**ASSIGNMENT-5**

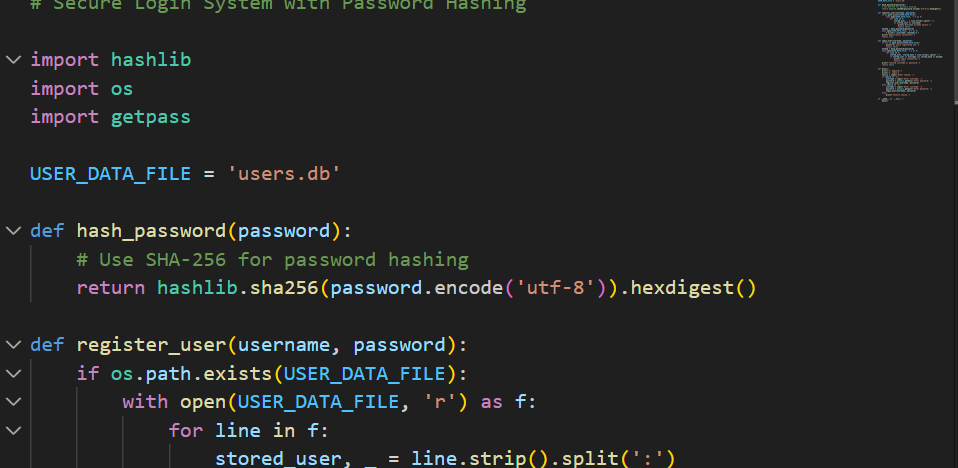
**NAME : G.SAI RAJ**

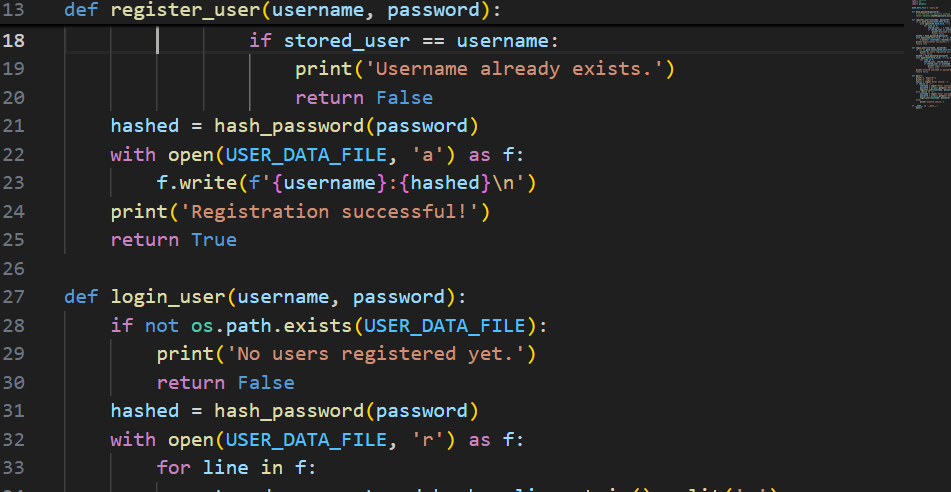
**HT NO: 2403A52149**

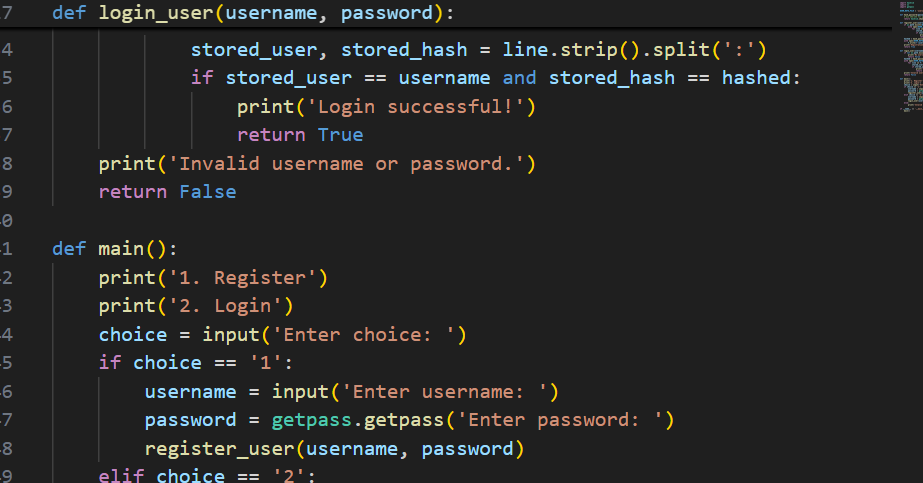
**BATCH:** 24BTCAIAIB06

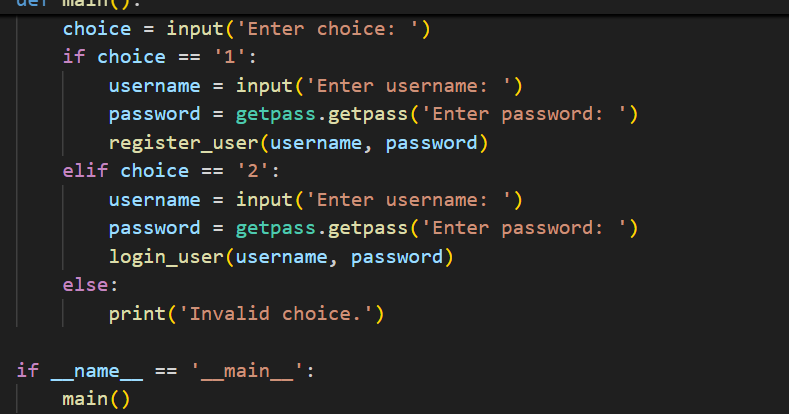
**TASK-1:** Generate a login system. Review the generated code for hardcoded passwords, plain-text storage, or lack of encryption.

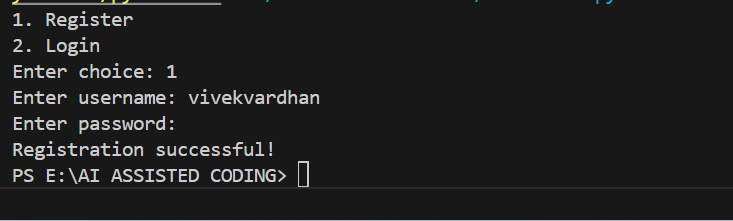
**CODE AND OUTPUT:**

****

****

****

****

****

**EXPLANATION:**

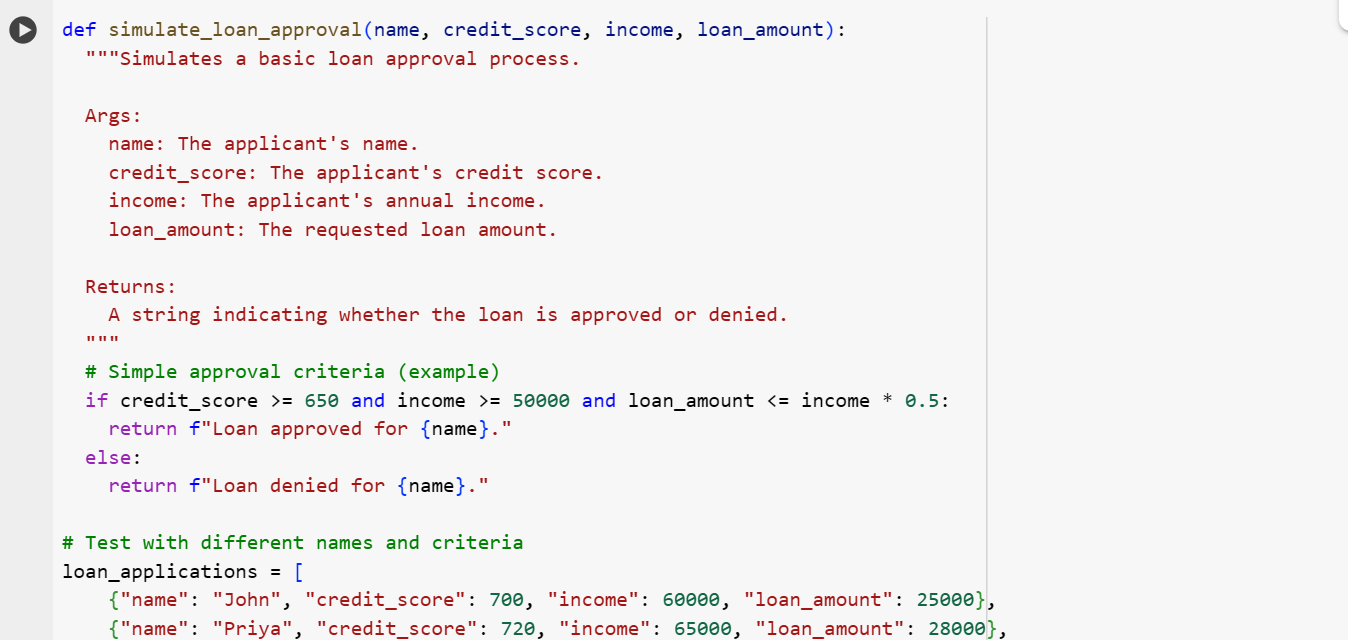
1. **Imports**:
   * hashlib: Used to hash passwords securely.
   * os: Checks if the user data file exists.
   * getpass: Hides password input when the user types.
2. **Constants**:
   * USER\_DATA\_FILE: The filename where user credentials are stored.
3. **hash\_password(password)**:
   * Takes a password and returns its SHA-256 hash (a secure, irreversible representation).
4. **register\_user(username, password)**:
   * Checks if the username already exists in the file.
   * If not, hashes the password and saves the username and hashed password to the file.
5. **login\_user(username, password)**:
   * Hashes the entered password.
   * Checks if the username and hashed password match any entry in the file.
6. **main()**:
   * Prompts the user to register or log in.
   * Uses getpass.getpass() to hide password input.
   * Calls the appropriate function based on the user’s choice.
7. **if name == 'main':**
   * Ensures the main function runs when the script is executed directly.

