

ASSIGNMENT-8.5

Name: A.Saiteja

Ht.No:2303A52167

Batch:35

Task Description #1 (Username Validator – Apply AI in Authentication Context)

- Task: Use AI to generate at least 3 assert test cases for a function `is_valid_username(username)` and then implement the function using Test-Driven Development principles.
- Requirements:
 - Username length must be between 5 and 15 characters.
 - Must contain only alphabets and digits.
 - Must not start with a digit.
 - No spaces allowed.

Example Assert Test Cases:

```
assert is_valid_username("User123") == True
```

```
assert is_valid_username("12User") == False
```

```
assert is_valid_username("Us er") == False
```

Expected Output #1:

- Username validation logic successfully passing all AI-generated test cases.

Output:

```
username validator

Gemini
+ Function
+def is_valid_username(username):
+    if not isinstance(username, str):
+        return False
+    if len(username) < 5:
+        return False
+    if not username.isalnum():
+        return False
+    if username[0].isdigit():
+        return False
+    return True
+
+Test cases
+assert is_valid_username('abc')
+assert is_valid_username('12345')
+assert is_valid_username('a1b2c3')
```

#1 (Username Validator – Apply AI in Authentication Context)

Empty cell

Let's implement the username validator function and its test cases in the selected cell.

Accept & Run Accept Cancel

What can I help you build?

Gemini 2.5 Flash ▾

Task Description #2 (Even–Odd & Type Classification – Apply AI for Robust Input Handling)

- Task: Use AI to generate at least 3 assert test cases for a function `classify_value(x)` and implement it using conditional logic and loops.

- Requirements:

- If input is an integer, classify as "Even" or "Odd".
- If input is 0, return "Zero".
- If input is non-numeric, return "Invalid Input".

Example Assert Test Cases:

```
assert classify_value(8) == "Even"
assert classify_value(7) == "Odd"
assert classify_value("abc") == "Invalid Input"
```

Expected Output #2:

- Function correctly classifying values and passing all test

cases.

Output:

The screenshot shows the Gemini AI interface. On the left, there's a code editor window titled "Gemini" containing Python code. The code defines a function `classify_input` that checks if a value is even or odd. It also includes some test cases using `assert` statements. A modal dialog box is open in the center, titled "Empty cell". It contains a brief description: "Let's create a Python function `classify_input` that determines if an input is an even or odd integer, or classifies its type if it's not an integer, and add some test cases." Below the description are three buttons: "Accept & Run" (with a checkmark), "Accept", and "Cancel". At the bottom of the dialog is a text input field asking "What can I help you build?". The status bar at the bottom right says "Gemini 2.5 Flash".

Task Description #3 (Palindrome Checker – Apply AI for String Normalization)

- Task: Use AI to generate at least 3 assert test cases for a function `is_palindrome(text)` and implement the function.

- Requirements:

- Ignore case, spaces, and punctuation.
 - Handle edge cases such as empty strings and single characters.

Example Assert Test Cases:

```
assert is_palindrome("Madam") == True
```

```
assert is_palindrome("A man a plan a canal Panama") ==  
True
```

```
assert is_palindrome("Python") == False
```

Expected Output #3:

- Function correctly identifying palindromes and passing all AI-generated tests.

Output:

```

Task-03
Gemini
+
def is_palindrome(text):
    # Normalize the string: convert to lowercase and remove non-alphanumeric characters
    normalized_text = ''.join([c.lower() for c in text if c.isalnum()])
    # Check if the normalized text is equal to its reverse
    return normalized_text == normalized_text[::-1]

# Test cases
assert is_palindrome("Racecar") == True
assert is_palindrome("A man a plan a canal Panama") == True
assert is_palindrome("Hello World") == False
assert is_palindrome("No lemon no melon") == True
assert is_palindrome("Python") == False
assert is_palindrome("Madam") == True
assert is_palindrome("Was it a car or a cat I saw") == True

```

M #3 (Palindrome Checker – Apply AI for String Normalization)

Empty cell

Let's create a Python function `is_palindrome` that checks if a string is a palindrome, ignoring non-alphanumeric characters and case, and add some

Accept & Run ✓ Accept X Cancel

What can I help you build?

Gemini 2.5 Flash ▶

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Task Description #4 (BankAccount Class – Apply AI for Object-Oriented Test-Driven Development)

- Task: Ask AI to generate at least 3 assert-based test cases for

a BankAccount class and then implement the class.

- Methods:

- `deposit(amount)`

- `withdraw(amount)`

- `get_balance()`

Example Assert Test Cases:

```
acc = BankAccount(1000)
```

```
acc.deposit(500)
```

```
assert acc.get_balance() == 1500
```

```
acc.withdraw(300)
```

```
assert acc.get_balance() == 1200
```

Expected Output #4:

- Fully functional class that passes all AI-generated assertions.

Output:

The screenshot shows the Gemini AI interface with the following details:

- Code Completion Panel (Top Left):** Shows the `BankAccount` class definition with methods `deposit` and `withdraw`. A tooltip says "#4 (BankAccount Class – Apply AI for Object-Oriented Test-Driven Development) Empty cell". Below it, a note says "Let's create a BankAccount class with methods for deposit, withdraw". Buttons for "Accept & Run", "Accept", and "Cancel" are shown.
- Code Editor (Bottom Left):** Displays a series of test cases for the `BankAccount` class. The tests cover negative initial balance, float initial balance, float deposit amount, float withdraw amount, and a general success case. All tests pass, with the message "... All BankAccount tests passed!" at the bottom.
- Status Bar (Top Right):** Shows RAM usage (~1.5GB), Disk usage (~100MB), and other system icons.

Task Description #5 (Email ID Validation – Apply AI for Data Validation)

- Task: Use AI to generate at least 3 assert test cases for a function `validate_email(email)` and implement the function.

- Requirements:
 - Must contain @ and .
 - Must not start or end with special characters.
 - Should handle invalid formats gracefully.

Example Assert Test Cases:

```
assert validate_email("user@example.com") == True
```

```
assert validate_email("userexample.com") == False
```

```
assert validate_email("@gmail.com") == False
```

Expected Output #5:

- Email validation function passing all AI-generated test cases and handling edge cases correctly.

Output:

The screenshot shows a code editor interface with a Python script titled "Task-05". The script contains a function `is_valid_email` that checks if an email address is valid based on common patterns. It includes a regular expression pattern and several test cases using the `assert` statement.

A modal dialog box from Gemini AI is overlaid on the code editor. The dialog has the following content:

- M #5 (Email ID Validation – Apply AI for Data Validation)**
- Empty cell**
- Let's create a Python function `is_valid_email` that validates email addresses based on common patterns, and include test cases for various**
- Accept & Run Accept Cancel**
- What can I help you build?**
- Gemini 2.5 Flash**

The code editor also shows standard navigation and status bars at the top and bottom.