Assignment 5.1

Course Title: Ethical Foundations – Responsible AI

Lab Title: Responsible AI Coding Practices

Lab No: 5

Week: 3

Day: Monday

Name: JyoshnoPranav

Roll No: 2303A53016

Aim

To understand ethical risks in AI-generated code and apply responsible AI coding practices focusing on privacy, security, transparency, and accountability.

Objectives

- Identify insecure coding patterns generated by AI
- Analyze privacy and security risks
- Ensure transparency and explainability in algorithms
- Understand developer responsibility in AI-assisted programming

---

Tools Used

- VS Code

Task-Wise Implementation

Task 1: Privacy in API Usage Problem Statement

Generate a Python program to fetch weather data securely without exposing API keys.

AI Prompt Used

Generate a Python program to fetch weather data from a weather API without hardcoding the API key.

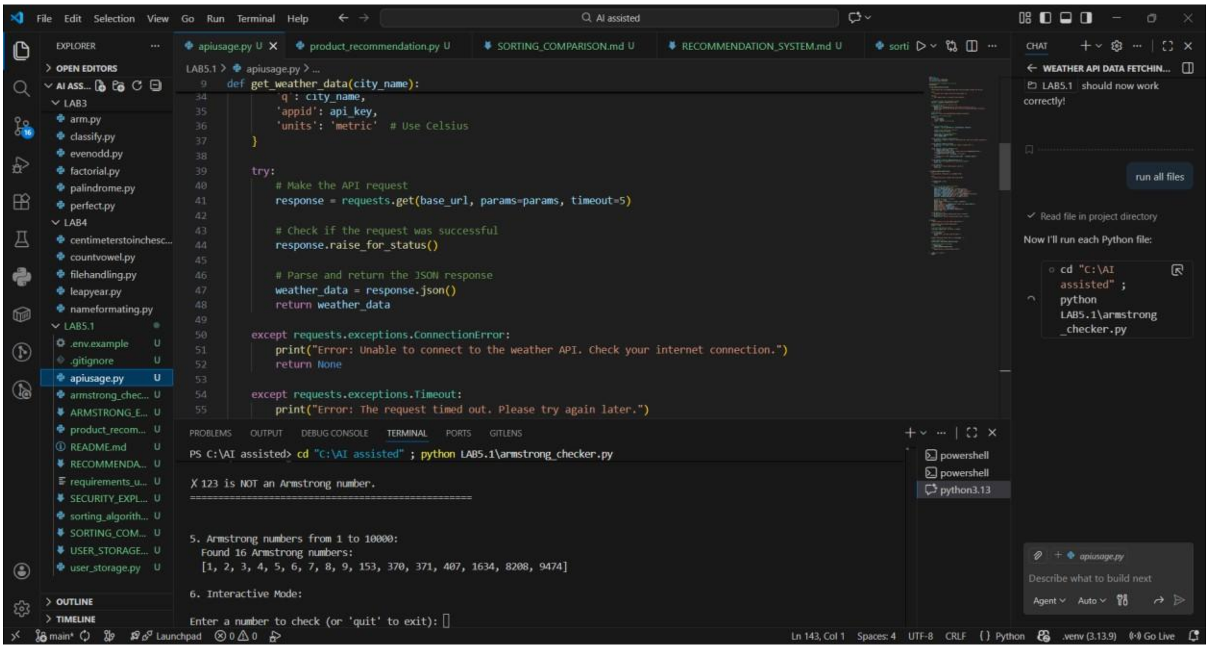
Use environment variables to store and access the API key securely.

Include basic error handling.

Observation

- AI-generated code avoided hardcoded API keys
- Environment variables were used for security

Secure Code



```python
def get_weather_data(city_name):
        'q': city_name,
        'appid': api_key,
        'units': 'metric'  # Use Celsius
    }

    try:
        # Make the API request
        response = requests.get(base_url, params=params, timeout=5)

        # Check if the request was successful
        response.raise_for_status()

        # Parse and return the JSON response
        weather_data = response.json()
        return weather_data

    except requests.exceptions.ConnectionError:
        print("Error: Unable to connect to the weather API. Check your internet connection.")
        return None

    except requests.exceptions.Timeout:
        print("Error: The request timed out. Please try again later.")
```

```
PS C:\AI assisted> cd "C:\AI assisted" ; python LAB5.1\armstrong_checker.py

X 123 is NOT an Armstrong number.
==================================================


5. Armstrong numbers from 1 to 10000:
   Found 16 Armstrong numbers:
   [1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208, 9474]

6. Interactive Mode:

Enter a number to check (or 'quit' to exit): 
```

WEATHER API DATA FETCHIN...

