# IUBAT-International University of Business Agriculture and Technology

## Assignment


**Submitted by:**

Name: Md. Fahim Hossain

ID: 22103093

Program: BCSE

Course code: CSC-397

Section: N

**Submitted to:**

Avijit Biswas

Lecturer

Dept. of Computer Science and Engineering


Date of Submission: 14/09/25

# DFA-Based Access Control System for a Smart Building

## 1. Introduction to DFA & Access Control

Deterministic Finite Automata (DFA) provide a rigorous mathematical framework for modeling authentication workflows. In a smart building, secure access requires users to follow a strict, zone-specific sequence of authentication factors (e.g., CARD → FINGER → RETINA → PIN**)**. Any deviation is immediately rejected to prevent bypass or replay attacks**.**

This project develops a family of zone-specific DFAs where:

- For each state and input symbol, there exists exactly one next state **(**determinism**).**
- Invalid or out-of-order inputs lead to a sink REJECT state with self-loops for all inputs, ensuring no escape from rejection.

## 2. Problem Analysis & Constraint Resolution

### Requirement-Level Constraints

- **Unique Sequences per Zone:** Eight restricted zones, each with a distinct policy of four or more steps, ensuring no reuse across zones.
- **Multi-Factor Enforcement:** Every accepted sequence requires ≥ 4 authentication steps.
- **Immediate Rejection:** Any invalid symbol at any stage transitions directly to the sink REJECT state.

### Technical-Level Constraints

- **Determinism:** Each pair (state, symbol) maps to exactly one next state. Since multiple zones may share prefixes (e.g., starting with CARD), we instantiate a **separate DFA per zone** to preserve determinism.
- **Variable Length Handling:** The DFA length equals the zone's policy length. Acceptance occurs only if the sequence ends exactly in the final accepting state with no extra inputs.
- **Invalid Symbol Handling:** Any token outside the defined alphabet is rejected immediately with an explanatory error.
- **Extra Inputs:** From the accept state, all inputs transition to REJECT, preventing overrun or ambiguous acceptance.

### 3. Input Symbols & Zone Policies

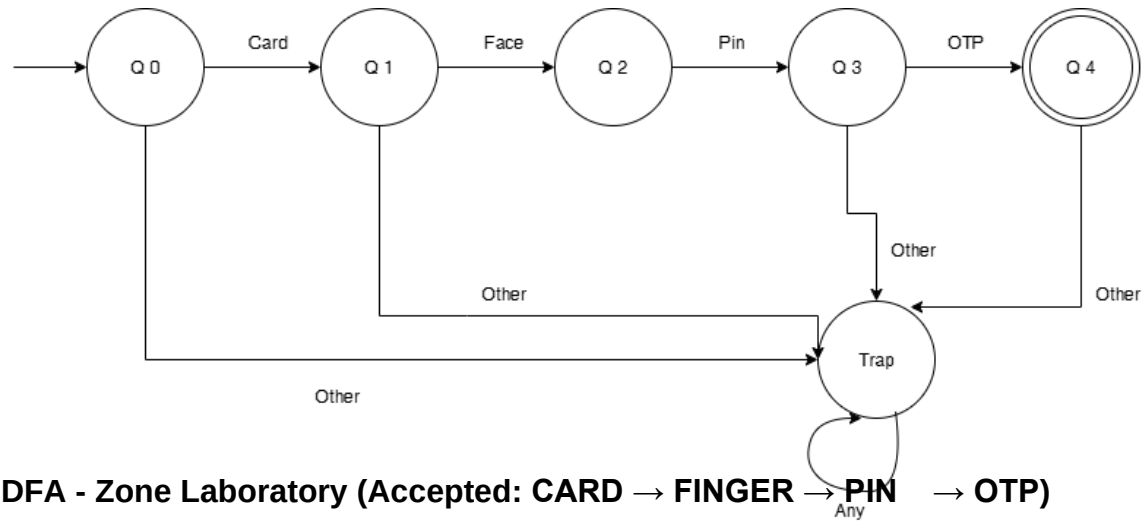
Authentication Alphabet (Σ): CARD, FINGER, RETINA, FACE, VOICE, PIN, ADMIN, OTP

