Republic of the Philippines
Laguna State Polytechnic University
Main Campus, Sta. Cruz, Laguna

ADVANCED PRESENTATION AND REASONING
Machine Problem 1

Name: Ramos, Jezreel R
Year/Section: BSCS3A **Instructor: Mark Bernardino**

Assessment Task: Applying Propositional Logic in Real-World Scenarios through a Mini Expert System

At the end of this activity, students should be able to:

1. Translate real-world conditions into propositional logic expressions.
2. Apply logical implication (P → Q) to decision-making scenarios.
3. Develop a Python program that implements logic rules.
4. Record and analyze system results using CSV as a simple database.
5. Communicate findings through a short written report.

Instructions for Students

1. Download or recreate the Mini Expert System provided in class.
2. Run the program and test at least 3 different students with different conditions for:
   - o Attendance Rule
   - Grading Rule
   - Login System Rule
   - Bonus Points Rule
3. Verify that all results are logged in the CSV file (logic_results.csv).
4. Extend the program by adding one new rule of your own.

Examples:

- Library borrowing (If ID is valid → Allowed to borrow books).
- Enrollment clearance (If fees are paid → Enrollment confirmed).
- Laboratory access (If safety gear is worn → Access granted).

Python Code:

```
"""
Enhanced Mini Expert System: University Logic Rules
With CSV Logging for Record Keeping
"""

import csv
from datetime import datetime
```

```python
# ---------------- Logic Functions ---------------- #
def impl(P, Q):
    return (not P) or Q  # Implication (P -> Q)

def tf(b: bool) -> str:
    return "T" if b else "F"

# -------------------- Logger -------------------- #
def log_result(student_name, rule_name, result):
     with open("logic_results.csv", "a", newline="", encoding="utf-8") as
file:
        writer = csv.writer(file)
        writer.writerow([
            datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            student_name, rule_name, result
        ])

# ---------- Rule 1: Attendance ---------- #
def attendance_rule(student_name):
    print("\n--- Attendance Rule Checker ---")
    late = input("Is the student late? (T/F): ").strip().upper() == "T"
          excuse  =  input("Did  you  bring  an  excuse  letter?  (T/F):
").strip().upper() == "T"

    result = impl(late, excuse)
    outcome = "Satisfied ✅" if result else "Violated ❌"

    print(f"P = {tf(late)} (Late), Q = {tf(excuse)} (Excuse Letter)")
    print("Result:", outcome)

    log_result(student_name, "Attendance Rule", outcome)

# ---------- Rule 2: Grading ---------- #
def grading_rule(student_name):
    print("\n--- Grading Rule Checker ---")
    try:
        grade = float(input("Enter Student Grade: "))
    except ValueError:
        print("Invalid Grade Input")
```

```
        return

    P = grade >= 75   # Condition: passing grade
    Q = grade >= 75   # Conclusion: student passes

    result = impl(P, Q)
    outcome = "Satisfied ✅" if result else "Violated ❌"

    print(f"P = {tf(P)} (grade >= 75), Q = {tf(Q)} (Student Passes)")
    print("Result:", outcome)

    log_result(student_name, "Grading Rule", outcome)