LAB5.1 should now work correctly!

run all files

✓ Read file in project directory

Now I'll run each Python file:

```
cd "C:\AI assisted" ; python LAB5.1\armstrong_checker.py
```

Ethical Analysis

Using environment variables protects sensitive credentials and prevents unauthorized misuse.

Task 2: Privacy & Security in File Handling Problem Statement

Store user data securely without exposing sensitive information.

AI Prompt Used

Generate a Python program to store user details (name, email, password) securely.

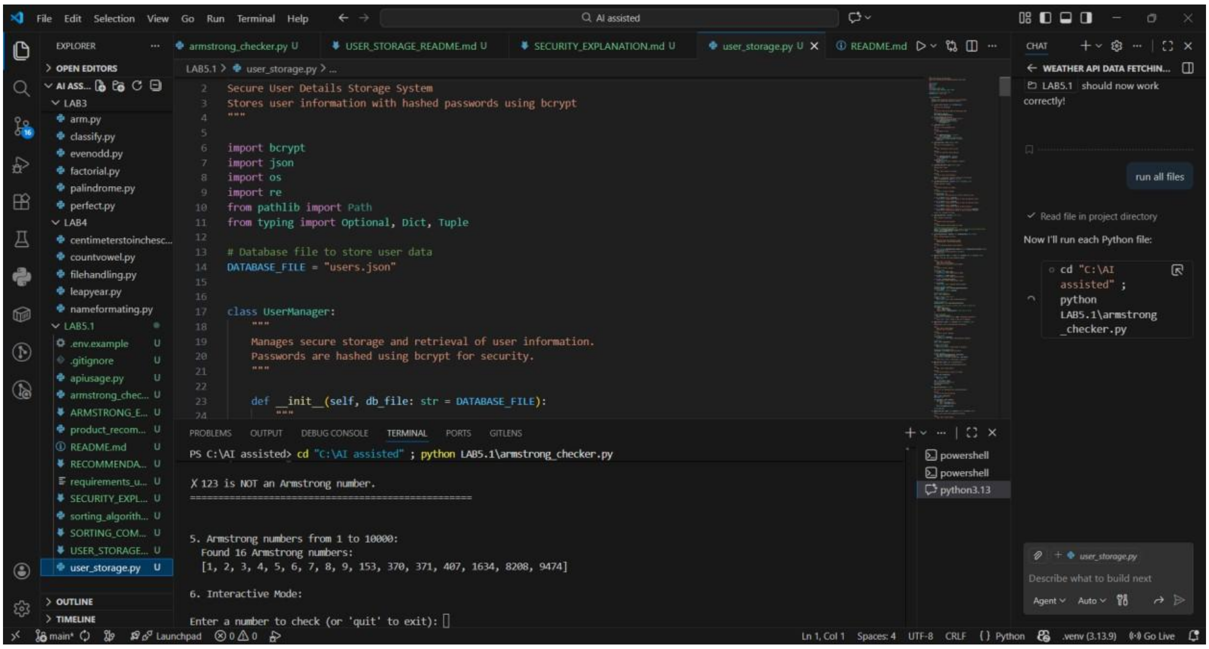
Do not store passwords in plain text.

Use hashing for password storage and explain why this approach is secure.

Privacy Risk Identified

- Plain-text password storage is insecure

Secure Code



```python
Secure User Details Storage System
Stores user information with hashed passwords using bcrypt
"""

import bcrypt
import json
import os
import re
from pathlib import Path
from typing import Optional, Dict, Tuple

# Database file to store user data
DATABASE_FILE = "users.json"


class UserManager:
    """
    Manages secure storage and retrieval of user information.
    Passwords are hashed using bcrypt for security.
    """

    def __init__(self, db_file: str = DATABASE_FILE):
        """
```

```
PS C:\AI assisted> cd "C:\AI assisted" ; python LAB5.1\armstrong_checker.py

X 123 is NOT an Armstrong number.
==================================================


5. Armstrong numbers from 1 to 10000:
  Found 16 Armstrong numbers:
  [1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208, 9474]

6. Interactive Mode:

Enter a number to check (or 'quit' to exit): 
```

LAB5.1 should now work correctly!

run all files

✓ Read file in project directory

Now I'll run each Python file:

```
cd "C:\AI assisted" ; python LAB5.1\armstrong_checker.py
```

Ethical Analysis

Hashing ensures passwords cannot be recovered even if the file is leaked.