**Zones and Their Unique Policies (δ along the main path):**

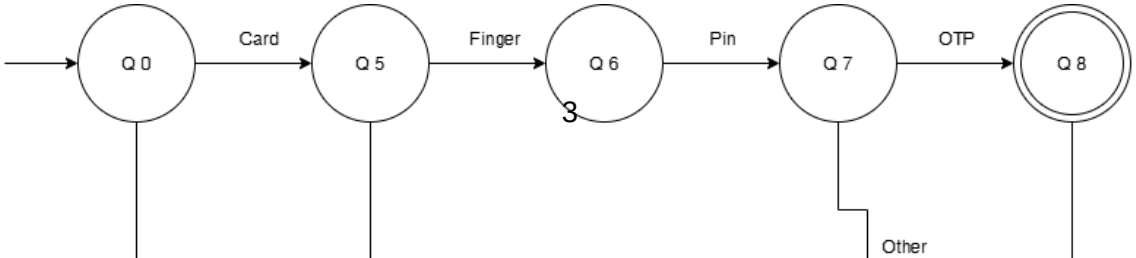
1) Lobby : CARD → FACE → PIN → OTP
2) Laboratory : CARD → FINGER → PIN → OTP
3) ServerRoom : CARD → FINGER → RETINA → PIN
4) ExecutiveLounge : FACE → VOICE → CARD → PIN
5) ResearchWing : CARD → VOICE → FINGER → PIN
6) DataCenterVault : CARD → RETINA → FINGER → PIN → OTP
7) MaintenanceWorkshop : CARD → PIN → FINGER → FACE
8) SecurityOpsRoom : ADMIN → CARD → RETINA → PIN

### 4. DFA State Diagram:

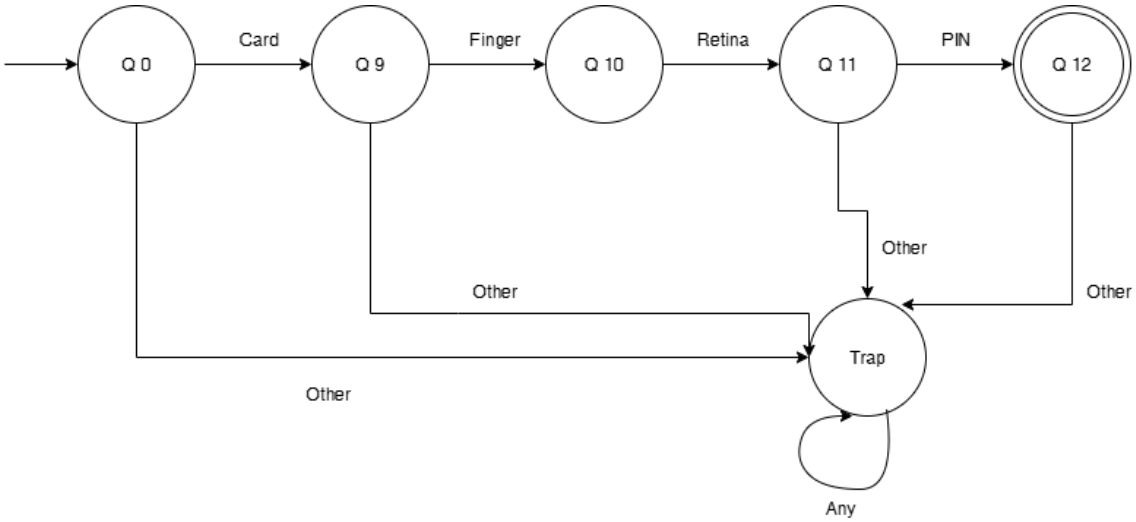
**DFA - Zone Lobby (Accepted: CARD → FACE → PIN → OTP)**