# ---------- Rule 3: Login System ---------- #
def login_rule(student_name):
    print("\n--- Login Rule Checker ---")
    correct_password = "admin123"
    attempt = input("Enter Password: ")

    P = (attempt == correct_password)  # Password Correct
    Q = (attempt == correct_password)  # Access granted if correct

    result = impl(P, Q)
    outcome = "Satisfied ✅" if result else "Violated ❌"

    print(f"P = {tf(P)} (Password Correct), Q = {tf(Q)} (Access Granted)")
    print("Result:", outcome)

    log_result(student_name, "Login Rule", outcome)

# ---------- Rule 4: Bonus Points ---------- #
def bonus_rule(student_name):
    print("\n--- Bonus Points Eligibility Checker ---")
      regular = input("Does the student have regular attendance? (T/F):
").strip().upper() == "T"
    bonus = regular  # If regular attendance, then eligible for bonus

    result = impl(regular, bonus)
    outcome = "Satisfied ✅" if result else "Violated ❌"
```

```
    print(f"P = {tf(regular)} (Regular Attendance), Q = {tf(bonus)} (Bonus
Eligible)")
    print("Result:", outcome)

    log_result(student_name, "Bonus Rule", outcome)

# ---------- Extension Rule: Library Borrowing ---------- #
def library_rule(student_name):
    print("\n--- Library Borrowing Rule Checker ---")
        valid_id = input("Does the student have a valid ID? (T/F):
").strip().upper() == "T"
    borrow_allowed = valid_id  # If ID is valid, borrowing is allowed

    result = impl(valid_id, borrow_allowed)
    outcome = "Satisfied ✅" if result else "Violated ❌"

     print(f"P = {tf(valid_id)} (Valid ID), Q = {tf(borrow_allowed)} (Can
Borrow Books)")
    print("Result:", outcome)

    log_result(student_name, "Library Rule", outcome)

# --------------------- Main Menu --------------------- #
def main():
    print("=== University Logic Rule System ===")
    student_name = input("Enter Student Name: ").strip()

    while True:
        print("\n==============================")
        print("    Main Menu")
        print("==============================")
        print("1) Attendance Rule Checker")
        print("2) Grading Rule Checker")
        print("3) Login System Rule Checker")
        print("4) Bonus Points Checker")
        print("5) Library Rule Checker (Extension)")
        print("6) Exit")
```

```
        choice = input("Choose an Option: ").strip()

        match choice:
            case "1": attendance_rule(student_name)
            case "2": grading_rule(student_name)
            case "3": login_rule(student_name)
            case "4": bonus_rule(student_name)
            case "5": library_rule(student_name)
            case "6":
                print("Exiting... Results saved to logic_results.csv")
                break
            case _: print("Unknown Choice")

# --------------------- Entry Point --------------------- #
if __name__ == "__main__":
    # Create CSV with headers if not exist
     with open("logic_results.csv", "a", newline="", encoding="utf-8") as
file:
        writer = csv.writer(file)
        if file.tell() == 0:  # Only write the header if file is empty
                    writer.writerow(["Timestamp", "Student Name", "Rule",
"Result"])

    main()
```

