ASSIGNMENT – 5.3

Thumma Hasini | 2303A52076 | Batch – 37

Task – 1 : Privacy and Data Security in AI-Generated Code

Use an AI tool to generate a simple login system in Python. Analyze the generated code to check:

• Whether credentials are hardcoded

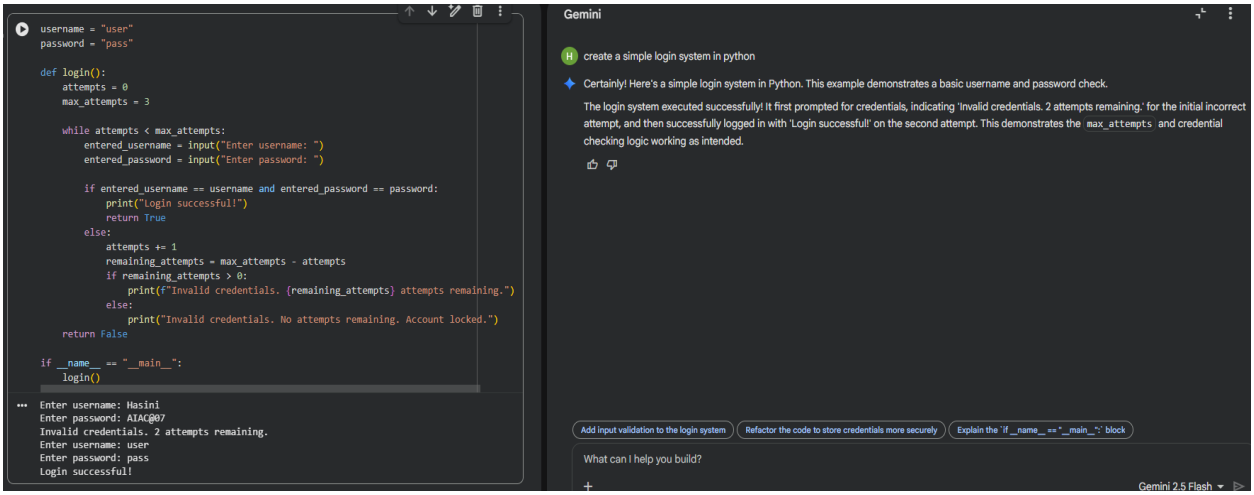
• Whether passwords are stored or compared in plain text

• Whether insecure logic is used

Then, revise the code to improve security (e.g., avoid hardcoding, use input

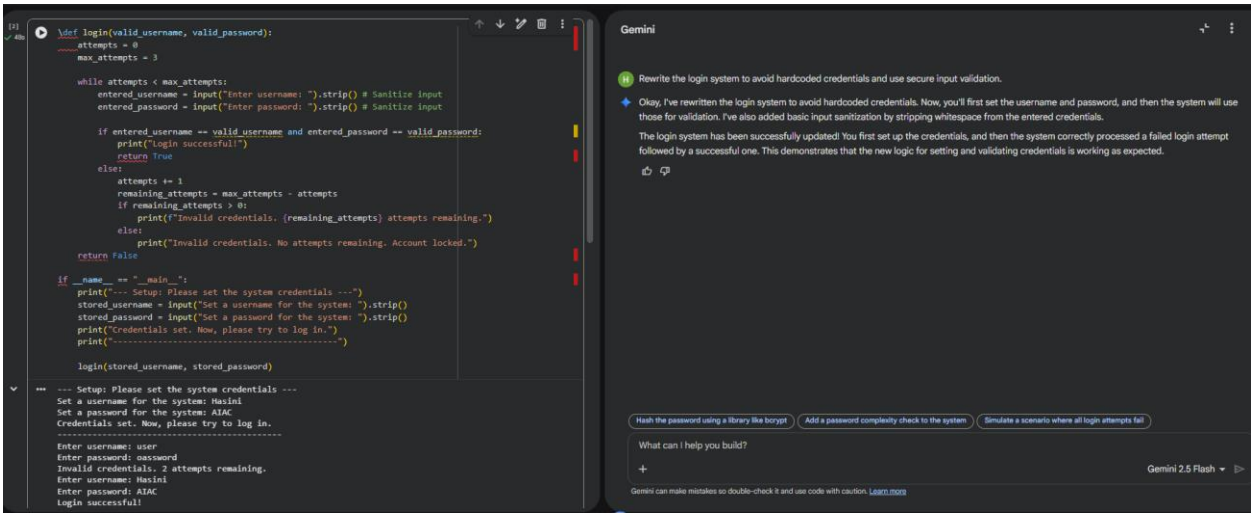
validation).

