system to mark a
# the program should aloows to add multiple students and their atter
class AttendanceSystem:
    def __init__(self):
        self.attendance_records = {}

    def add_student(self, student_name):
        if student_name not in self.attendance_records:
            self.attendance_records[student_name] = {'present': 0,
            print(f'Student "{student_name}" added to the attendance
        else:
            print(f'Student "{student_name}" already exists in the s

    def mark_attendance(self, student_name, present):
        if student_name in self.attendance_records:
            self.attendance_records[student_name]['total'] += 1
            if present:
                self.attendance_records[student_name]['present'] +=
            print(f'Attendance marked for "{student_name}".')
        else:
            print(f'Student "{student_name}" not found in the syster

    def display_attendance_percentage(self, student_name):
        if student_name in self.attendance_records:
            record = self.attendance_records[student_name]
            if record['total'] > 0:
                percentage = (record['present'] / record['total']) *
                print(f'Attendance percentage for "{student_name}":
            else:
                print(f'No attendance records for "{student_name}".'
        else:
            print(f'Student "{student_name}" not found in the syster
# Example usage
attendance_system = AttendanceSystem()
print("\nAttendance System")
while True:
    print("\n1. Add Student")
    print("2. Mark Attendance")
    print("3. Display Attendance Percentage")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")

    if choice == '1':
        student_name = input("Enter the name of the student to add:
        attendance_system.add_student(student_name)
    elif choice == '2':
        student_name = input("Enter the name of the student to mark
        present_input = input("Is the student present? (yes/no): ")
        present = present_input.lower() == 'yes'
        attendance_system.mark_attendance(student_name, present)
    elif choice == '3':
        student_name = input("Enter the name of the student to displ
        attendance_system.display_attendance_percentage(student_name
    elif choice == '4':
        print("Exiting the program.")
        break
    else:
        print("Invalid choice. Please try again.")
```

Output:-

SR UNIVERSITY

```
Attendance System

1. Add Student
2. Mark Attendance
3. Display Attendance Percentage
4. Exit
Enter your choice (1-4): 1
Enter the name of the student to add: sagar
Student "sagar" added to the attendance system.

1. Add Student
2. Mark Attendance
3. Display Attendance Percentage
4. Exit
Enter your choice (1-4): 1
Enter the name of the student to add: saiteja
Student "saiteja" added to the attendance system.

1. Add Student
2. Mark Attendance
3. Display Attendance Percentage
4. Exit
Enter your choice (1-4): 2
Enter the name of the student to mark attendance: sagar
Is the student present? (yes/no): yes
Attendance marked for "sagar".

1. Add Student
2. Mark Attendance
3. Display Attendance Percentage
4. Exit
Enter your choice (1-4): 2
Enter the name of the student to mark attendance: saiteja
Is the student present? (yes/no): no
Attendance marked for "saiteja".
```

SR UNIVERSITY

```
1. Add Student
2. Mark Attendance
3. Display Attendance Percentage
4. Exit
Enter your choice (1-4): 3
Enter the name of the student to display attendance percentage: sagar
Attendance percentage for "sagar": 100.00%

1. Add Student
2. Mark Attendance
3. Display Attendance Percentage
4. Exit
Enter your choice (1-4): 3
Enter the name of the student to display attendance percentage: saiteja
Attendance percentage for "saiteja": 0.00%

1. Add Student
2. Mark Attendance
3. Display Attendance Percentage
4. Exit
Enter your choice (1-4): 4
Exiting the program.
```

Justification:-
The code above implements an Attendance System using a class-based approach. It allows adding multiple students, marking their attendance, and displaying their attendance percentage. The program uses loops and conditional statements to interact with the user and manage the attendance records effectively.

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."

Prompt:-
write a python program to complete a navigation Sytem using loops and conditionals to simulate an ATM System.

Code:-

SR UNIVERSITY

```python
#Task-5
#write a python program to complete a navigation Sytem
class ATM:
    def __init__(self, balance=0):
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f'Deposited: ${amount:.2f}')
        else:
            print("Invalid deposit amount.")

    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f'Withdrew: ${amount:.2f}')
        else:
            print("Invalid withdrawal amount or insuffi

    def check_balance(self):
        print(f'Current balance: ${self.balance:.2f}')
# Example usage
atm = ATM()
print("\nATM System")
while True:
    print("\n1. Deposit")
    print("2. Withdraw")
    print("3. Check Balance")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")

    if choice == '1':
        amount = float(input("Enter amount to deposit:
        atm.deposit(amount)
    elif choice == '2':
        amount = float(input("Enter amount to withdraw:
        atm.withdraw(amount)
    elif choice == '3':
        atm.check_balance()
    elif choice == '4':
        print("Exiting the program.")
        break
    else:
        print("Invalid choice. Please try again.")
```

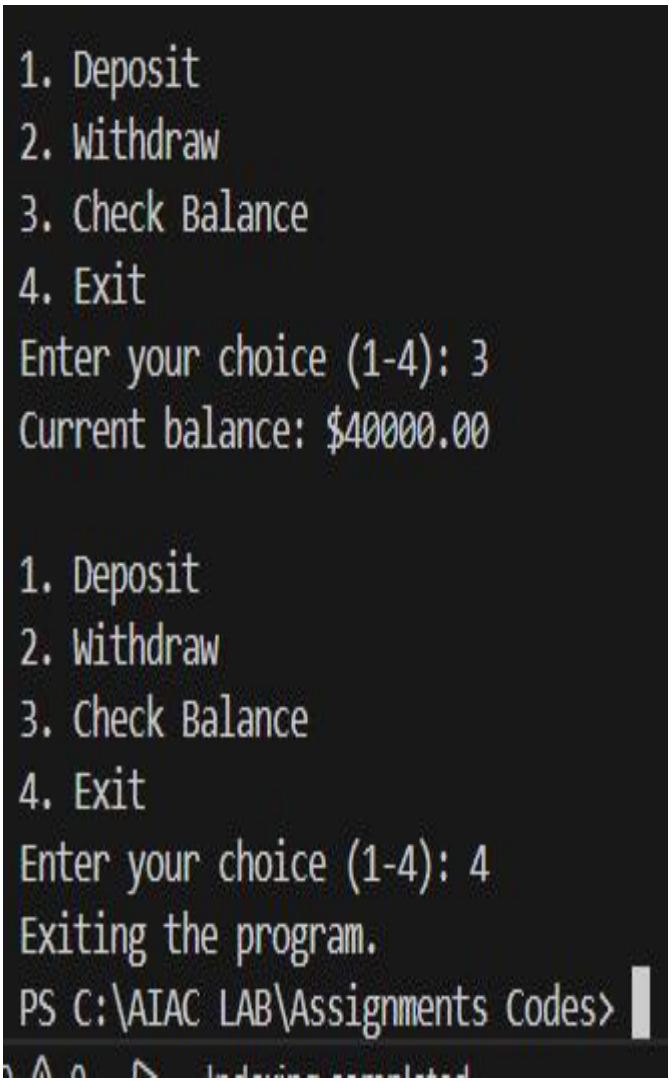
Output:-

```
ATM System

1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 1
Enter amount to deposit: 50000
Deposited: $50000.00

1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 3
Current balance: $50000.00

1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 2
Enter amount to withdraw: 10000
Withdrew: $10000.00
```



Justification:-
The user input is not one of the valid options (1-4), so the program informs them of the invalid choice and prompts them to try again. and continues the loop until a valid choice is made or the user decides to exit.