Colab Notebook:

```
"""
Enhanced Mini Expert System: University Logic Rules
With CSV Logging for Record Keeping (Google Colab Version)
"""

import csv
from datetime import datetime
from google.colab import files

# ----------------- Logic Functions ----------------- #
def impl(P, Q):
```

```
    return (not P) or Q  # Implication (P -> Q)

def tf(b: bool) -> str:
    return "T" if b else "F"

# --------------------- Logger --------------------- #
def log_result(student_name, rule_name, result):
     with open("logic_results.csv", "a", newline="", encoding="utf-8") as
file:
        writer = csv.writer(file)
        writer.writerow([
            datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            student_name, rule_name, result
        ])

# ---------- Rule 1: Attendance ---------- #
def attendance_rule(student_name):
    print("\n--- Attendance Rule Checker ---")
    late = input("Is the student late? (T/F): ").strip().upper() == "T"
          excuse  =  input("Did  you  bring  an  excuse  letter?  (T/F):
").strip().upper() == "T"

    result = impl(late, excuse)
    outcome = "Satisfied ✅" if result else "Violated ❌"

    print(f"P = {tf(late)} (Late), Q = {tf(excuse)} (Excuse Letter)")
    print("Result:", outcome)

    log_result(student_name, "Attendance Rule", outcome)

# ---------- Rule 2: Grading ---------- #
def grading_rule(student_name):
    print("\n--- Grading Rule Checker ---")
    try:
        grade = float(input("Enter Student Grade: "))
    except ValueError:
        print("Invalid Grade Input")
        return
```

```
    P = grade >= 75   # Condition: passing grade
    Q = grade >= 75   # Conclusion: student passes

    result = impl(P, Q)
    outcome = "Satisfied ✅" if result else "Violated ❌"

    print(f"P = {tf(P)} (grade >= 75), Q = {tf(Q)} (Student Passes)")
    print("Result:", outcome)

    log_result(student_name, "Grading Rule", outcome)

# ---------- Rule 3: Login System ---------- #
def login_rule(student_name):
    print("\n--- Login Rule Checker ---")
    correct_password = "admin123"
    attempt = input("Enter Password: ")

    P = (attempt == correct_password)  # Password Correct
    Q = (attempt == correct_password)  # Access granted if correct

    result = impl(P, Q)
    outcome = "Satisfied ✅" if result else "Violated ❌"

    print(f"P = {tf(P)} (Password Correct), Q = {tf(Q)} (Access Granted)")
    print("Result:", outcome)

    log_result(student_name, "Login Rule", outcome)

# ---------- Rule 4: Bonus Points ---------- #
def bonus_rule(student_name):
    print("\n--- Bonus Points Eligibility Checker ---")
      regular = input("Does the student have regular attendance? (T/F):
").strip().upper() == "T"
    bonus = regular  # If regular attendance, then eligible for bonus

    result = impl(regular, bonus)
    outcome = "Satisfied ✅" if result else "Violated ❌"
```

```
    print(f"P = {tf(regular)} (Regular Attendance), Q = {tf(bonus)} (Bonus
Eligible)")
    print("Result:", outcome)

    log_result(student_name, "Bonus Rule", outcome)

# ---------- Extension Rule: Library Borrowing ---------- #
def library_rule(student_name):
    print("\n--- Library Borrowing Rule Checker ---")
        valid_id = input("Does the student have a valid ID? (T/F):
").strip().upper() == "T"
    borrow_allowed = valid_id  # If ID is valid, borrowing is allowed

    result = impl(valid_id, borrow_allowed)
    outcome = "Satisfied ✅" if result else "Violated ❌"

     print(f"P = {tf(valid_id)} (Valid ID), Q = {tf(borrow_allowed)} (Can
Borrow Books)")
    print("Result:", outcome)

    log_result(student_name, "Library Rule", outcome)

# --------------------- Main Menu --------------------- #
def main():
    print("=== University Logic Rule System ===")
    student_name = input("Enter Student Name: ").strip()

    while True:
        print("\n==============================")
        print("    Main Menu")
        print("==============================")
        print("1) Attendance Rule Checker")
        print("2) Grading Rule Checker")
        print("3) Login System Rule Checker")
        print("4) Bonus Points Checker")
        print("5) Library Rule Checker (Extension)")
        print("6) Exit")

        choice = input("Choose an Option: ").strip()
```

```
        match choice:
            case "1": attendance_rule(student_name)
            case "2": grading_rule(student_name)
            case "3": login_rule(student_name)
            case "4": bonus_rule(student_name)
            case "5": library_rule(student_name)
            case "6":
                print("Exiting... Results saved to logic_results.csv")
                # Auto-download the CSV file in Colab
                files.download("logic_results.csv")
                break
            case _: print("Unknown Choice")

# --------------------- Entry Point --------------------- #
if __name__ == "__main__":
    # Create CSV with headers if not exist
     with open("logic_results.csv", "a", newline="", encoding="utf-8") as
file:
        writer = csv.writer(file)
        if file.tell() == 0:  # Only write the header if file is empty
                    writer.writerow(["Timestamp", "Student Name", "Rule",
"Result"])

    main()
```

