LAB NAME : AI ASSISTED CODING

LAB NUMBER :02

ROLL NO :2503A51L16

BRANCH : CSE

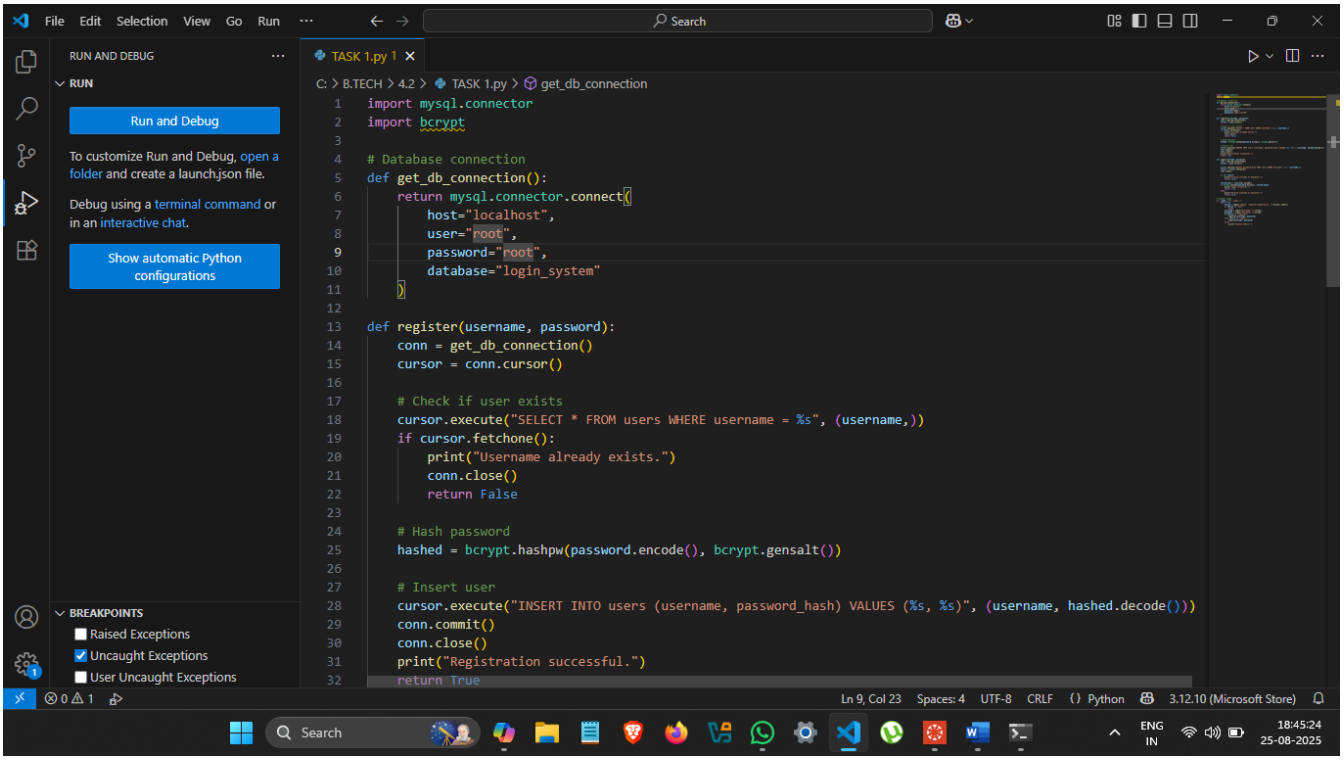
NAME : K.JASHUVA

TASK 1

TASK DESCRIPTION: Use an AI tool (e.g., Copilot, Gemini, Cursor) to generate a login system. Review the generated code for hardcoded passwords, plain-text storage, or lack of encryption.

PROMPT: Generate a secure login system in Python with user registration, hashed password storage, and login verification, then review the code for hardcoded passwords, plain-text storage, or missing encryption.

CODE:-



```python
def register(username, password):
    print("Registration successful.")
    return True

def login(username, password):
    conn = get_db_connection()
    cursor = conn.cursor()

    cursor.execute("SELECT password_hash FROM users WHERE username = %s", (username,))
    result = cursor.fetchone()
    conn.close()

    if not result:
        print("Invalid username or password.")
        return False

    stored_hash = result[0].encode()
    if bcrypt.checkpw(password.encode(), stored_hash):
        print("Login successful.")
        return True
    else:
        print("Invalid username or password.")
        return False

# Example usage
if __name__ == "__main__":
    while True:
        choice = input("Choose: [register/login/exit]: ").strip().lower()
        if choice == "exit":
            break
        username = input("Username: ").strip()
        password = input("Password: ").strip()
```

```python
def login(username, password):
    if not result:
        print("Invalid username or password.")
        return False

    stored_hash = result[0].encode()
    if bcrypt.checkpw(password.encode(), stored_hash):
        print("Login successful.")
        return True
    else:
        print("Invalid username or password.")
        return False

# Example usage
if __name__ == "__main__":
    while True:
        choice = input("Choose: [register/login/exit]: ").strip().lower()
        if choice == "exit":
            break
        username = input("Username: ").strip()
        password = input("Password: ").strip()
        if choice == "register":
            register(username, password)
        elif choice == "login":
            login(username, password)
        else:
            print("Invalid choice.")
```