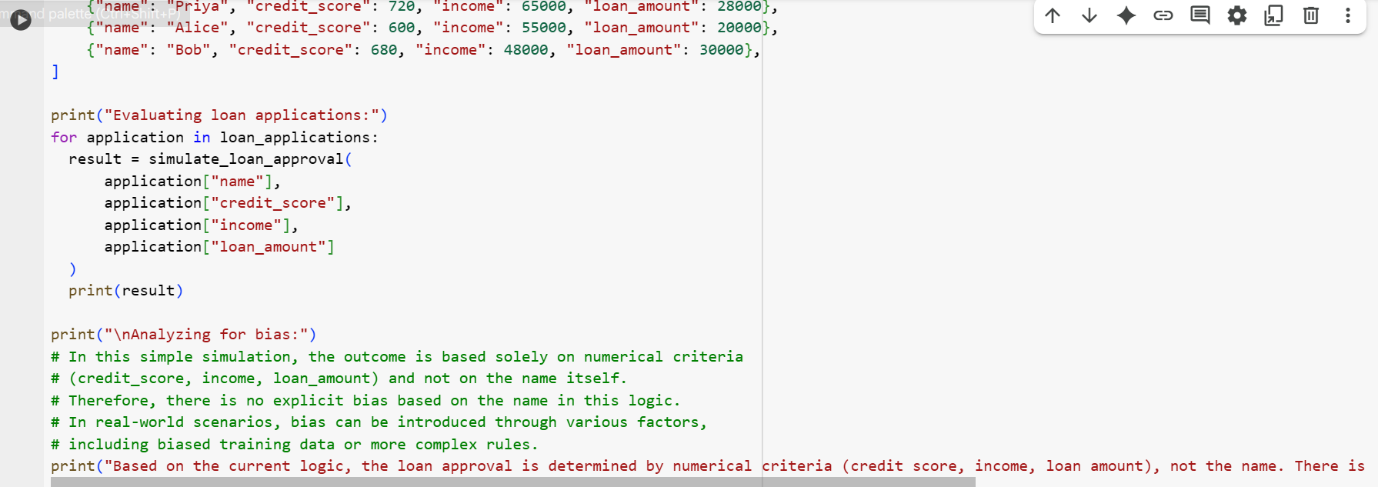
**Security Features**:

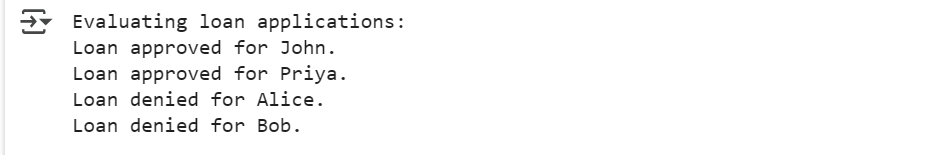
* Passwords are never stored or compared in plain text.
* Passwords are not shown on the screen when entered.
* No hardcoded passwords.