Task 3: Transparency in Algorithm Design

Problem Statement

Design an Armstrong number checking program with clear explanation.

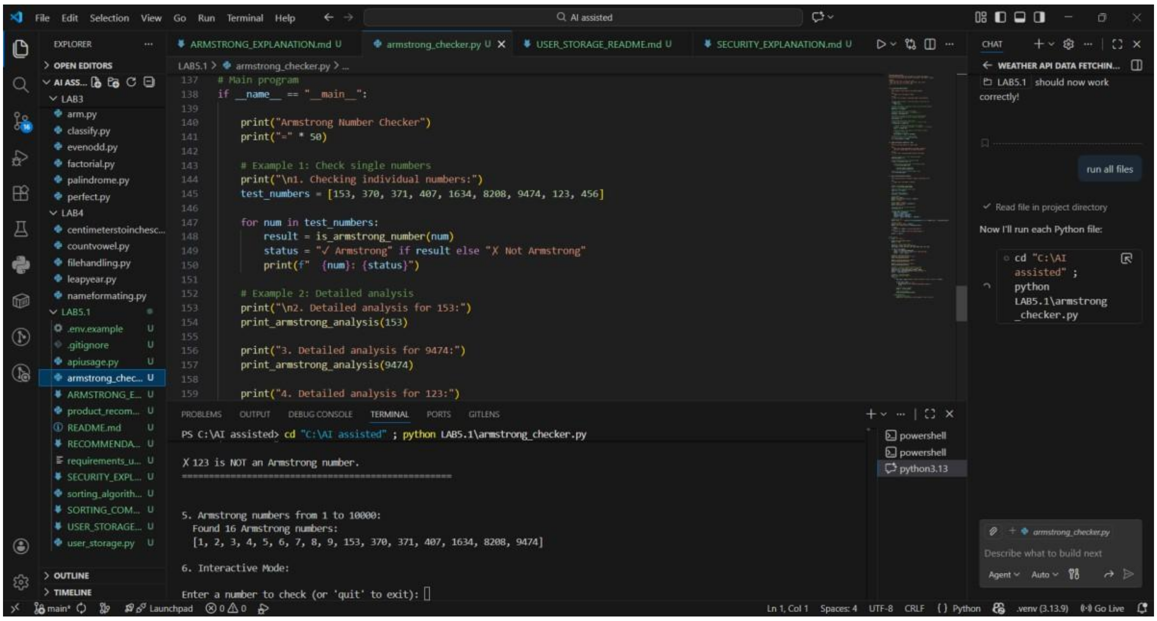
AI Prompt Used

Generate a Python function to check whether a given number is an Armstrong number.

Add clear comments to every important line of code.

Also explain the code line-by-line in simple terms.

Implementation



```python
# Main program
if __name__ == "__main__":

    print("Armstrong Number Checker")
    print("=" * 50)

    # Example 1: Check single numbers
    print("\n1. Checking individual numbers:")
    test_numbers = [153, 370, 371, 407, 1634, 8208, 9474, 123, 456]

    for num in test_numbers:
        result = is_armstrong_number(num)
        status = "✓ Armstrong" if result else "X Not Armstrong"
        print(f"  {num}: {status}")

    # Example 2: Detailed analysis
    print("\n2. Detailed analysis for 153:")
    print_armstrong_analysis(153)

    print("3. Detailed analysis for 9474:")
    print_armstrong_analysis(9474)

    print("4. Detailed analysis for 123:")
```

```
PS C:\AI assisted> cd "C:\AI assisted" ; python LAB5.1\armstrong_checker.py

X 123 is NOT an Armstrong number.
==================================================


5. Armstrong numbers from 1 to 10000:
  Found 16 Armstrong numbers:
  [1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208, 9474]

6. Interactive Mode:

Enter a number to check (or 'quit' to exit):
```

WEATHER API DATA FETCHIN...

LAB5.1 should now work correctly!

run all files

✓ Read file in project directory

Now I'll run each Python file:

```
cd "C:\AI assisted" ; python LAB5.1\armstrong_checker.py
```

Transparency Evaluation

The explanation clearly matches the program logic, ensuring understandability.