OUTPUT:

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS                                   Python Debug Console

PS C:\B.TECH\4.2>  c:; cd 'c:\B.TECH\4.2'; & 'c:\Users\kamer\AppData\Local\Microsoft\WindowsApps\python3.12.exe' 'c:\Users\kamer\.vscode\extension
s\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '62613' '--' 'C:\B.TECH\4.2\TASK 1.py'
Choose: [register/login/exit]: register
Username: Shashi
Password: Shashi@123
Registration successful.
Choose: [register/login/exit]: login
Username: Shashi
Password: Shashi@123
Login successful.
Choose: [register/login/exit]: login
Username: Jashuva
Password: Jashu@123
Login successful.
Choose: [register/login/exit]:

Ln 63, Col 41   Spaces: 4   UTF-8   CRLF   {} Python   3.12.10 (Microsoft Store)
```

TASK 2

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

PROMPT: Generate a loan approval system, test it with names like John, Priya, Ahmed, and Maria, check if the logic is biased by name or gender

CODE:

```python
def approve_loan(applicant):
    # Criteria thresholds
    min_income = 30000
    min_credit_score = 650
    max_debt_to_income_ratio = 0.4

    income = applicant["income"]
    credit_score = applicant["credit_score"]
    debt = applicant["debt"]

    # Calculate debt-to-income ratio
    dti = debt / income if income > 0 else 1

    # Decision logic
    if income >= min_income and credit_score >= min_credit_score and dti <= max_debt_to_income_ratio:
        return f"Loan approved for {applicant['name']}."
    else:
        return f"Loan denied for {applicant['name']}."

# Test applicants
applicants = [
    {"name": "John", "income": 50000, "credit_score": 700, "debt": 10000},
    {"name": "Priya", "income": 45000, "credit_score": 680, "debt": 12000},
    {"name": "Ahmed", "income": 28000, "credit_score": 720, "debt": 5000},
    {"name": "Maria", "income": 60000, "credit_score": 640, "debt": 15000}
]

# Run tests
for person in applicants:
    print(approve_loan(person))
```

OUTPUT:

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS                                   Python Debug Console

PS C:\B.TECH\4.2>  c:; cd 'c:\B.TECH\4.2'; & 'c:\Users\kamer\AppData\Local\Microsoft\WindowsApps\python3.12.exe' 'c:\Users\kamer\.vscode\extension
s\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63415' '--' 'C:\B.TECH\4.2\TASK 2.py'
Loan approved for John.
Loan approved for Priya.
Loan denied for Ahmed.
Loan denied for Maria.
PS C:\B.TECH\4.2>

Ln 30, Col 32   Spaces: 4   UTF-8   {} Python   3.12.10 (Microsoft Store)
```

TASK 3

TASK DESCRIPTION: Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document

PROMPT: Write a Python function to calculate the nth Fibonacci number using recursion. Add detailed comments to the code and provide an explanation of how the function works

CODE:-

```python
def fibonacci(n):
    """
    Recursively calculates the nth Fibonacci number.

    Parameters:
    n (int): The position in the Fibonacci sequence (0-indexed).

    Returns:
    int: The nth Fibonacci number.
    """
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

# 🚀 Take input from the user
try:
    user_input = int(input("Enter a non-negative integer to find its Fibonacci number: "))

    if user_input < 0:
        print("Please enter a non-negative integer only.")
    else:
        result = fibonacci(user_input)
        print(f"The {user_input}th Fibonacci number is: {result}")

except ValueError:
    print("Invalid input! Please enter a valid integer.")
```

OUTPUT:

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS                    Python Debug Console

PS C:\B.TECH\4.2>  c:; cd 'c:\B.TECH\4.2'; & 'c:\Users\kamer\AppData\Local\Microsoft\WindowsApps\python3.12.exe' 'c:\Users\kamer\.vscode\extension
s\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63833' '--' 'C:\B.TECH\4.2\TASK 3.py'
Enter a non-negative integer to find its Fibonacci number: 16
The 16th Fibonacci number is: 987
PS C:\B.TECH\4.2>

Ln 29, Col 58   Spaces: 4   UTF-8   CRLF   {} Python   3.12.10 (Microsoft Store)
```

TASK 4

TASK DESCRIPTION: Ask to generate a job applicant scoring system based on input features (e.g., education, experience, gender, age). Analyse the scoring logic for bias or unfair weightings

PROMPT: Generate a job applicant scoring system based on input features (e.g., education, experience, gender, age). Analyse the scoring logic for bias or unfair weighting, Analyze is there any bias with respect to gender or any.

CODE:

```python
def score_applicant(education, experience, gender, age):

    score = 0

    # 🎓 Education weighting
    education_weights = {
        'HighSchool': 10,
        'Diploma': 20,
        'BTech': 30,
        'MTech': 40,
        'PhD': 50
    }
    score += education_weights.get(education, 0)

    # 💼 Experience weighting
    score += min(experience * 2, 30)  # Max 30 points for experience

    # ⚠️ Gender weighting (problematic!)
    gender_weights = {
        'Male': 10,
        'Female': 10,
        'Other': 10
    }
    score += gender_weights.get(gender, 0)

    # ⏳ Age weighting (potentially biased)
    if 25 <= age <= 35:
        score += 10  # Favoring mid-career age
    elif 18 <= age < 25:
        score += 5   # Entry-level age
    elif age > 35:
        score += 5   # Senior age
```

```python
        score += 5   # Senior age
    else:
        score += 0   # Unusual age range

    return round(score, 2)

def score_applicant(education, experience, gender, age):
    education_weights = {
        'HighSchool': 10,
        'Diploma': 20,
        'BTech': 30,
        'MTech': 40,
        'PhD': 50
    }
    score = education_weights.get(education, 0)
    score += min(experience * 2, 30)

    gender_weights = {
        'Male': 10,
        'Female': 10,
        'Other': 10
    }
    score += gender_weights.get(gender, 0)

    if 25 <= age <= 35:
        score += 10
    elif 18 <= age < 25 or age > 35:
        score += 5

    return round(score, 2)

# ✅ List of applicants
```

```python
def score_applicant(education, experience, gender, age):
    return round(score, 2)

# ✅ List of applicants
applicants = [
    {'education': 'Diploma', 'experience': 2, 'gender': 'Female', 'age': 22},
    {'education': 'BTech', 'experience': 4, 'gender': 'Male', 'age': 28},
    {'education': 'MTech', 'experience': 1, 'gender': 'Other', 'age': 36}
]

# ✅ Loop through applicants and score each one
for i, applicant in enumerate(applicants, start=1):
    score = score_applicant(
        education=applicant['education'],
        experience=applicant['experience'],
        gender=applicant['gender'],
        age=applicant['age']
    )
    print(f"Applicant {i} Score: {score}")

for i, applicant in enumerate(applicants, start=1):
    score = score_applicant(**applicant)
    print(f"Applicant {i} Score: {score}")
```

OUTPUT:

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS                    Python Debug Console

PS C:\B.TECH\4.2>  c:; cd 'c:\B.TECH\4.2'; & 'c:\Users\kamer\AppData\Local\Microsoft\WindowsApps\python3.12.exe' 'c:\Users\kamer\.vscode\extension
s\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '64226' '--' 'C:\B.TECH\4.2\TASK 4.py'
Applicant 1 Score: 39
Applicant 2 Score: 58
Applicant 3 Score: 57
Applicant 1 Score: 39
Applicant 2 Score: 58
Applicant 3 Score: 57
PS C:\B.TECH\4.2>

Ln 78, Col 43   Spaces: 4   UTF-8   CRLF   {} Python   3.12.10 (Microsoft Store)
```

TASK 5

Code Snippet:

```python
def greet_user(name, gender):
    if gender.lower() == "male":
        title = "Mr."
    else:
        title = "Mrs."
    return f"Hello, {title} {name}! Welcome."
```

CODE:

```python
def greet_user(name, gender):
    # Normalize gender input to lowercase
    gender = gender.lower()

    # Assign title based on gender
    if gender == "male":
        title = "Mr."
    elif gender == "female":
        title = "Mrs."
    else:
        title = "Mx."  # Gender-neutral title

    return f"Hello, {title} {name}! Welcome."
# Define the function
def greet_user(name, gender):
    gender = gender.lower()
    if gender == "male":
        title = "Mr."
    elif gender == "female":
        title = "Mrs."
    else:
        title = "Mx."  # Gender-neutral title
    return f"Hello, {title} {name}! Welcome."

# Call the function with sample input
output = greet_user("Jashuva", "Other")

# Print the result
print(output)
```

OUTPUT:

```
PS C:\B.TECH\4.2>  c:; cd 'c:\B.TECH\4.2'; & 'c:\Users\kamer\AppData\Local\Microsoft\WindowsApps\python3.12.exe' 'c:\Users\kamer\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '64631' '--' 'C:\B.TECH\4.2\TASK 5.py'
Hello, Mx. Jashuva! Welcome.
PS C:\B.TECH\4.2>
```

OBSERVATION : I observed that GitHub copilot can quickly generate working code for tasks such as login systems, loan approvals, Fibonacci functions, and job applicant scoring. However, the generated code sometimes contains issues like hardcoded values, lack of encryption, or biased decision logic. This shows that AI tools are helpful for faster coding but require human review for security, fairness, and correctness. GitHub Copilot is a fascinating tool to observe—especially in how it transforms the developer experience. Here's a breakdown of key observations across its functionality, impact, and adoption