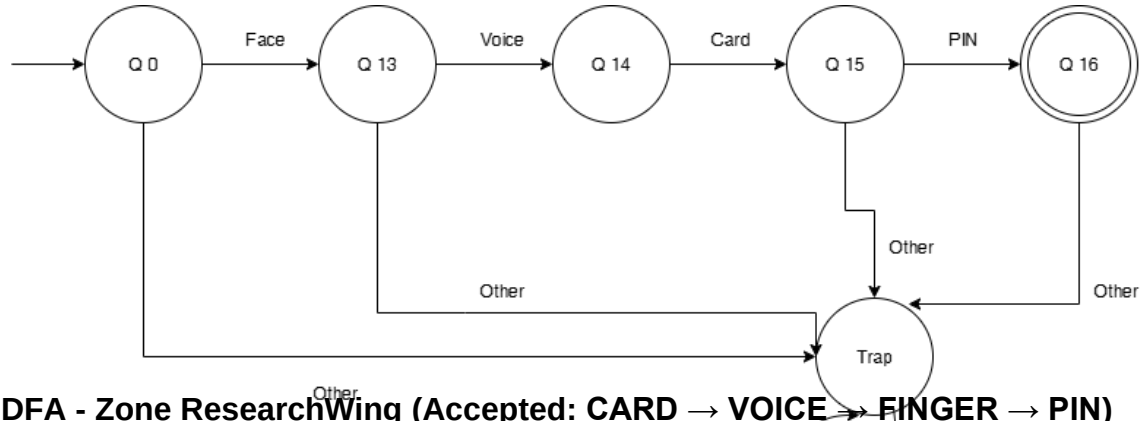

**DFA - Zone Laboratory (Accepted: CARD → FINGER → PIN → OTP)**



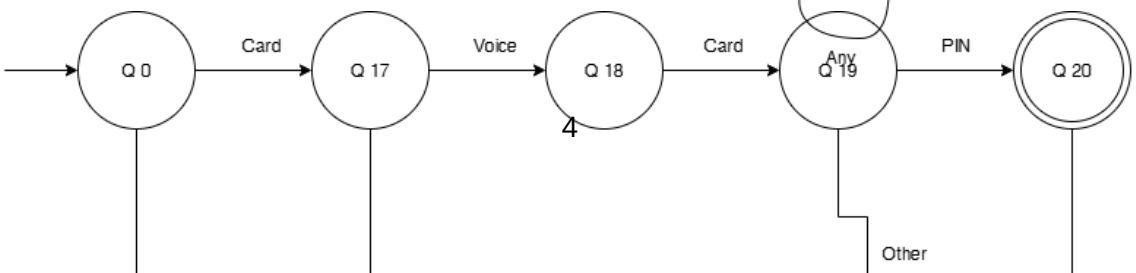
**DFA - Zone ServerRoom (Accepted: CARD → FINGER → RETINA → PIN)**



