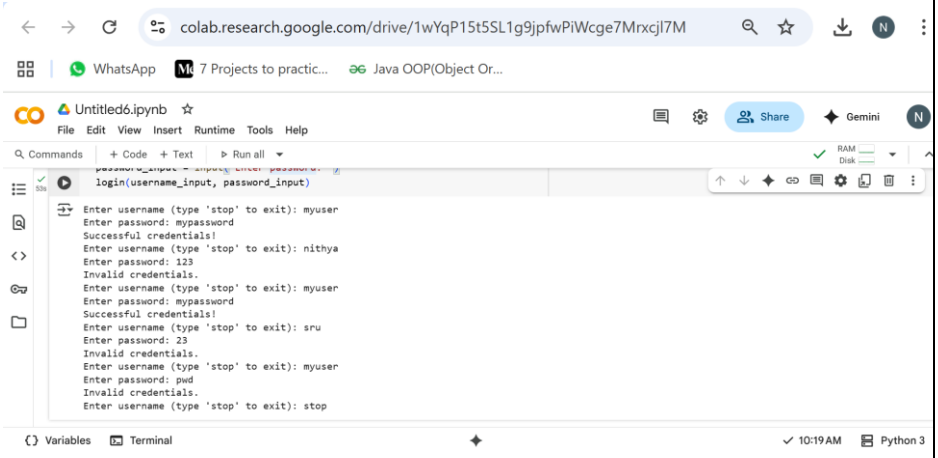
SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName:B. Tech		**Assignment Type: Lab**	**AcademicYear:**2025-2026
CourseCoordinatorName	Venkataramana Veeramsetty		
Instructor(s)Name	Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)		
CourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week3 - Tuesday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	
AssignmentNumber:5.2(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	Lab 5: Ethical Foundations – Responsible AI Coding Practices **Lab Objectives:** • To explore the ethical risks associated with AI-generated code. • To recognize issues related to security, bias, transparency, and	Week3 - Wednesday

	copyright. • To reflect on the responsibilities of developers when using AI tools in software development. • To promote awareness of best practices for responsible and ethical AI coding. **Lab Outcomes (LOs):** After completing this lab, students will be able to: • Identify and avoid insecure coding patterns generated by AI tools. • Detect and analyze potential bias or discriminatory logic in AI-generated outputs. • Evaluate originality and licensing concerns in reused AI-generated code. • Understand the importance of explainability and transparency in AI-assisted programming. • Reflect on accountability and the human role in ethical AI coding practices.. **Task Description#1 (Privacy and Data Security)** • 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. **Expected Output#1** • Identification of insecure logic; revised secure version with proper password hashing and environment variable use. **PROMPT1:** write a python code to generate a login system that should review for hardcorded passwords,plain-text storage or lack of encryption **PROMPT2:**I will write username and password in the execution. **PROMPT3:**if I given the same username and password which I already save in the code then it should show the successful credentials if not then print invalid credentials	

OUTPUT:



OBSERVATION:

What I observed in the above python code is we will write inputs of username and password after execution if the stored username and stored password are same then we get output successful credentials Or else we get invalid credentials in the output .

Task Description#2 (Bias)

- 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.

Expected Output#2

- Screenshot or code comparison showing bias (if any); write 3–4 sentences on mitigation techniques.

PROMPT1:write a python for testing bias take the loan for “loan approval for John”, “loan approval for Priya”.take input of name ,age ,income ,creditscore of john and priya give inputs after execution only.