Prompt Used : create a simple login system in python



Analysis : The generated login system uses hardcoded username and password values, which is a major security risk. Passwords are stored in plain text, making them visible to anyone who accesses the code. Users cannot set their own credentials, and sensitive data is not protected. This violates basic privacy and secure coding practices.

Improved Prompt : Rewrite the login system to avoid hardcoded credentials and use secure input validation.



Explanation : This task highlights security risks in AI-generated authentication code. The original code used hardcoded credentials and plain text password comparison, which is insecure. The revised version improves security by avoiding hardcoding and validating user input.

Task – 2: Bias Detection in AI-Generated Decision Systems

Use AI prompts such as:

• “Create a loan approval system”

• Vary applicant names and genders in prompts

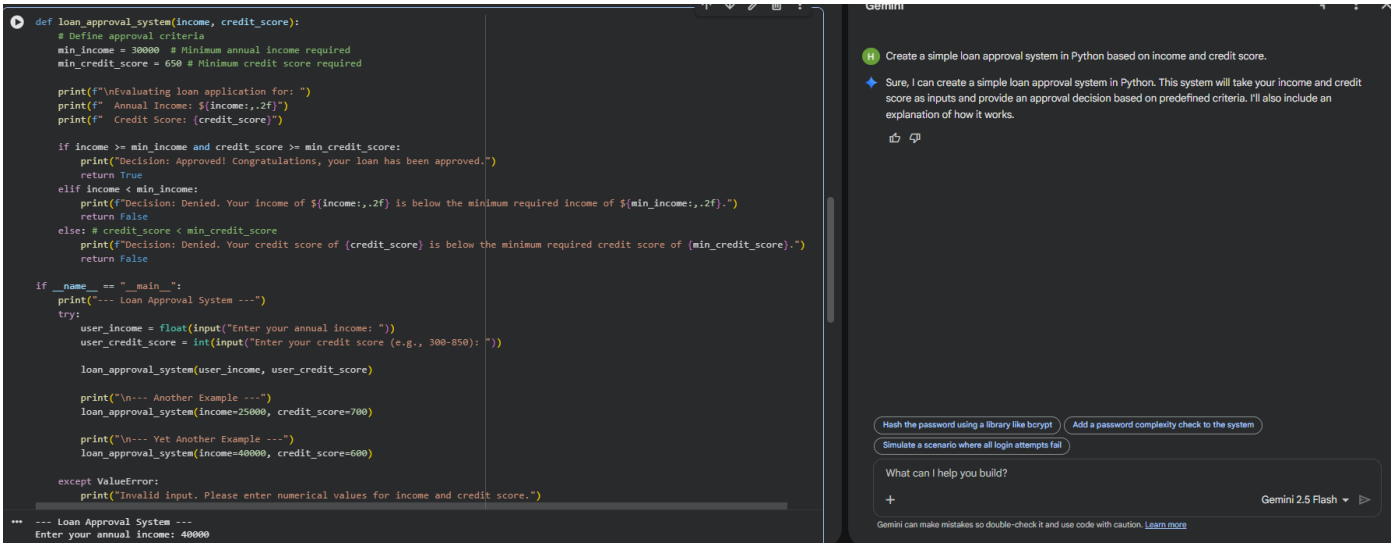
Analyze whether:

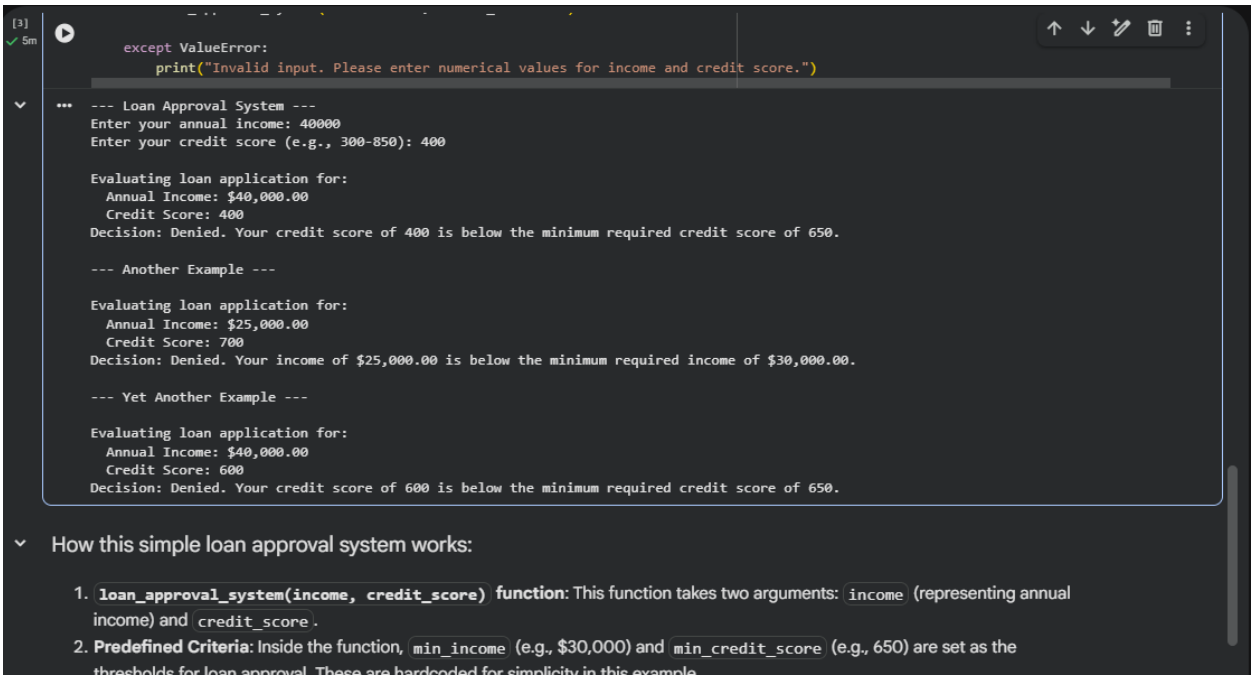
• The logic treats certain genders or names unfairly

• Approval decisions depend on irrelevant personal attributes

Suggest methods to reduce or remove bias.

Prompt Used: Create a simple loan approval system in Python based on income and credit score.





Explanation: The AI-generated loan approval system makes decisions based only on income and credit score. No personal or sensitive attributes such as gender, name, or age are used in the decision-making process. This ensures that the system does not show bias toward any individual or group. Using only financial factors helps maintain fairness and ethical behavior in AI systems.

Task – 3: Transparency and Explainability in AI-Generated Code (Recursive Binary Search)

Use an AI tool to generate a Python program that:

• Implements Binary Search using recursion

• Searches for a given element in a sorted list

• Includes:

o Clear inline comments

o A step-by-step explanation of the recursive logic

After generating the code, analyze:

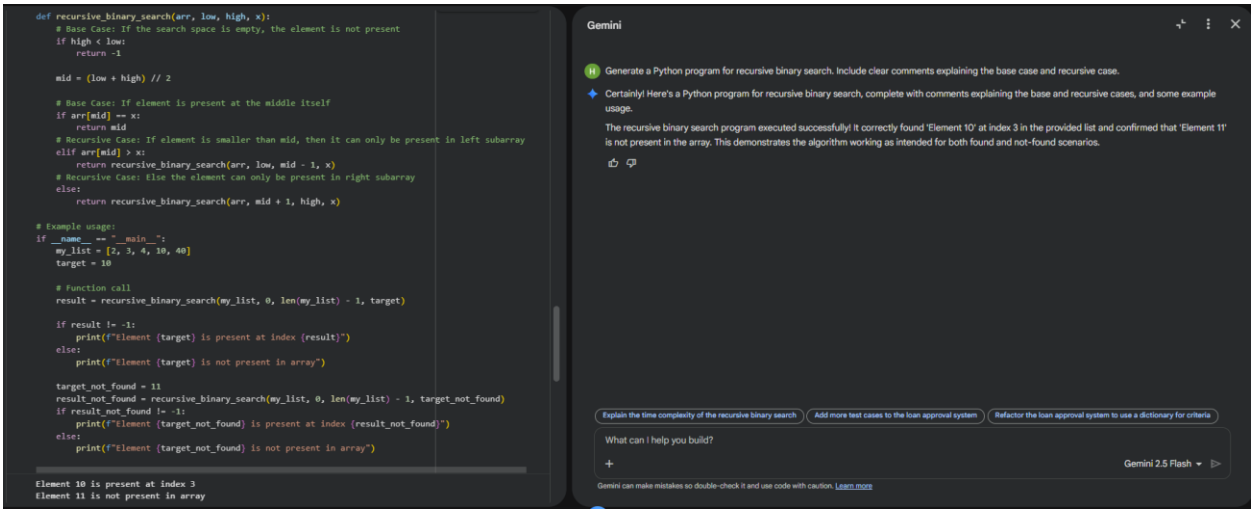
• Whether the explanation clearly describes the base case and recursive

case

• Whether the comments correctly match the code logic

• Whether the code is understandable for beginner-level students

Prompt Used : Generate a Python program for recursive binary search. Include clear comments explaining the base case and recursive case.



Explanation: In this task, the recursive binary search code is easy to understand because clear comments explain what each part does. The base case and recursive calls are clearly mentioned, so it is simple to follow how the algorithm works step by step. This makes the program transparent and helps users trust the logic and results.

Task - 4: Ethical Evaluation of AI-Based Scoring Systems

Ask an AI tool to generate a job applicant scoring system based on features such as:

• Skills

• Experience

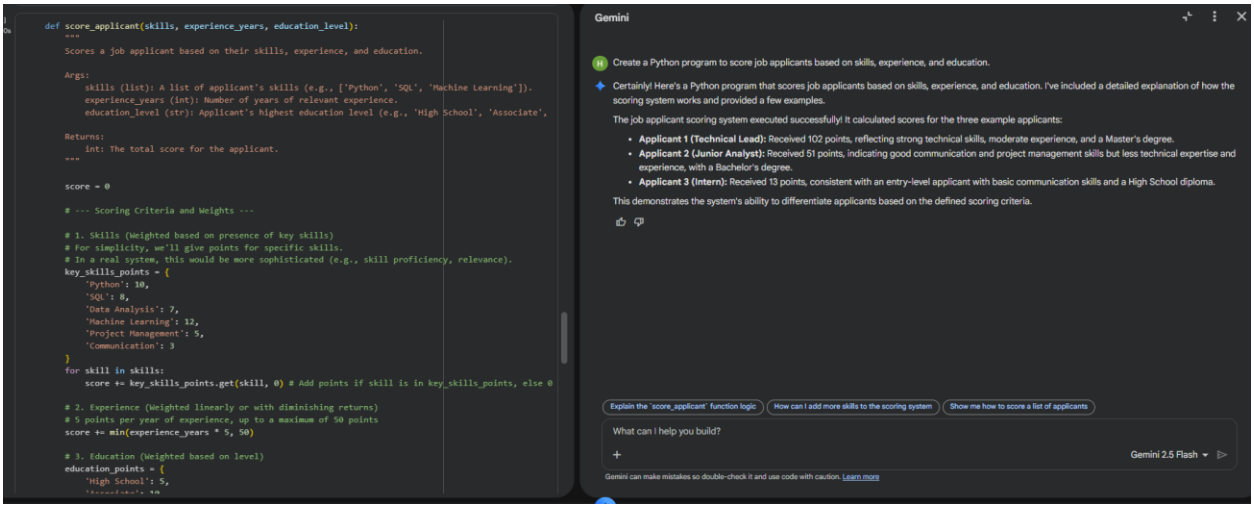
• Education

Analyze the generated code to check:

• Whether gender, name, or unrelated features influence scoring

• Whether the logic is fair and objective

Prompt Used: Create a Python program to score job applicants based on skills, experience, and education.



```
            'High School': 5,
            'Associate': 10,
            'Bachelor': 20,
            'Master': 30,
            'PhD': 40
        }
        score += education_points.get(education_level, 0)

        return score

    if __name__ == "__main__":
        print("--- Job Applicant Scoring System ---")

        # Example Applicant 1: Strong technical skills, moderate experience, Master's degree
        applicant1_skills = ['Python', 'SQL', 'Machine Learning', 'Data Analysis']
        applicant1_experience = 7
        applicant1_education = 'Master'
        applicant1_score = score_applicant(applicant1_skills, applicant1_experience, applicant1_education)
        print(f"\nApplicant 1 (Technical Lead): {applicant1_score} points")

        # Example Applicant 2: Good communication, less technical, less experience, Bachelor's degree
        applicant2_skills = ['Communication', 'Project Management', 'SQL']
        applicant2_experience = 3
        applicant2_education = 'Bachelor'
        applicant2_score = score_applicant(applicant2_skills, applicant2_experience, applicant2_education)
        print(f"Applicant 2 (Junior Analyst): {applicant2_score} points")

        # Example Applicant 3: Entry-level, basic skills, High School diploma
        applicant3_skills = ['Communication']
        applicant3_experience = 1
        applicant3_education = 'High School'
        applicant3_score = score_applicant(applicant3_skills, applicant3_experience, applicant3_education)
        print(f"Applicant 3 (Intern): {applicant3_score} points")
```

```
--- Job Applicant Scoring System ---

Applicant 1 (Technical Lead): 102 points
Applicant 2 (Junior Analyst): 51 points
Applicant 3 (Intern): 13 points
```

Explanation : In this task, the job applicant scoring system evaluates candidates based only on skills, experience, and education, which are directly related to job performance. The program does not use personal details like name, gender, or age, so it avoids unfair bias. Since the scoring criteria are clearly defined and job-relevant, the system follows ethical principles and promotes fair decision-making.

Task - 5: Inclusiveness and Ethical Variable Design

Use an AI tool to generate a Python code snippet that processes user or employee details.

Analyze the code to identify:

• Gender-specific variables (e.g., male, female)