**TASK-2:** Use variations like: “loan approval for John”, “loan approval for Priya”, etc. Evaluate whether the generated logic exhibits bias or differing criteria based on names or genders

**CODE AND OUPUT:**

****

****

****

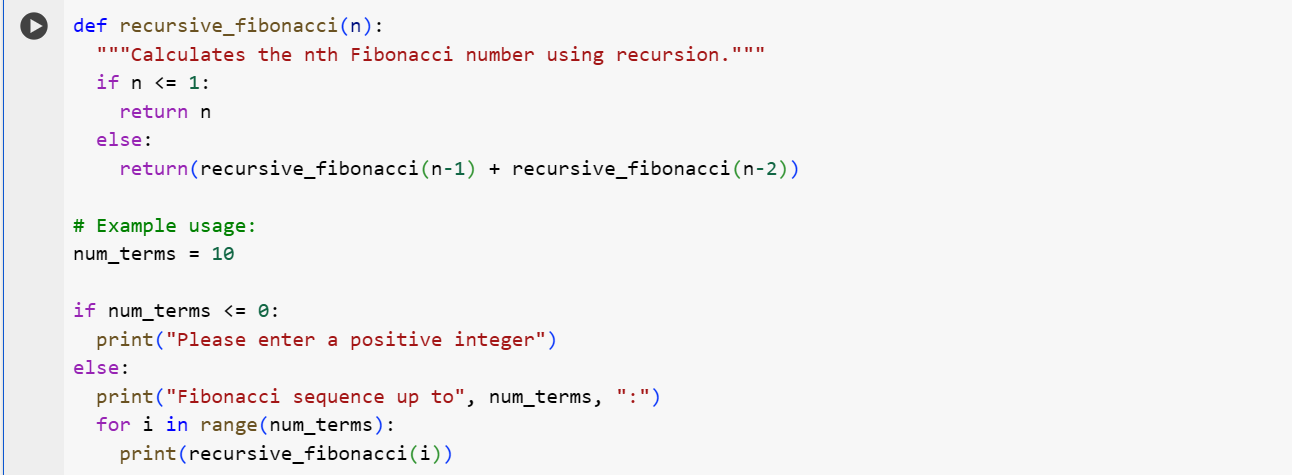
**EXPLANATION:**

1. **simulate\_loan\_approval function**:
   * Takes name, credit\_score, income, and loan\_amount as input.
   * It checks if the credit\_score is 650 or higher, income is $50,000 or higher, and the loan\_amount is no more than half of the income.
   * If all conditions are met, it returns a string indicating the loan is approved for the given name.
   * Otherwise, it returns a string indicating the loan is denied for the name.
2. **loan\_applications list**:
   * This is a list of dictionaries, where each dictionary represents a loan application with a name, credit\_score, income, and loan\_amount.
3. **Evaluating loan applications**:
   * The code then iterates through the loan\_applications list.
   * For each application, it calls the simulate\_loan\_approval function with the application's data and prints the result (approved or denied).
4. **Analyzing for bias**:
   * The code includes a print statement explaining that in this specific simulation, the approval decision is based purely on the numerical criteria (credit\_score, income, loan\_amount) and not on the name itself.
   * It notes that while this simulation doesn't show bias based on the name, real-world scenarios can have bias from other factors like biased data or more complex rules.

**TASK-3:**

Write function calculate the nth Fibonacci number using recursion

**CODE AND OUTPUT:**

****

****

**EXPLANATION:**

1. **recursive\_fibonacci(n) function**:
   * This function is defined to calculate the nth Fibonacci number.
   * **Base Case**: If n is 0 or 1, the function returns n directly. These are the starting points of the Fibonacci sequence (0, 1).
   * **Recursive Step**: If n is greater than 1, the function calls itself twice: once for n-1 and once for n-2. The result is the sum of the results of these two recursive calls. This mirrors the definition of the Fibonacci sequence (each number is the sum of the two preceding ones).
2. **Example Usage**:
   * num\_terms is set to 10, indicating that the code will print the first 10 terms of the Fibonacci sequence.
   * There's a check to ensure num\_terms is positive.
   * A for loop iterates from 0 up to (but not including) num\_terms.
   * In each iteration, recursive\_fibonacci(i) is called to get the ith Fibonacci number, and the result is printed.