Python alternate Code:

```
"""
Enchanced Mini Expert System: University Logic Rules
With CSV Logging for Record Keeping
"""

import csv
from datetime import datetime

# ----------------- Logic Functions ----------------- #
def impl(P, Q):
```

```
    return (not P) or Q # Implication (P -> Q)

def tf(b: bool) -> str:
    return "T" if b else "F"

# --------------------- Logger --------------------- #
def log_result(student_name, rule_name, result):
    with open("logic_results.csv", "a", newline="") as file:
        writer = csv.writer(file)
        writer.writerow([
            datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            student_name, rule_name, result
        ])

# ---------- Rule 1: Attendance ---------- #
def attendance_rule(student_name):
    print("\n--- Attendance Rule Checker ---")
    late = input("Is the student late? (T/F): ").strip().upper() == "T"
          excuse = input("Did you bring an excuse letter (T/F):
").strip().upper() == "T"

    result = impl(late, excuse)
    outcome = "Satisfied" if result else "Violated"
    print(f"P = {tf(late)} (Late), Q = {tf(excuse)} (Excuse Letter)")
    print("Result:", outcome)

    log_result(student_name, "Attendance Rule", outcome)

# ---------- Rule 2: Grading ---------- #
def grading_rule(student_name):
    print("\n--- Grading Rule Checker ---")
    try:
        grade = float(input("Enter Student Grade: "))
    except ValueError:
        print("Invalid Grade Input")
        return

    P = grade >= 75
    Q = grade >= 75
```

```
    result = impl(P, Q)
    outcome = "Satisfied" if result else "Violated"

    print(f"P = {tf(P)} (grade >= 75), Q = {tf(Q)} (Student Passes)")
    print("Result:", outcome)

    log_result(student_name, "Grading Rule", outcome)

# ---------- Rule 3: Login System ---------- #
def login_rule(student_name):
    print("\n--- Login Rule Checker ---")
    correct_password = "admin123"
    attempt = input("Enter Password: ")

    P = (attempt == correct_password) # Password Correct
    Q = (attempt == correct_password) # Access granted if correct

    result = impl(P, Q)
    outcome = "Satisfied" if result else "Violated"

    print(f"P = {tf(P)} (Password Correct), Q = {tf(Q)} (Access Granted)")
    print("Result:", outcome)

    log_result(student_name, "Login Rule", outcome)

# ---------- Rule 3: Bonus Points ---------- #
def bonus_rule(student_name):
    print("\n--- Bonus Points Eligibility Checker ---")
      regular = input("Does the student have regular attendance? (T/F):
").strip().upper() == "T"
    bonus = regular

    result = impl(regular, bonus)
    outcome = "Satisfied" if result else "Violated"

     print(f"P = {tf(result)} (Regular Attendance), Q = {tf(bonus)} (Bonus
Eligible)")
    print("Result:", outcome)
```

```
    log_result(student_name, "Bonus Rule", outcome)

def main():
    print("=== University Logic Rule System ===")
    student_name = input("Enter Student Name: ".strip())

    while True:
        print("\n==============================")
        print("    Main Menu")
        print("==============================")

        print("1) Attendance Rule Checker")
        print("2) Grading Rule Checker")
        print("3) Login System Rule Checker")
        print("4) Bonus Points Checker")
        print("5) Exit")

        choice = input("Choose an Option: ").strip()

        match choice:
            case "1": attendance_rule(student_name)
            case "2": grading_rule(student_name)
            case "3": login_rule(student_name)
            case "4": bonus_rule(student_name)
            case "5":
                print("Exiting... Results saved to logic_results.csv")
                break
            case _: print("Unknown Choice")


if __name__ == "__main__":
    # Create CSV with headers if not exist
    with open("logic_results.csv", "a", newline="") as file:
        writer = csv.writer(file)
        if file.tell() == 0: # Only write the header if file is empty
                    writer.writerow(["Timestamp", "Student Name", "Rule",
"Result"])

main()
```