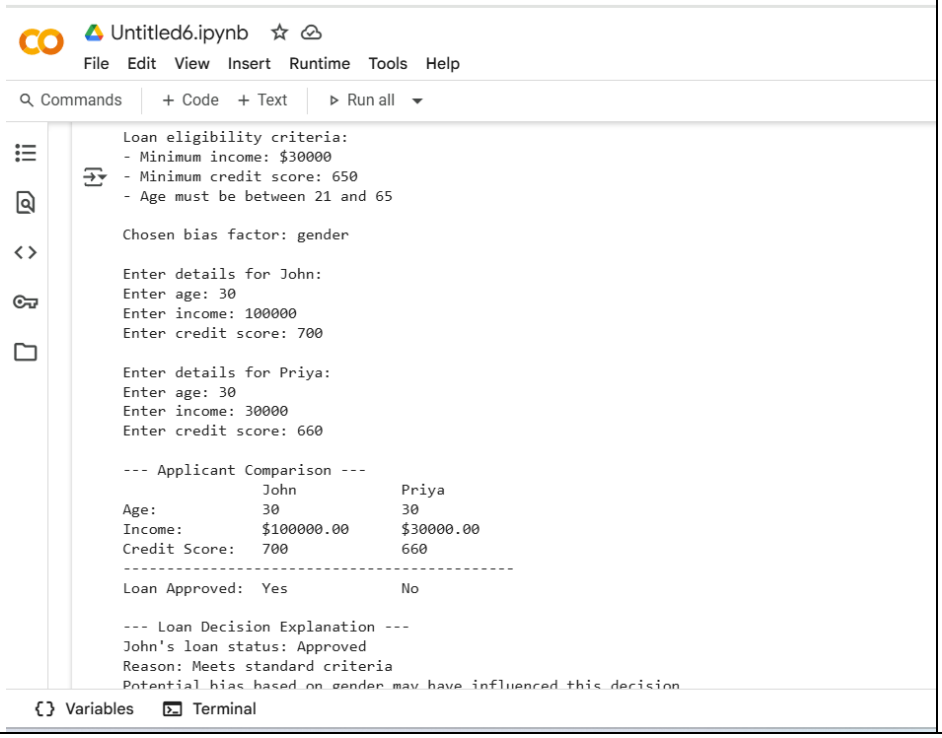
PROMPT2:compare them side by side. but give loan any one check the differnce between gender or income that your wish .

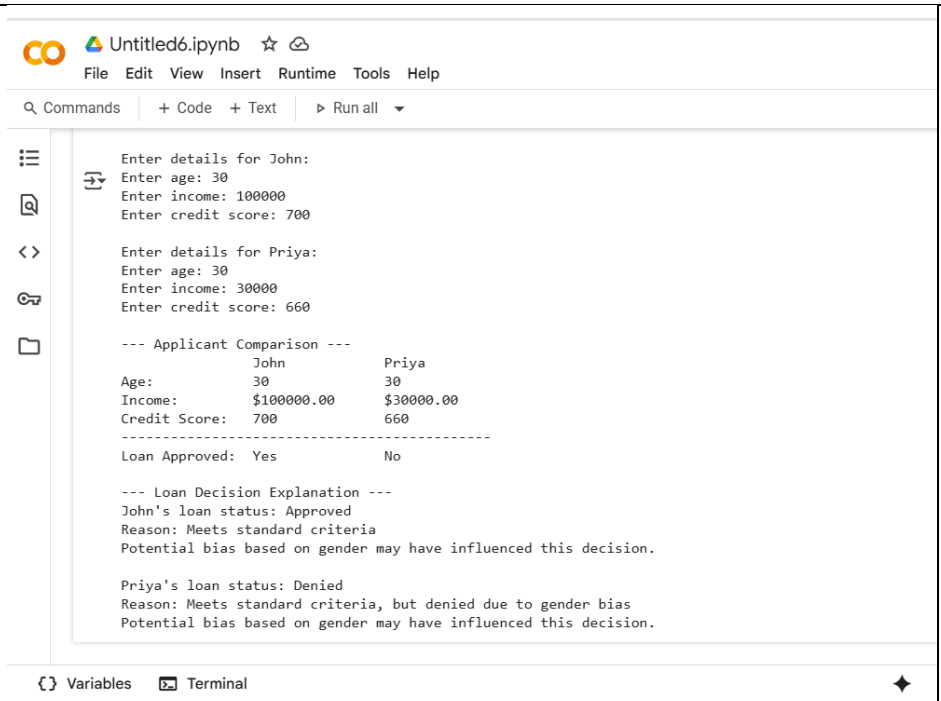
PROMPT3: In output it should be meniton. why u given loan and why

you not given loan?


colab.research.google.com/drive/1wYqP15t5SL1g9jpfwPiWcge7MrxcjI7M#sc...
WhatsApp
7 Projects to practic...
Java OOP(Object Or...
Untitled6.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Share
Gemini
priya_approved, priya_reason = approve_loan("Priya", priya_age, priya_income, priya_credit_score, BIAS_FACTOR)
# 5. Compare inputs and decisions
print("\n--- Applicant Comparison ---")
print(f"{'':<15} {'John':<15} {'Priya':<15}")
print(f"{'Age:':<15} {john_age:<15} {priya_age:<15}")
print(f"{'Income:':<15} ${john_income:<14.2f} ${priya_income:<14.2f}")
print(f"{'Credit Score:':<15} {john_credit_score:<15} {priya_credit_score:<15}")
print("-" * 45)
print(f"{'Loan Approved:':<15} {'Yes' if john_approved else 'No':<15} {'Yes' if priya_approved else 'No':<15}")
# 6. Explain the decisions and potential bias
print("\n--- Loan Decision Explanation ---")
print(f"John's loan status: {'Approved' if john_approved else 'Denied'}")
print(f"Reason: {john_reason}")
if BIAS_FACTOR == 'gender':
print(f"Potential bias based on {BIAS_FACTOR} may have influenced this decision.")
elif BIAS_FACTOR == 'income':
print(f"Potential bias based on {BIAS_FACTOR} may have influenced this decision.")
print(f"\nPriya's loan status: {'Approved' if priya_approved else 'Denied'}")
print(f"Reason: {priya_reason}")
if BIAS_FACTOR == 'gender':
print(f"Potential bias based on {BIAS_FACTOR} may have influenced this decision.")
elif BIAS_FACTOR == 'income':
print(f"Potential bias based on {BIAS_FACTOR} may have influenced this decision.")
Variables
Terminal
10:39 AM
Python 3