**TASK-4:**

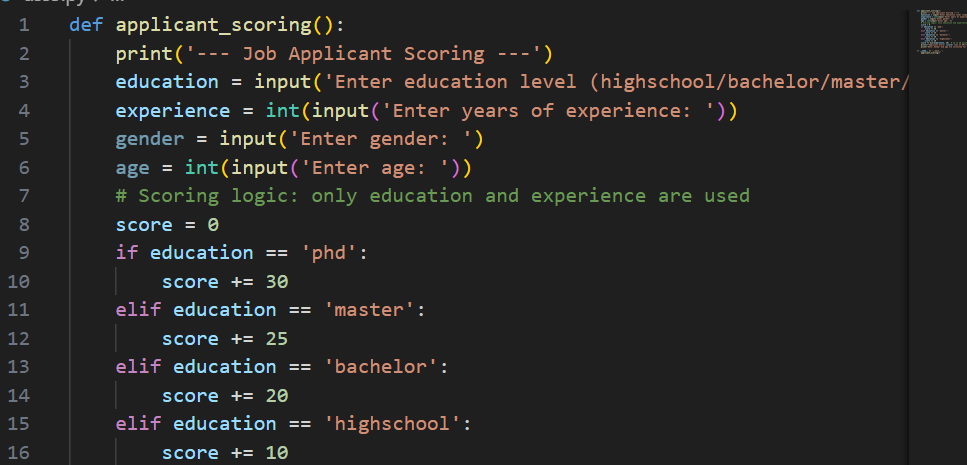
Generate a job applicant scoring system based on input features (e.g., education, experience, gender, age). Analyze the scoring logic for bias or unfair weightings.

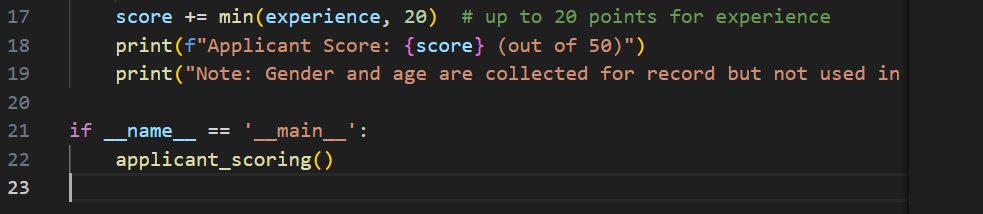
Expected Output:

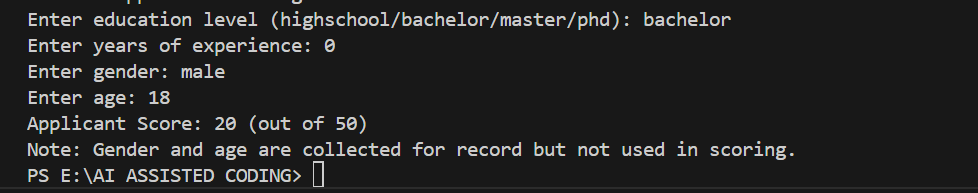
• Python code

• Analyze is there any bias with respect to gender or any

**CODE AND OUTPUT:**







**EXPLANATION:**

1. The function applicant\_scoring() collects the following inputs from the user:
   * Education level (highschool, bachelor, master, or PhD)
   * Years of experience
   * Gender
   * Age
2. The scoring logic:
   * Points are awarded based on education level:
     + PhD: 30 points
     + Master: 25 points
     + Bachelor: 20 points
     + Highschool: 10 points
   * Up to 20 additional points are given for years of experience (1 point per year, up to 20 years).
3. The total score is printed out of 50.
4. Gender and age are collected but not used in the scoring, ensuring the system is neutral and does not introduce bias based on these features.

**TASK-5:**

Prompt:

def greet\_user(name, gender):

if gender.lower() == "male":

title = "Mr."