Republic of the Philippines
Laguna State Polytechnic University
Main Campus, Sta. Cruz, Laguna

Short Report Enhanced Mini Expert System – University Logic Rules
Explanation of Rules Tested

In this activity, we implemented a Mini Expert System using Python to simulate how propositional logic can be applied to real-world university scenarios. The system evaluates logical rules and logs results into a CSV file for record-keeping. Below are the rules tested

Attendance Rule

- Logic If a student is late (P), then they must bring an excuse letter (Q).
- Represented as P → Q
- If the student is late without an excuse, the rule is violated; otherwise, it is satisfied.

Grading Rule

- Logic If a student's grade is ≥ 75 (P), then the student passes (Q).
- Represented as P → Q
- Ensures that only students who meet the grade requirement are marked as passing.

Login System Rule

- Logic If the password entered is correct (P), then access is granted (Q).
- Represented as P → Q
- Incorrect passwords automatically lead to access denial.

Bonus Points Rule

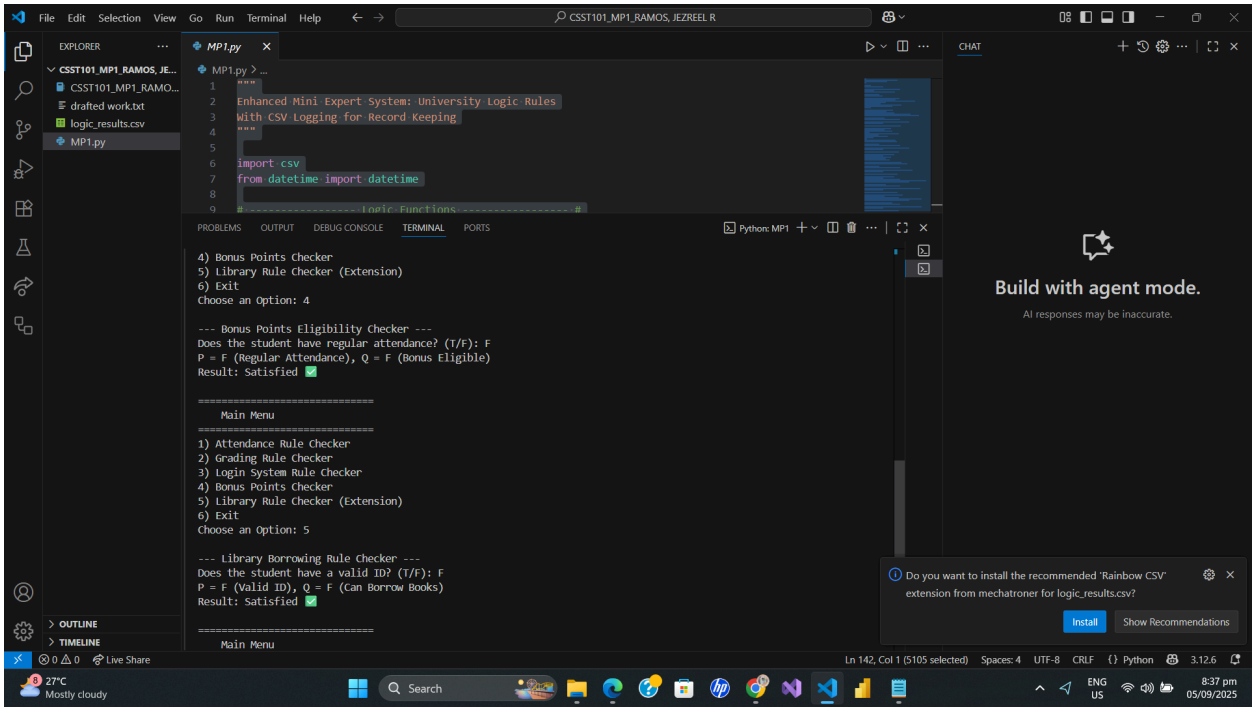
- Logic If a student has regular attendance (P), then they are eligible for bonus points (Q).
- Represented as P → Q
- Promotes consistent attendance by rewarding students.