Task 4: Transparency in Algorithm Comparison Problem Statement

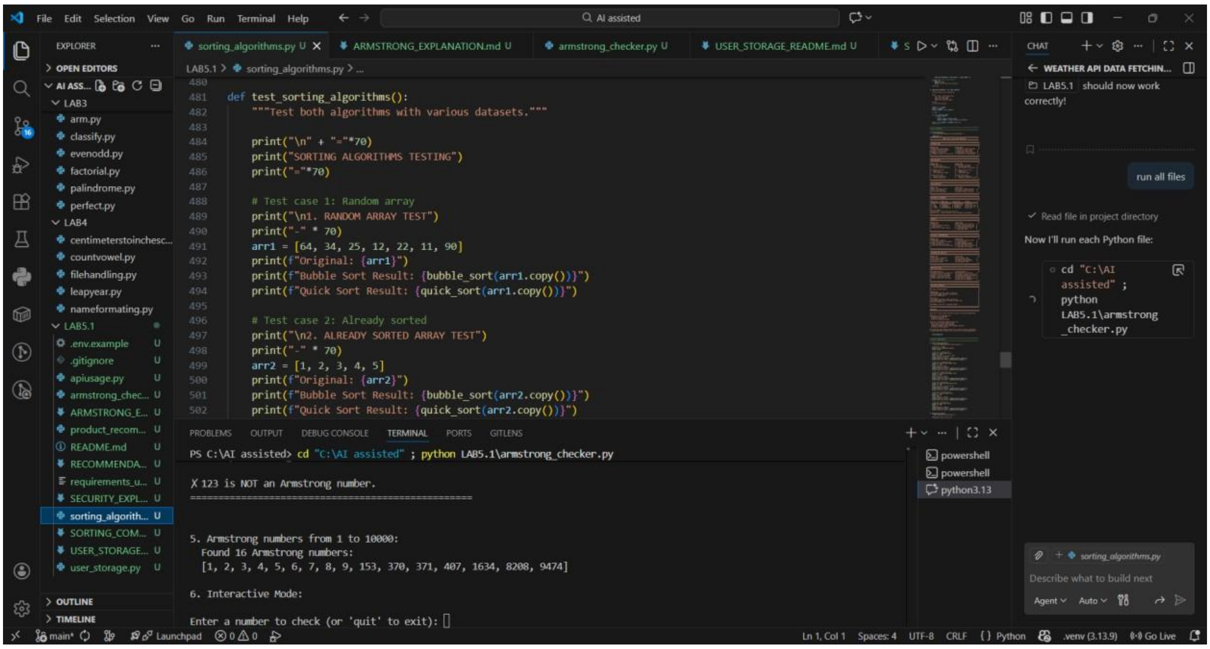
Implement and compare two sorting algorithms.

AI Prompt Used

Generate Python code for Bubble Sort and Quick Sort.

Include step-by-step comments explaining how each algorithm works.

Compare both algorithms in terms of logic, time complexity, and efficiency.



```python
def test_sorting_algorithms():
    """Test both algorithms with various datasets."""

    print("\n" + "="*70)
    print("SORTING ALGORITHMS TESTING")
    print("="*70)

    # Test case 1: Random array
    print("\n1. RANDOM ARRAY TEST")
    print("-" * 70)
    arr1 = [64, 34, 25, 12, 22, 11, 90]
    print(f"Original: {arr1}")
    print(f"Bubble Sort Result: {bubble_sort(arr1.copy())}")
    print(f"Quick Sort Result: {quick_sort(arr1.copy())}")

    # Test case 2: Already sorted
    print("\n2. ALREADY SORTED ARRAY TEST")
    print("-" * 70)
    arr2 = [1, 2, 3, 4, 5]
    print(f"Original: {arr2}")
    print(f"Bubble Sort Result: {bubble_sort(arr2.copy())}")
    print(f"Quick Sort Result: {quick_sort(arr2.copy())}")
```

```
PS C:\AI assisted> cd "C:\AI assisted" ; python LAB5.1\armstrong_checker.py

X 123 is NOT an Armstrong number.
==================================================


5. Armstrong numbers from 1 to 10000:
  Found 16 Armstrong numbers:
  [1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208, 9474]

6. Interactive Mode:

Enter a number to check (or 'quit' to exit):
```

WEATHER API DATA FETCHIN...