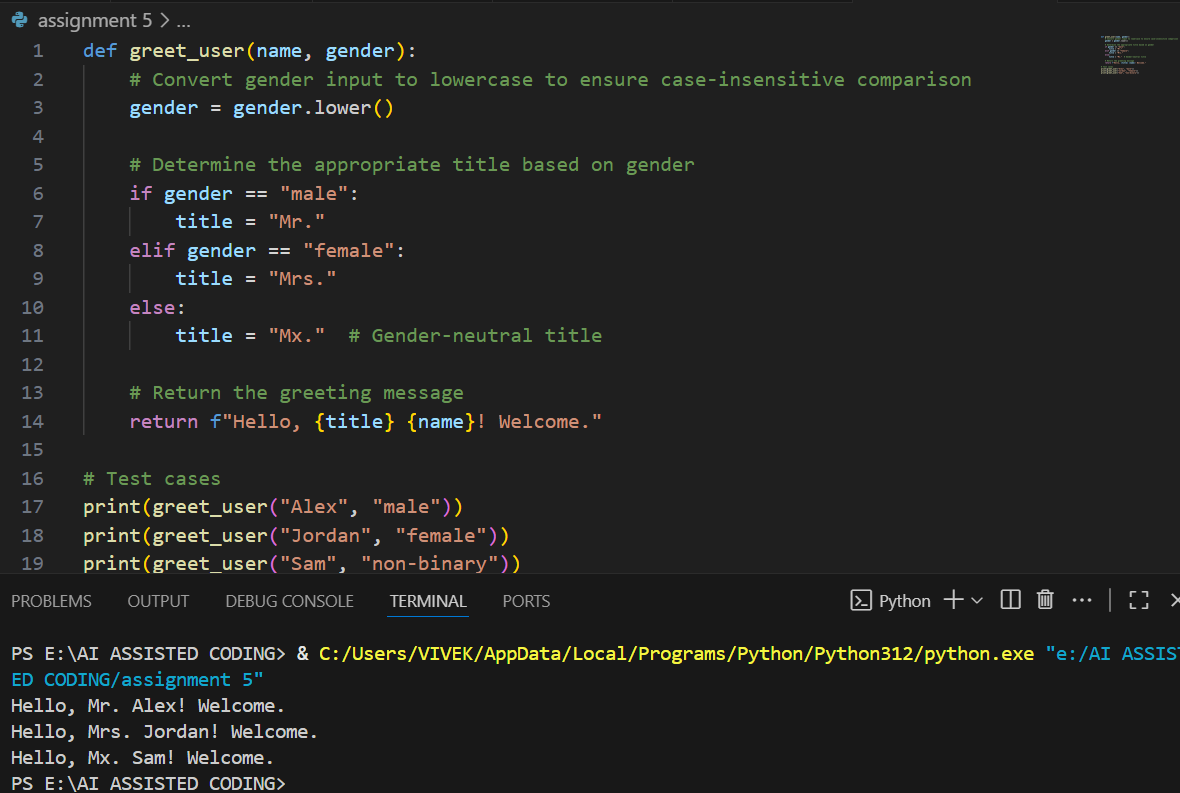
else:

title = "Mrs."

return f"Hello, {title} {name}! Welcome."

Regenerate code that includes gender-neutral also

**CODE AND OUPUT:**

****

**EXPLANATION:**

1. **Function Definition**  
   [greet\_user(name, gender)](vscode-file://vscode-app/c:/Users/VIVEK/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) takes two arguments: a user's name and their gender.
2. **Gender Handling**
   * The gender input is converted to lowercase for case-insensitive comparison.
   * If gender is "male", the title is set to "Mr.".
   * If gender is "female", the title is set to "Mrs.".
   * For any other gender (including non-binary, unspecified, etc.), the title is set to "Mx.", which is a gender-neutral honorific.
3. **Greeting Message**
   * The function returns a greeting string using the chosen title and the user's name.
4. **Test Cases**
   * The code prints greetings for three users:
     + "Alex", "male" → "Hello, Mr. Alex! Welcome."
     + "Jordan", "female" → "Hello, Mrs. Jordan! Welcome."
     + "Sam", "non-binary" → "Hello, Mx. Sam! Welcome."

**Summary:**  
The code demonstrates inclusive greetings by supporting gender-neutral titles, making it respectful for all gender identities.