OUTPUT:


Untitled6.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Loan eligibility criteria:
- Minimum income: $30000
- Minimum credit score: 650
- Age must be between 21 and 65
Chosen bias factor: gender
Enter details for John:
Enter age: 30
Enter income: 100000
Enter credit score: 700
Enter details for Priya:
Enter age: 30
Enter income: 30000
Enter credit score: 660
--- Applicant Comparison ---
John Priya
Age: 30 30
Income: $100000.00 $30000.00
Credit Score: 700 660
Loan Approved: Yes No
--- Loan Decision Explanation ---
John's loan status: Approved
Reason: Meets standard criteria
Potential bias based on gender may have influenced this decision
Variables
Terminal



OBSERVATION:

In the above code I given the inputs of age ,income and credicts of john and Priya.Based on their bias Like gender,income or anything checks the difference and gives the code.if there is a chance of getiing loan of 2 memebers then in output its shows the explaination of "YES" on the 2 memebers.

Task Description#3 (Transparency)

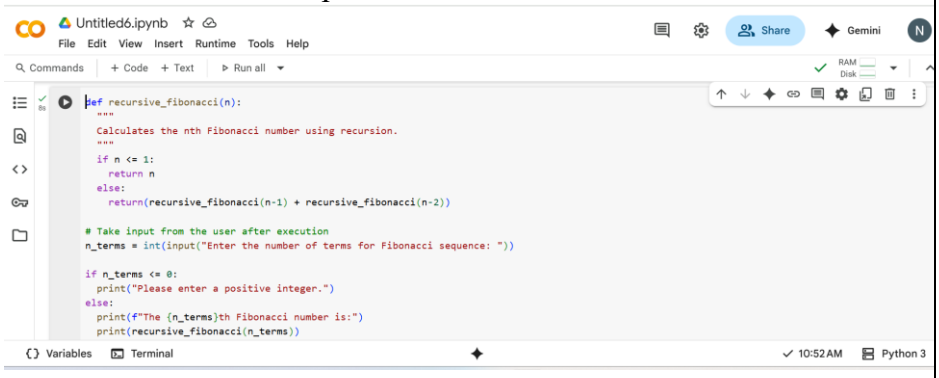
- Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document

Expected Output#3

- Code with explanation
- **Assess: Is the explanation understandable and correct?**

PROMPT1:write a python code using function to calculate the nth Fibonacci numbers using recursion

PROMPT2: Take the input after the execution



OUTPUT:

```
Enter the number of terms for Fibonacci sequence: 6
The 6th Fibonacci number is:
8
```

OBSERVATION:

In the above code it finds the given number of the digit like the Fibonacci series starts from 0 so if we write what is the 1st term its give 0 like that the output we get.

