Experiment 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals

Name: Syed Murtaza

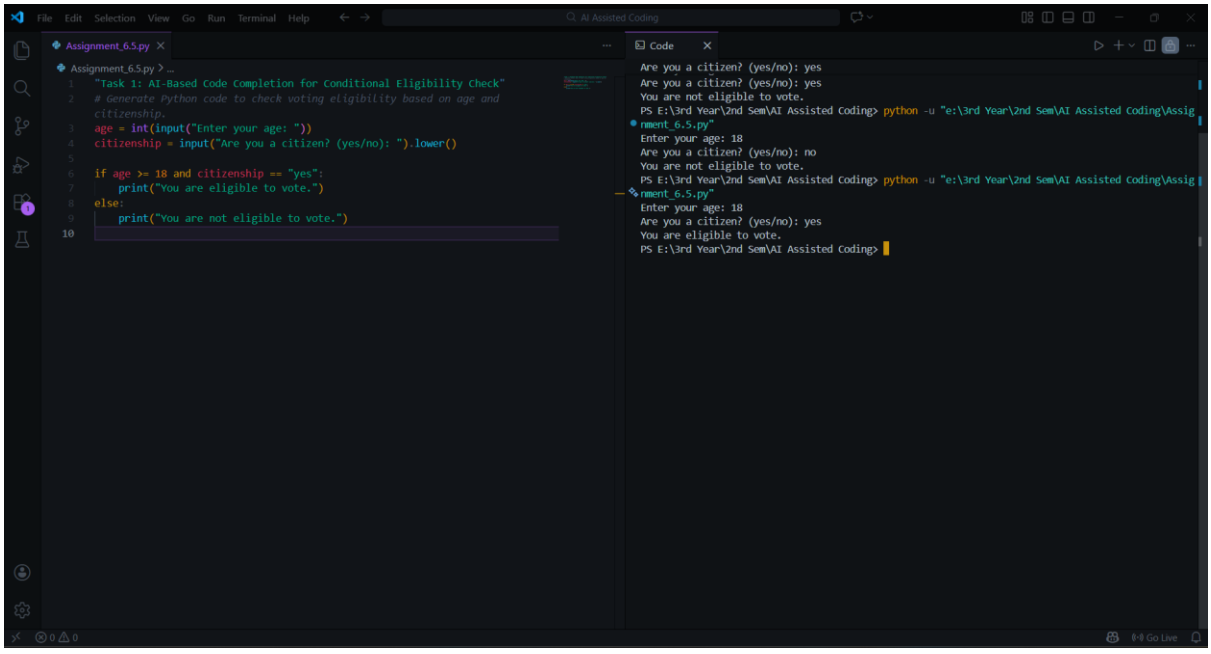
Hall ticket no: 2303A51259

Task 1: AI-Based Code Completion for Conditional Eligibility Check

Prompt:

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

Code & Output:



Explanation:

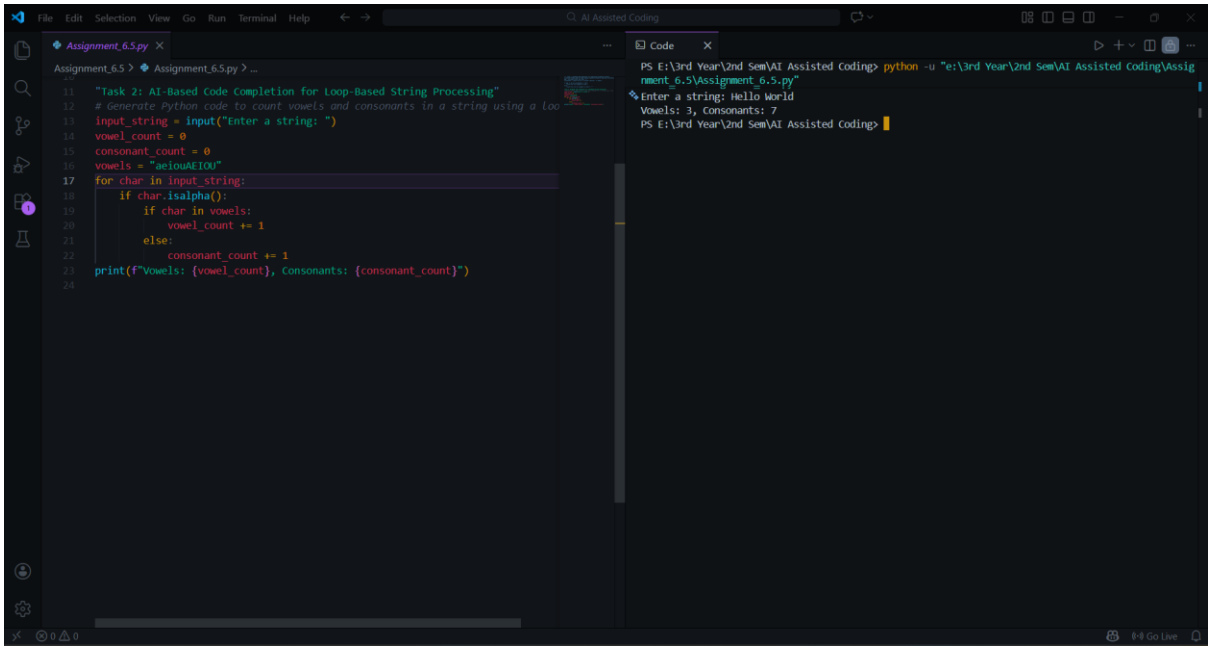
The AI-generated code uses conditional statements to check voting eligibility. It verifies whether the age is 18 or above and whether the user is a citizen. Both conditions must be true for eligibility. This demonstrates correct use of conditional logic generated through AI-based code completion.