LAB5.1 should now work correctly!

run all files

✓ Read file in project directory

Now I'll run each Python file:

```
cd "C:\AI assisted" ; python LAB5.1\armstrong_checker.py
```

Comparison

Algorithm	**Time Complexity**	**Efficiency**
Bubble Sort	$O(n^2)$	Low
Quick Sort	$O(n \log n)$	High

Task 5: Transparency in AI Recommendations Problem Statement

Create an explainable recommendation system.

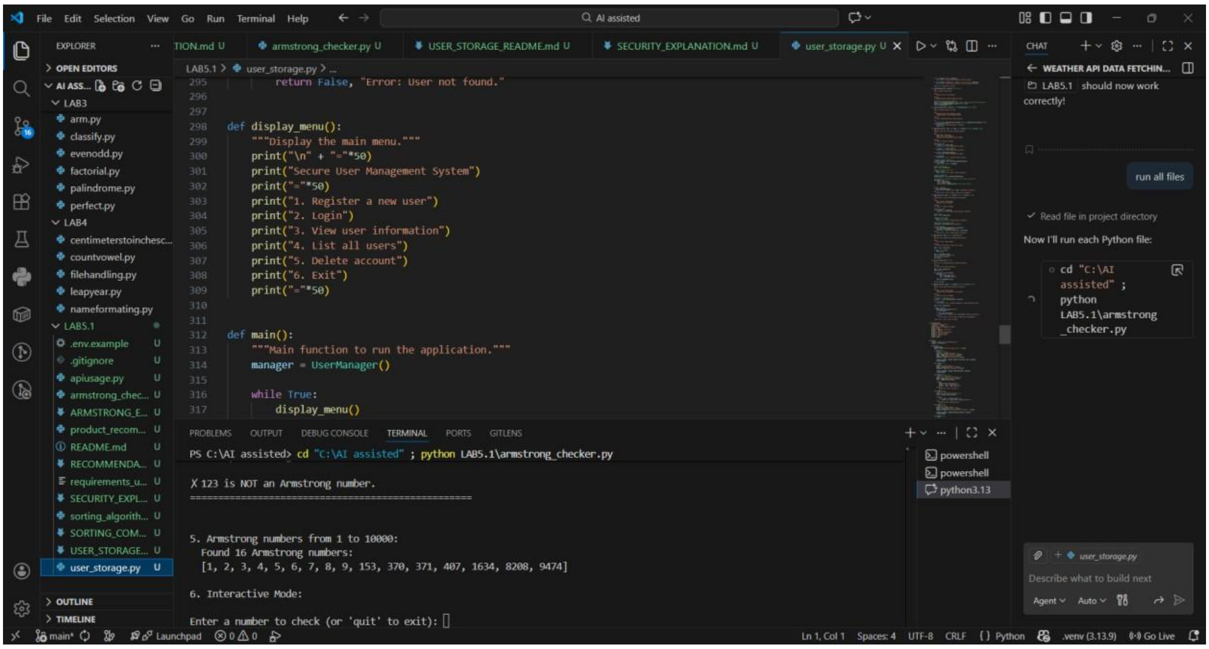
AI Prompt Used

Generate a simple product recommendation system in Python.

For each recommended product, also provide a clear explanation of why it was recommended.

Ensure the recommendations are explainable and transparent.

Implementation



```python
            return False, "Error: User not found."


def display_menu():
    """Display the main menu."""
    print("\n" + "="*50)
    print("Secure User Management System")
    print("="*50)
    print("1. Register a new user")
    print("2. Login")
    print("3. View user information")
    print("4. List all users")
    print("5. Delete account")
    print("6. Exit")
    print("="*50)


def main():
    """Main function to run the application."""
    manager = UserManager()

    while True:
        display_menu()
```

```
PS C:\AI assisted> cd "C:\AI assisted" ; python LAB5.1\armstrong_checker.py

X 123 is NOT an Armstrong number.
============================================================


5. Armstrong numbers from 1 to 10000:
   Found 16 Armstrong numbers:
   [1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208, 9474]

6. Interactive Mode:

Enter a number to check (or 'quit' to exit):
```

LAB5.1 should now work correctly!

run all files

✓ Read file in project directory

Now I'll run each Python file:

```
cd "C:\AI assisted" ; python LAB5.1\armstrong_checker.py
```

Transparency Evaluation

Each recommendation includes a reason, making the system explainable.

Result

All tasks were implemented successfully using ethical AI coding practices with proper privacy, security, and transparency.

Conclusion

AI-generated code must be reviewed by developers to ensure ethical compliance. Human accountability is essential in responsible AI usage.