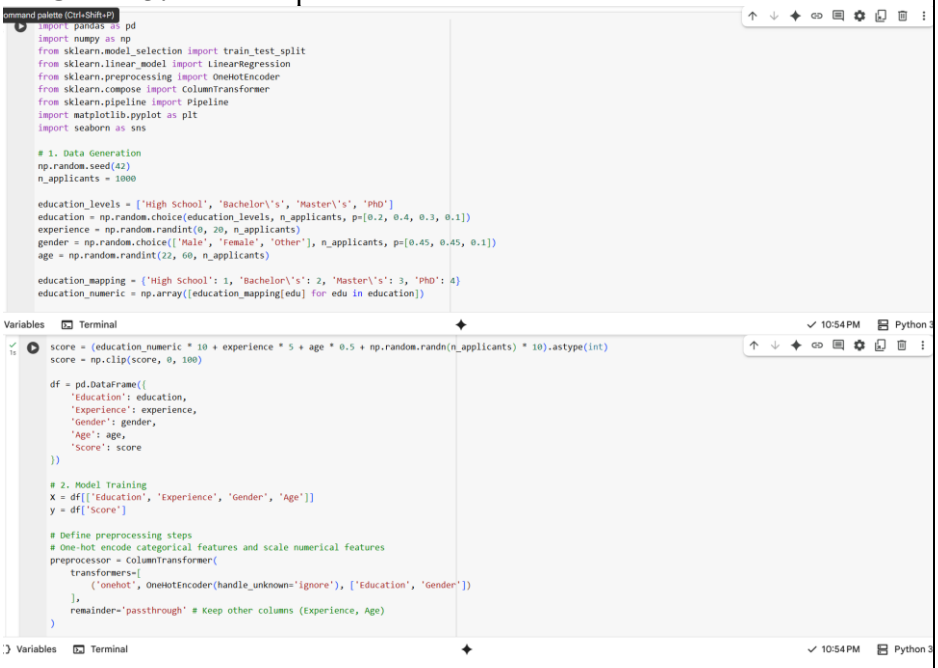
Task Description#4 (Bias)

- Ask to 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#4

- Python code
- Analyze is there any bias with respect to gender or any

PROMPT1: write a python code to generate a job applicant scoring system based on input features like education,experience,gender,age.
PROMPT2: Analyze is there any bias with respect to gender or any
PROMPT3:take all inputs after the execution.



```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Data Generation
np.random.seed(42)
n_applicants = 1000

education_levels = ['High School', 'Bachelor\'s', 'Master\'s', 'PhD']
education = np.random.choice(education_levels, n_applicants, p=[0.2, 0.4, 0.3, 0.1])
experience = np.random.randint(0, 20, n_applicants)
gender = np.random.choice(['Male', 'Female', 'Other'], n_applicants, p=[0.45, 0.45, 0.1])
age = np.random.randint(22, 60, n_applicants)

education_mapping = {'High School': 1, 'Bachelor\'s': 2, 'Master\'s': 3, 'PhD': 4}
education_numeric = np.array([education_mapping[edu] for edu in education])
```

Variables Terminal 10:54 PM Python 3

```
score = (education_numeric * 10 + experience * 5 + age * 0.5 + np.random.randn(n_applicants) * 10).astype(int)
score = np.clip(score, 0, 100)

df = pd.DataFrame({
    'Education': education,
    'Experience': experience,
    'Gender': gender,
    'Age': age,
    'Score': score
})

# 2. Model Training
X = df[['Education', 'Experience', 'Gender', 'Age']]
y = df['Score']

# Define preprocessing steps
# One-hot encode categorical features and scale numerical features
preprocessor = ColumnTransformer(
    transformers=[
        ('onehot', OneHotEncoder(handle_unknown='ignore'), ['Education', 'Gender'])
    ],
    remainder='passthrough' # Keep other columns (Experience, Age)
)
```

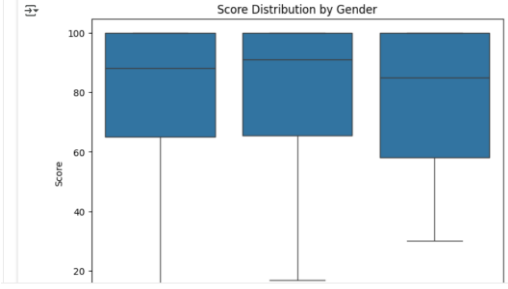
Variables Terminal 10:54 PM Python 3

```
# Create a pipeline with preprocessor and regression model
model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('regressor', LinearRegression())])

# Train the model on the entire dataset
model.fit(X, y)

# 3. Applicant Scoring
# You can take inputs here for new applicants
# For demonstration, let's create a sample new applicant
new_applicant_data = {
    'Education': ['Master\'s'],
    'Experience': [7],
    'Gender': ['Female'],
    'Age': [30]
}
new_applicant_df = pd.DataFrame(new_applicant_data)

predicted_score = model.predict(new_applicant_df)
print(f"Predicted score for the new applicant: {predicted_score[0]:.2f}")

# To take multiple inputs after execution, you could loop and ask for input:
# new_applicants_list = []
# while True:
```