Task 2: AI-Based Code Completion for Loop-Based String Processing

Prompt:

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

Code & Output:



Explanation:

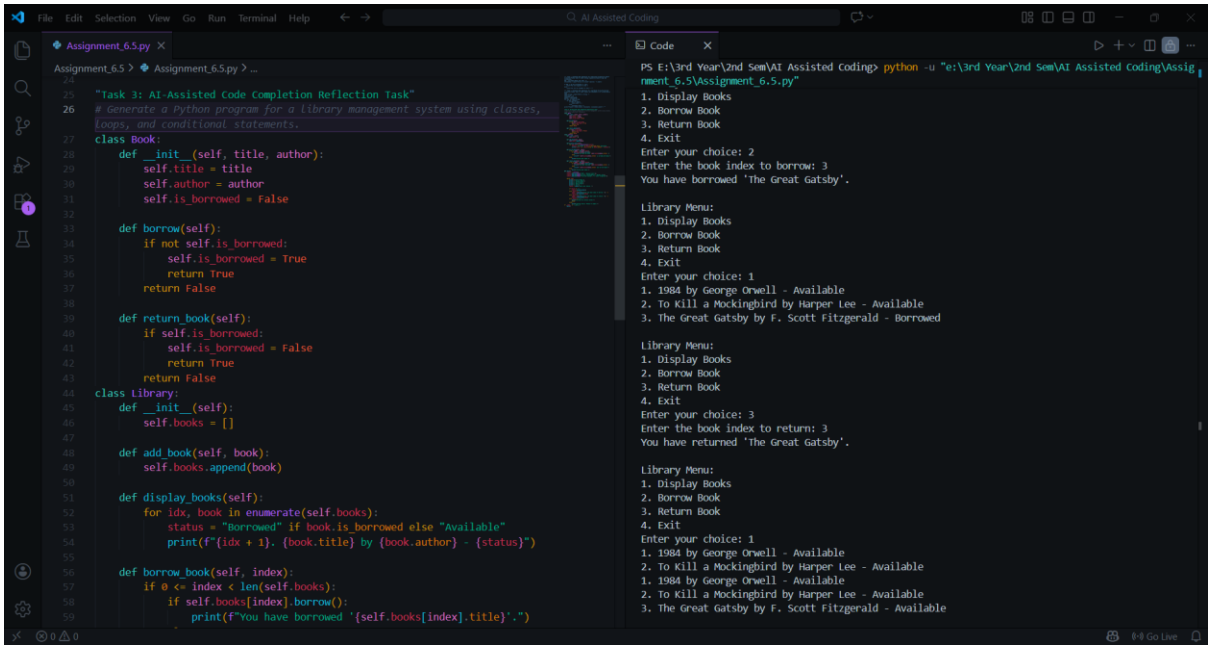
The AI-generated code processes the input string using a loop. Each character is checked to determine whether it is a vowel or a consonant. Alphabetic characters are counted correctly, while non-letter characters are ignored. The output verifies that the logic works as expected.