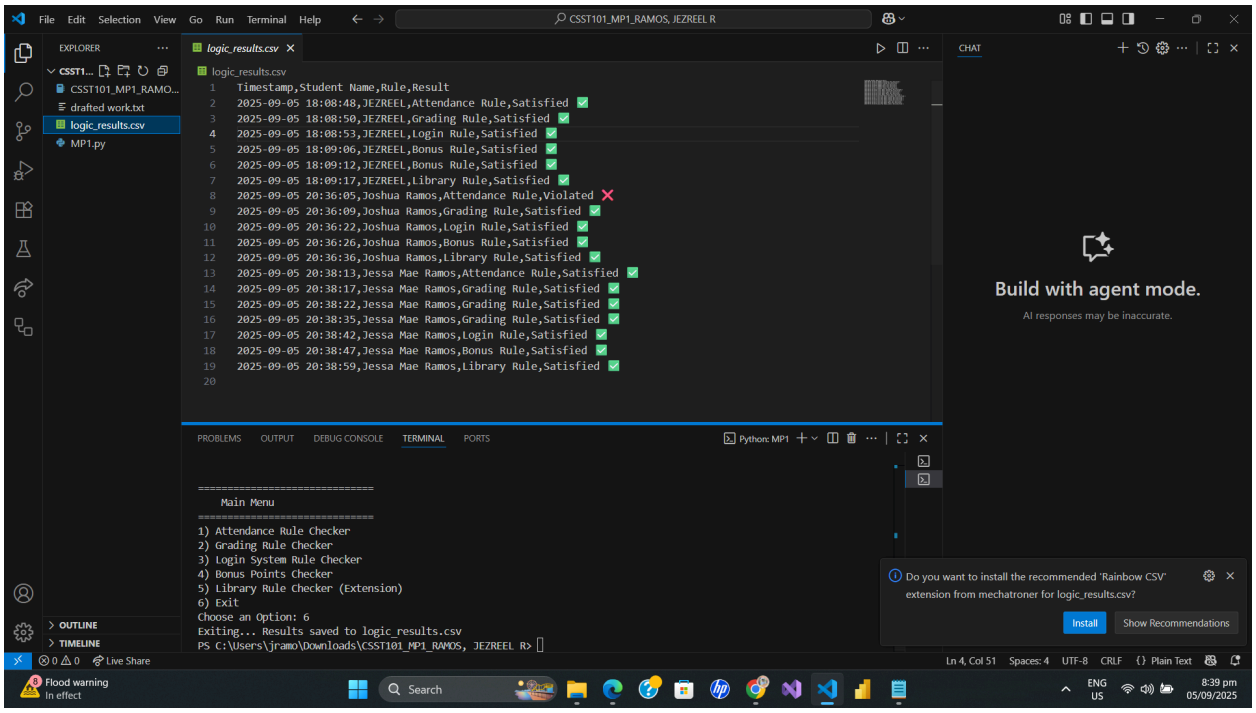
Extension Rule – Library Borrowing

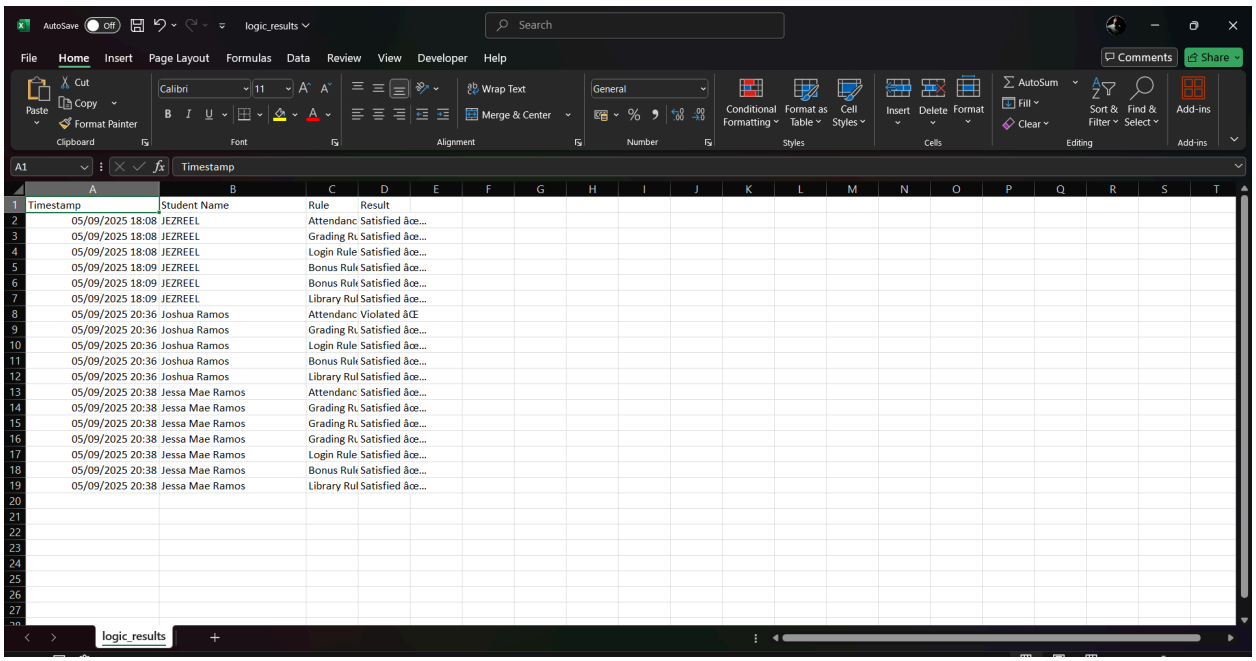
- Logic If a student has a valid ID (P), then they are allowed to borrow books (Q).
- Represented as P → Q
- This was added as the custom extension rule, applying logic to library services.