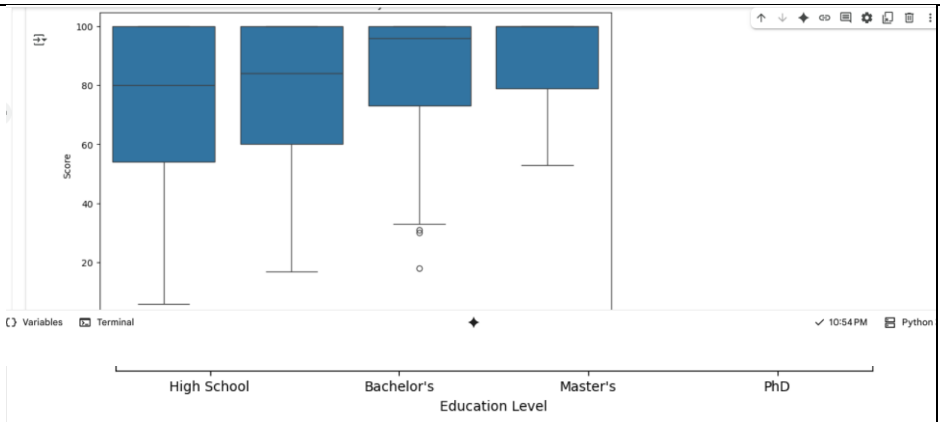
```
# while True:
#     education_input = input("Enter Education (High School, Bachelor's, Master's, PhD) or 'done' to finish: ")
#     if education_input.lower() == 'done':
#         break
#     experience_input = int(input("Enter Experience: "))
#     gender_input = input("Enter Gender (Male, Female, Other): ")
#     age_input = int(input("Enter Age: "))
#     new_applicants_list.append({'Education': education_input, 'Experience': experience_input, 'Gender': gender_input, 'Age': age_input})

# if new_applicants_list:
#     new_applicants_df = pd.DataFrame(new_applicants_list)
#     predicted_scores = model.predict(new_applicants_df)
#     print("\nPredicted scores for entered applicants:")
#     for i, score in enumerate(predicted_scores):
#         print(f"Applicant {i+1}: {score:.2f}")


# 4. Bias Analysis
# Analyze score distribution by Gender
plt.figure(figsize=(8, 6))
sns.boxplot(x='Gender', y='Score', data=df)
plt.title('Score Distribution by Gender')
plt.ylabel('Score')
plt.xlabel('Gender')
```

```
# 4. Bias Analysis
# Analyze score distribution by Gender
plt.figure(figsize=(8, 6))
sns.boxplot(x='Gender', y='Score', data=df)
plt.title('Score Distribution by Gender')
plt.ylabel('Score')
plt.xlabel('Gender')
plt.show()

# Analyze score distribution by Education
plt.figure(figsize=(10, 6))
sns.boxplot(x='Education', y='Score', data=df, order=education_levels)
plt.title('Score Distribution by Education')
plt.ylabel('Score')
plt.xlabel('Education Level')
plt.show()

# You can perform further analysis, like statistical tests or comparing average scores

# 5. Finish task
print("\nAnalysis Complete.")
print("Bias analysis by gender and education level has been performed visually using box plots.")
print("Further statistical analysis can be performed to quantify potential biases.")
```

```
Predicted score for the new applicant: 74.82
```



Variables Terminal 10:54 PM Python



Observation:
The code involves generating a synthetic dataset of job applicants and training a linear regression model to predict applicant scores based on education, experience, age, and gender. One-hot encoding was used for categorical data, and visualizations like box plots were created to observe score distribution.

Task Description#5 (Inclusiveness)

- Code Snippet

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

Expected Output#5

- Regenerate code that includes **gender-neutral** also

Prompt:
Write a Python program that greets the user based on their gender male/female/neutral.

```python
# 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."


# Test the function with examples
print(greet_user("Alex", "male"))     # Output: Hello, Mr. Alex! Welcome.
print(greet_user("Taylor", "female")) # Output: Hello, Mrs. Taylor! Welcome.
print(greet_user("Jordan", "non-binary"))  # Output: Hello, Mx. Jordan! Welcome.
print(greet_user("Sam", "Other"))     # Output: Hello, Mx. Sam! Welcome.
```

```
Hello, Mr. Alex! Welcome.
Hello, Mrs. Taylor! Welcome.
Hello, Mx. Jordan! Welcome.
Hello, Mx. Sam! Welcome.
```

Observation:

The updated function is designed to greet users respectfully based on their gender. It assigns the title "Mr." for males, "Mrs." for females, and "Mx." for users who are gender-neutral or prefer not to specify their gender. This makes the function more inclusive and considerate of diverse gender identities. It also handles different ways of typing gender by converting the input to lowercase, so it works correctly even if the user enters "MALE", "female", or "Other". Overall, the function ensures a personalized and respectful greeting for everyone.

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria	Max Marks
Transparency	0.5
Bias	1.0
Inclusiveness	0.5
Data security and Privacy	0.5
Total	**2.5 Marks**