Task 3: AI-Assisted Code Completion Reflection Task

Prompt:

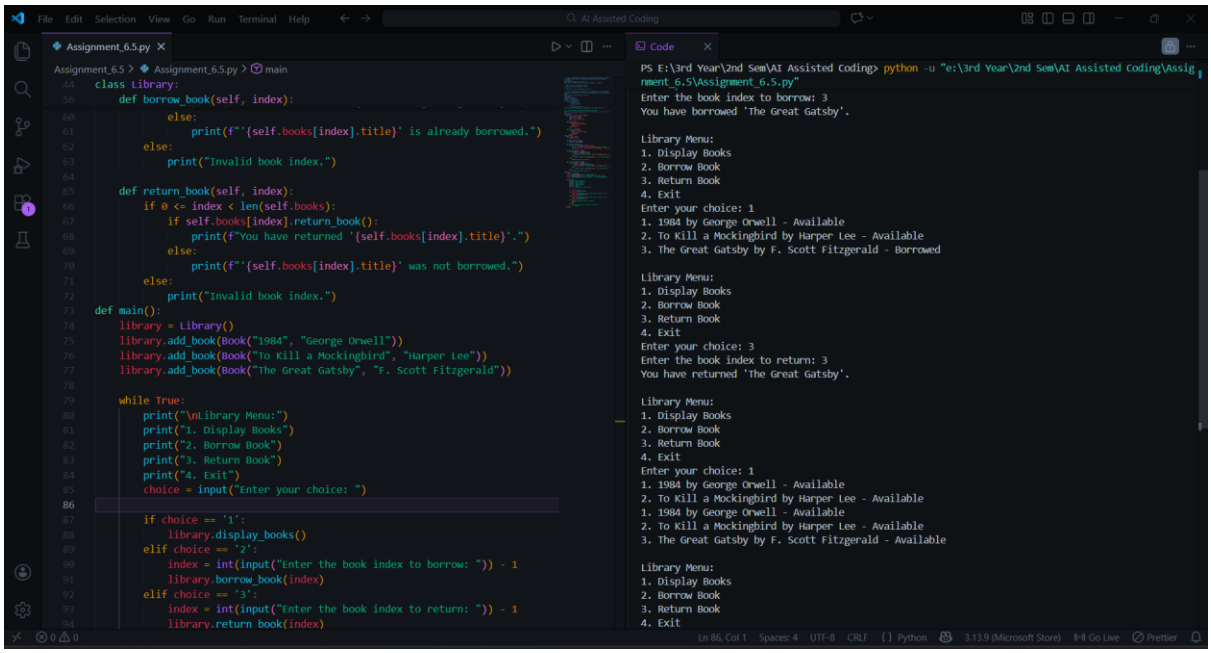
Generate a Python program for a library management system using classes, loops, and conditional statements.

Code & Output:



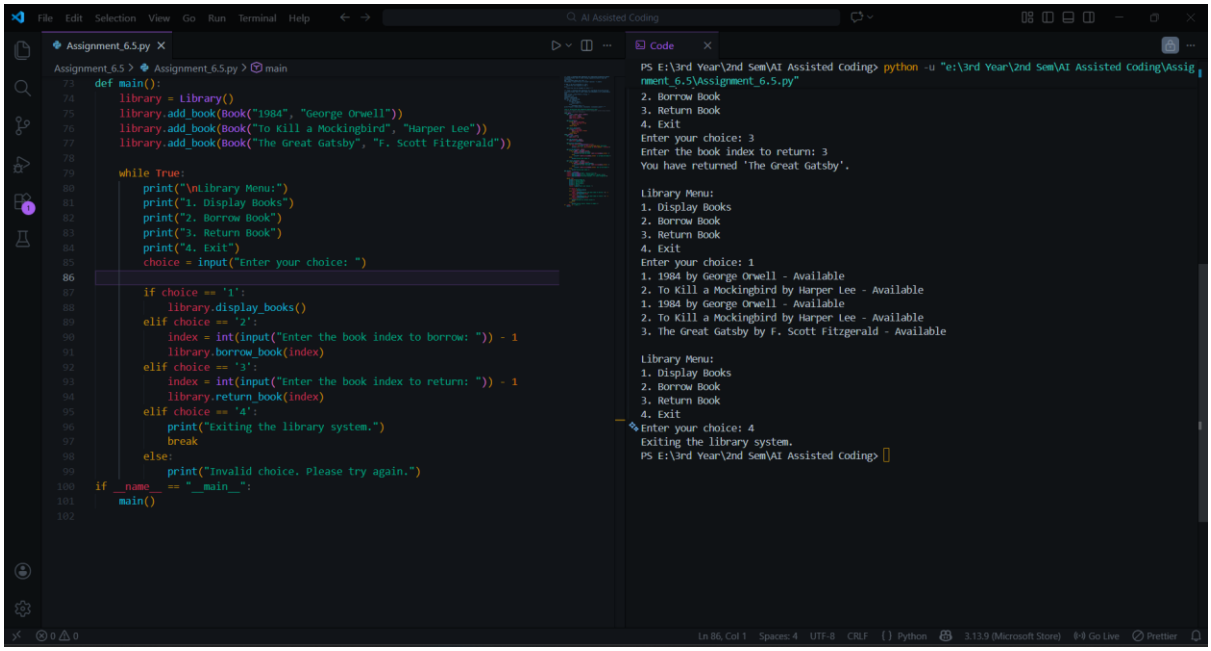
```
"Task 3: AI-Assisted Code Completion Reflection Task"
# Generate a Python program for a library management system using classes,
loops, and conditional statements.
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.is_borrowed = False

    def borrow(self):
        if not self.is_borrowed:
            self.is_borrowed = True
            return True
        return False

    def return_book(self):
        if self.is_borrowed:
            self.is_borrowed = False
            return True
        return False
class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book):
        self.books.append(book)

    def display_books(self):
        for idx, book in enumerate(self.books):
            status = "Borrowed" if book.is_borrowed else "Available"
            print(f"{idx + 1}. {book.title} by {book.author} - {status}")

    def borrow_book(self, index):
        if 0 <= index < len(self.books):
            if self.books[index].borrow():
                print(f"You have borrowed '{self.books[index].title}'.")
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_6.5\Assignment_6.5.py"
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 2
Enter the book index to borrow: 3
You have borrowed 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Borrowed

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 3
Enter the book index to return: 3
You have returned 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Available
```



```
class Library:
    def borrow_book(self, index):
            else:
                print(f"'{self.books[index].title}' is already borrowed.")
        else:
            print("Invalid book index.")

    def return_book(self, index):
        if 0 <= index < len(self.books):
            if self.books[index].return_book():
                print(f"You have returned '{self.books[index].title}'.")
            else:
                print(f"'{self.books[index].title}' was not borrowed.")
        else:
            print("Invalid book index.")
def main():
    library = Library()
    library.add_book(Book("1984", "George Orwell"))
    library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
    library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald"))

    while True:
        print("\nLibrary Menu:")
        print("1. Display Books")
        print("2. Borrow Book")
        print("3. Return Book")
        print("4. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            library.display_books()
        elif choice == '2':
            index = int(input("Enter the book index to borrow: ")) - 1
            library.borrow_book(index)
        elif choice == '3':
            index = int(input("Enter the book index to return: ")) - 1
            library.return_book(index)
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_6.5\Assignment_6.5.py"
Enter the book index to borrow: 3
You have borrowed 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Borrowed

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 3
Enter the book index to return: 3
You have returned 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Available

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
```



Explanation:

The AI-generated program uses a class to represent a library and includes loops and conditional statements for menu-driven interaction. The loop allows continuous user input, and conditionals control program flow. The program correctly demonstrates AI-assisted use of object-oriented programming concepts.

Reflection on AI-Assisted Coding:

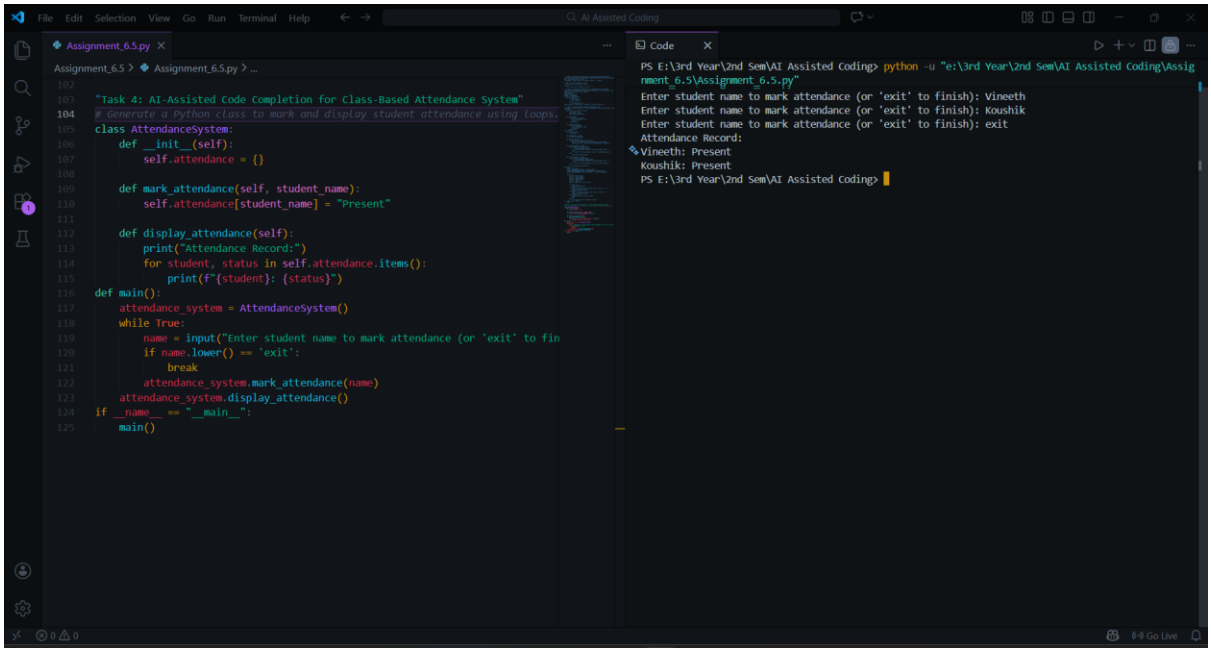
The AI tool generated a complete and functional program quickly. While the logic is correct, the code can be further improved with input validation and advanced features. This task shows that AI is useful for speeding up development but still requires human review and optimization.