Republic of the Philippines
Laguna State Polytechnic University
Main Campus, Sta. Cruz, Laguna

Screenshots of Program Runs







Description of the New Rule Added

The Library Borrowing Rule was the extension rule included in the system. This rule checks whether a student has a valid ID before allowing them to borrow books. It reflects a common university policy where possession of a valid ID is a requirement for accessing library services.

This rule fits the logical structure P → Q, where:

- P = Student has a valid ID.
- Q = Borrowing is allowed.

If a student does not have a valid ID (P is false), the rule still holds true (vacuously true). However, if P is true but Q is false (i.e., the student has a valid ID but borrowing is denied), the rule is violated.

Conclusion

The Enhanced Mini Expert System successfully demonstrated how propositional logic can be applied in academic settings through rule-based decision-making. The system ensured that all results were automatically saved in a CSV file for analysis. By implementing the Attendance, Grading, Login, Bonus, and Library Borrowing rules, we were able to practice translating real-world conditions into logical implications. This activity reinforced understanding of how logical implication (P → Q) works and highlighted the practical use of expert systems in university management processes.

Sample Assessment Questions (for Written Part)

1. Represent the following in propositional logic:
   a. "If a student is late, then they must bring an excuse letter."
   b. "If a grade ≥ 75, then the student passes."
2. Why is the rule P → Q considered satisfied when P is false?
3. Examine the following CSV excerpt:

```
2025-09-04 14:33:15, Juan Dela Cruz, Grading Rule, Satisfied ✅
2025-09-04 14:34:22, Ana Santos, Login Rule, Access denied ❌
```

   - What does each row represent?
   - Which logical implication failed and why?

Answers:

1. Propositional forms
   a) Let L = "student is late," E = "brings an excuse letter." L → E
   b) Let G = "grade ≥ 75," P = "student passes." G → P

2. Why P → Q is satisfied when P is false?

Because P → Q ≡ ¬P ∨ Q. If P is false, then ¬P is true, so the whole expression is true regardless of Q. Intuitively, the promise “if P then Q” isn’t broken when P never happens.

3. CSV excerpt analysis

- Each row is one rule evaluation with: timestamp, student name, rule checked, outcome.
- Which implication failed and why?
    - Row 1 (Juan Dela Cruz, Grading Rule, “Satisfied”): the implication G → P held (grade met the cutoff, so pass).
    - Row 2 (Ana Santos, Login Rule, “Access denied”): the rule is CorrectPassword → AccessGranted. “Access denied” means Q is false and (by context) P is false (password incorrect). For P false, P → Q is still true (vacuously).

Therefore, no logical implication failed in the excerpt. Access was denied not because the implication was violated, but because the antecedent (correct password) wasn’t met. The only way P → Q fails is P true and Q false, which isn’t shown here.