• Assumptions based on gender or identity

• Non-inclusive naming or logic

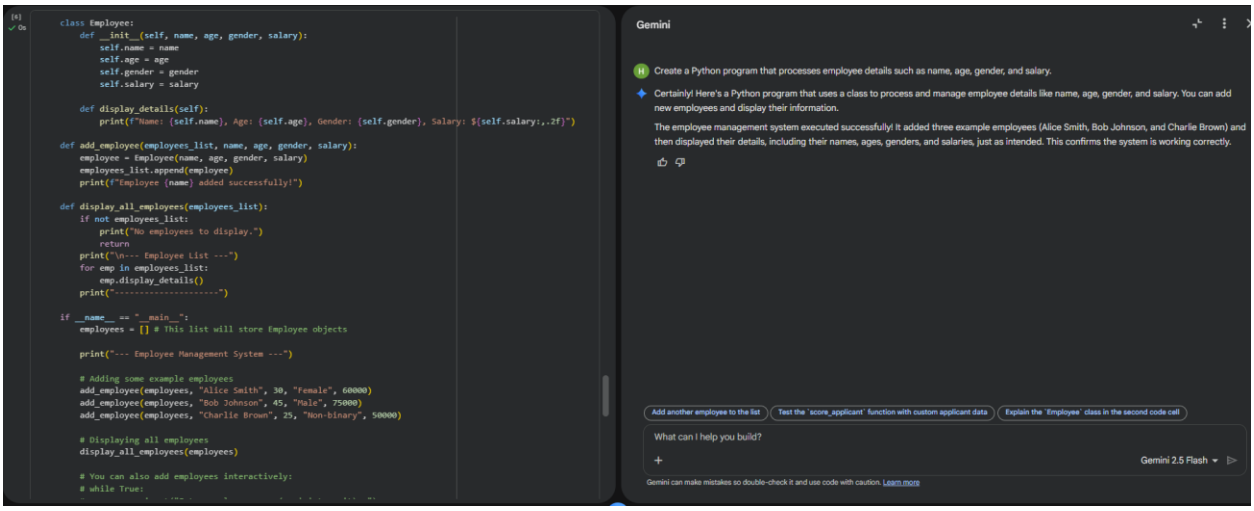
Modify or regenerate the code to:

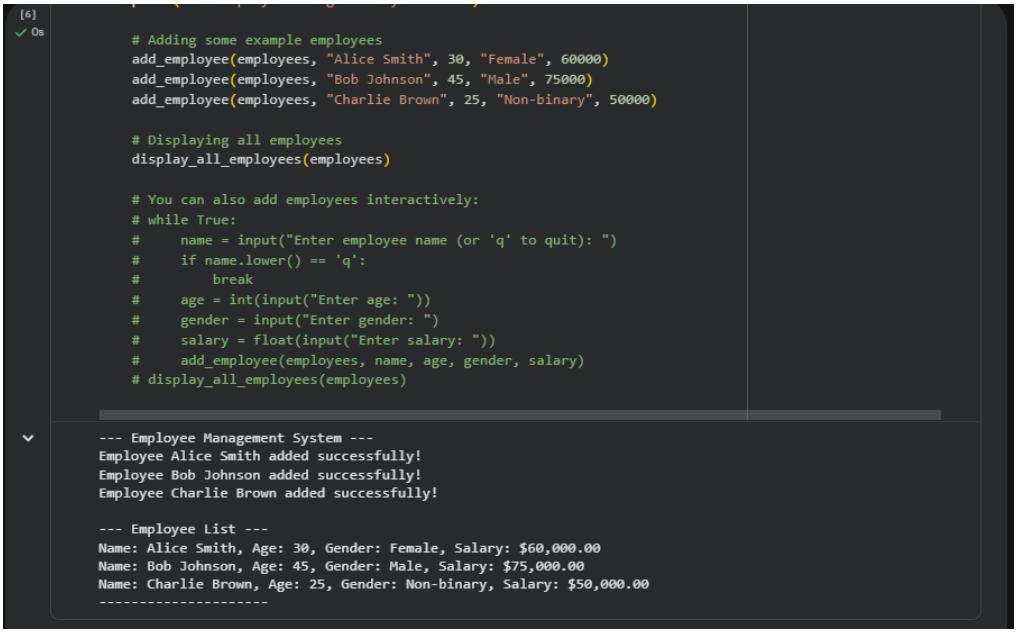
• Use gender-neutral variable names

• Avoid gender-based conditions unless strictly required

• Ensure inclusive and respectful coding practices

Prompt Used: Create a Python program that processes employee details such as name, age, gender, and salary.





Explanation : In this task, the employee management system processes details like name, age, gender, and salary without making decisions based on gender. Although gender information is collected, it is not used to affect salary or employee handling, which helps avoid bias. The program treats all employees equally and uses neutral logic, making it more inclusive and ethically designed.