Task 4: AI-Assisted Code Completion for Class-Based Attendance System

Prompt:

Generate a Python class to mark and display student attendance using loops.

Code & Output:



Explanation:

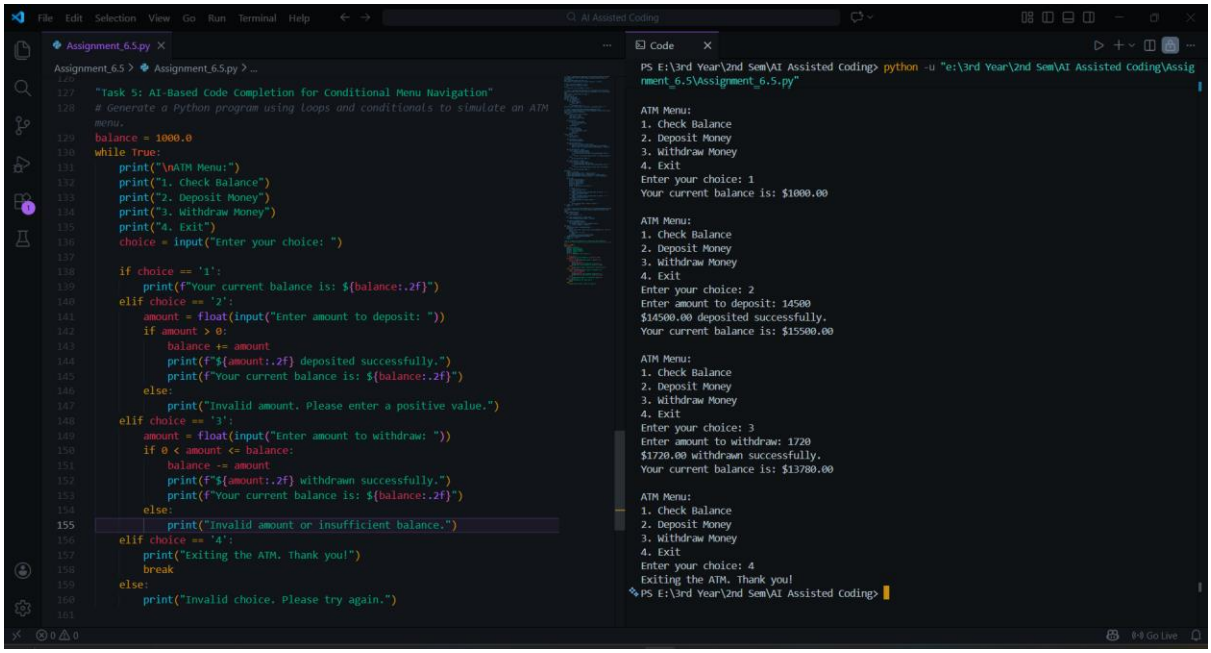
The AI-generated attendance system uses a class to store attendance data. A loop is used to take multiple student entries, and another loop displays the attendance records. The code works correctly and demonstrates class-based AI code completion.

Task 5: AI-Based Code Completion for Conditional Menu Navigation

Prompt:

Generate a Python program using loops and conditionals to simulate an ATM menu.

Code & Output:



Explanation:

The AI-generated ATM program uses a loop to display the menu repeatedly and conditional statements to handle user choices. The logic correctly updates the balance and prevents invalid withdrawals. This task demonstrates effective AI-based code completion for menu-driven programs.

Final Conclusion:

This experiment shows how AI-based code completion tools can generate useful Python code involving classes, loops, and conditionals. While AI speeds up development, developers must still review logic, handle edge cases, and ensure ethical and responsible use of AI-generated code.