Nick Narry Mendoza **September 6, 2025**
BSCS-3A **Mr. Bernardino**

ASSESSMENT REPORT: Applying Propositional Logic in Real-World Scenarios through a Mini Expert System

I. INTRODUCTION

This report presents the implementation and testing of a Mini Expert System that applies propositional logic to real-world student scenarios. The system evaluates five distinct rules using logical implication (P → Q) to make automated decisions based on student data.

II. RULES IMPLEMENTED AND TESTED

2.1 Core Rules

1. Attendance Rule

- **Logic**: attendance_pct ≥ 75 → eligible
- **Implementation**: If student attendance is 75% or higher, they are eligible for course completion
- **Real-world application**: Academic institutions require minimum attendance for course eligibility

2. Grading Rule

- **Logic**: final_grade ≥ 75 → pass
- **Implementation**: Students with final grade 75 or above pass the course
- **Real-world application**: Standard passing grade threshold in educational systems

3. Login System Rule

- **Logic**: (username_valid ∧ password_valid ∧ ¬account_locked) → login_success
- **Implementation**: Successful login requires valid credentials and unlocked account
- **Real-world application**: System security and access control

4. Bonus Points Rule

- **Logic**: participated → bonus_points_added
- **Implementation**: Students who participated in activities receive bonus points (capped at 100)
- **Real-world application**: Incentivizing student engagement and participation

2.2 Extended Rule (New Implementation)

5. Library Borrowing Rule

- **Logic**: (id_valid ∧ ¬has_overdue_books) → borrowing_allowed
- **Implementation**: Students can borrow books only if they have valid ID and no overdue items
- **Real-world application**: Library management and resource allocation

```
test_students = [
   Student(name="Avellaneda", attendance_pct=82.5, final_grade=78.0,
           username_ok=True, password_ok=True, is_locked=False,
           participated=True, base_score=88.0, id_valid=True, has_overdue=False),

   Student(name="Capili", attendance_pct=70.0, final_grade=72.0,
           username_ok=True, password_ok=False, is_locked=False,
           participated=False, base_score=65.0, id_valid=True, has_overdue=True),

   Student(name="Ramos", attendance_pct=95.0, final_grade=92.0,
           username_ok=True, password_ok=True, is_locked=True,
           participated=True, base_score=96.0, id_valid=False, has_overdue=False)
]
```

```
Student: Avellaneda
 ✓ AttendanceRule     : attendance=82.5% -> eligible
 ✓ GradingRule        : grade=78.0 -> pass
 ✓ LoginSystemRule    : user_ok=True, pass_ok=True, locked=False -> login success
 ✓ BonusPointsRule    : participated=True, base=88.0 -> bonus +5.0, final=93.0
 ✓ LibraryBorrowingRule: id_valid=True, overdue=False -> allowed
```

```
Student: Capili
 ✗ AttendanceRule     : attendance=70.0% -> not eligible
 ✗ GradingRule        : grade=72.0 -> fail
 ✗ LoginSystemRule    : user_ok=True, pass_ok=False, locked=False -> login denied
 ✗ BonusPointsRule    : participated=False, base=65.0 -> no bonus, final=65.0
 ✗ LibraryBorrowingRule: id_valid=True, overdue=True -> not allowed

Student: Ramos
 ✓ AttendanceRule     : attendance=95.0% -> eligible
 ✓ GradingRule        : grade=92.0 -> pass
 ✗ LoginSystemRule    : user_ok=True, pass_ok=True, locked=True -> login denied
 ✓ BonusPointsRule    : participated=True, base=96.0 -> bonus +5.0, final=100.0
 ✗ LibraryBorrowingRule: id_valid=False, overdue=False -> not allowed
```

IV. CSV DATA ANALYSIS

The generated CSV file contains structured data with the following columns:

- timestamp: Execution time
- student: Student name
- Rule columns: Boolean results (True/False)
- Detail columns: Detailed explanations of each rule evaluation

Sample CSV content:

```
timestamp,student,AttendanceRule,AttendanceDetail,GradingRule,GradingDetail,LoginSystemRule,LoginDetail,BonusPointsRule,BonusDetail,LibraryBorrowingRule,LibraryDetail
2025-09-06T14:30:41,Avellaneda,True,attendance=82.5% -> eligible,True,grade=78.0 -> pass,True,"user_ok=True, pass_ok=True, locked=False -> login success",True,"participated=True, base=88.0 -> bonus +5.0, final=93.0",True,"id_valid=True, overdue=False -> allowed"
```

V. LOGICAL ANALYSIS

5.1 Propositional Logic Implementation

Each rule demonstrates proper logical implication:

1. **Attendance**: P = (attendance ≥ 75), Q = eligible → P → Q
2. **Grading**: P = (grade ≥ 75), Q = pass → P → Q
3. **Login**: P = (valid_user ∧ valid_pass ∧ ¬locked), Q = access_granted → P → Q
4. **Bonus**: P = participated, Q = bonus_added → P → Q
5. **Library**: P = (valid_id ∧ ¬overdue), Q = borrowing_allowed → P → Q

5.2 Decision-Making Scenarios

The system successfully demonstrates automated decision-making based on logical conditions, providing clear reasoning for each outcome.

VI. NEW RULE DESCRIPTION

Library Borrowing Rule Implementation:

```
def library_borrowing_rule(self, id_valid: bool, has_overdue: bool) -> Tuple[bool, str]:
    """
    Library Borrowing Rule: If ID valid AND no overdue items -> Allowed; else -> Not allowed
    """
    ok = id_valid and (not has_overdue)
    return ok, f"id_valid={id_valid}, overdue={has_overdue} -> {'allowed' if ok else 'not allowed'}"
```

This rule implements the logical expression: (valid_id ∧ ¬overdue_books) → borrowing_allowed

Real-world significance: This rule prevents students with overdue books from borrowing additional materials and ensures only registered students can access library resources.

VII. CONCLUSION

The Mini Expert System successfully demonstrates:

1. Translation of real-world conditions into propositional logic
2. Implementation of logical implication in decision-making
3. Automated rule evaluation with detailed explanations
4. Structured data logging for analysis
5. Extensibility through additional rule implementation

The system provides a solid foundation for understanding how propositional logic can be applied to automate decision-making in educational and administrative contexts.

VIII. TECHNICAL SPECIFICATIONS

- **Programming Language**: Python 3.12
- **Data Structure**: CSV file for persistence
- **Architecture**: Object-oriented design with separation of concerns
- **Error Handling**: Input validation and exception management
- **Output Format**: Structured logging with timestamps and detailed explanations