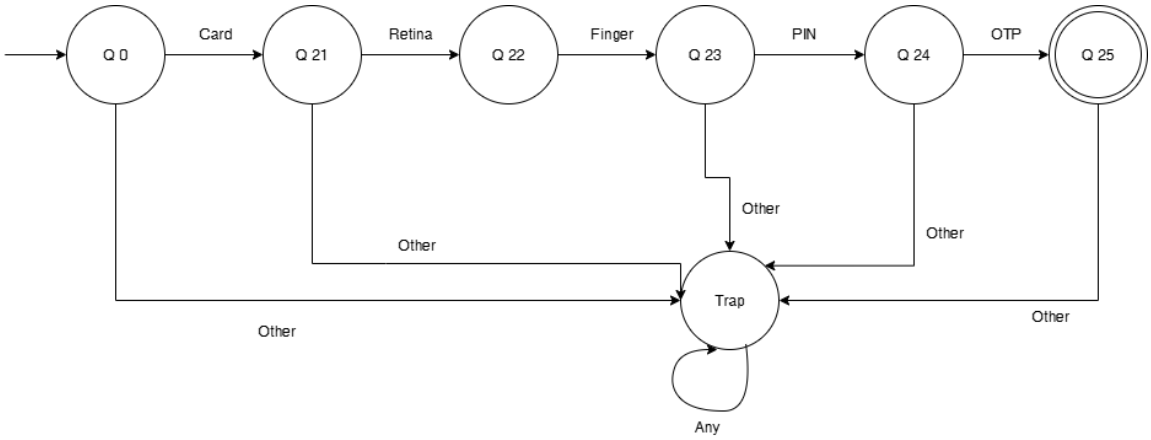

**DFA - Zone ExecutiveLounge (Accepted: FACE → VOICE → CARD → PIN)**



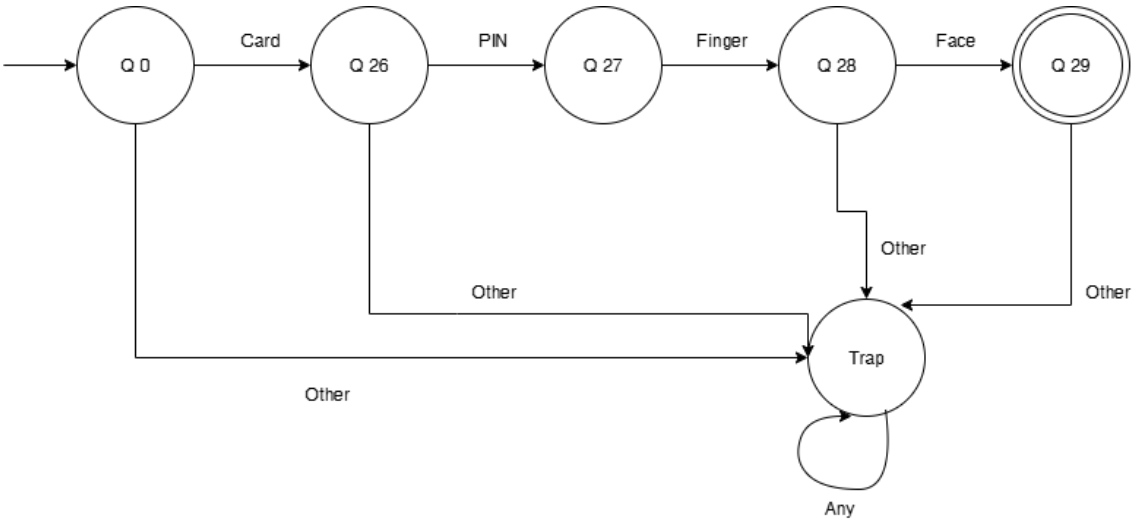

**DFA - Zone ResearchWing (Accepted: CARD → VOICE → FINGER → PIN)**



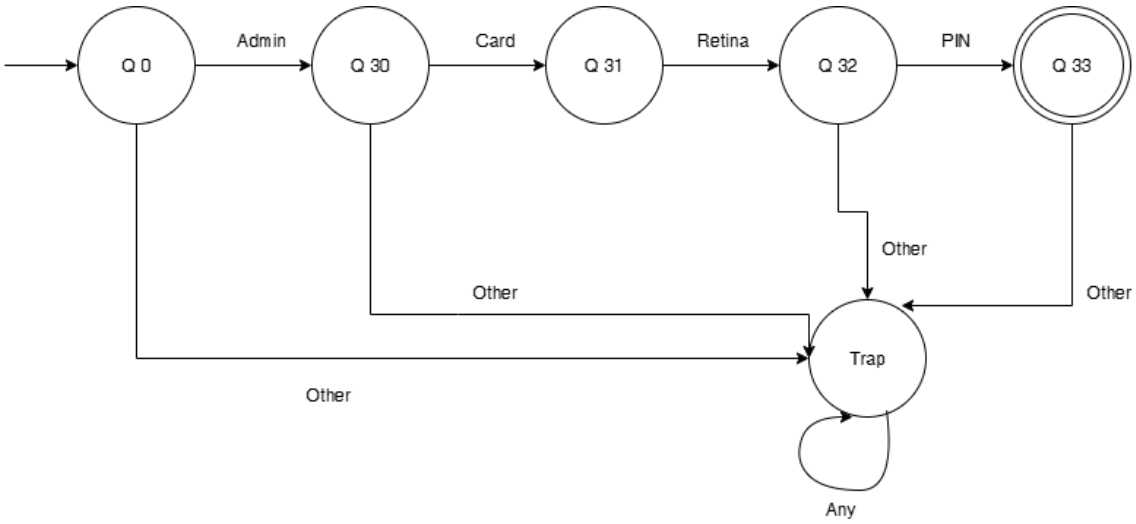
**DFA - Zone DataCenterVault (Accepted: CARD → RETINA → FINGER → PIN → OTP)**




**DFA - Zone MaintenanceWorkshop (Accepted: CARD → PIN → FINGER → FACE)**



**DFA - Zone SecurityOpsRoom (Accepted: ADMIN → CARD → RETINA → PIN)**




## 5. Transition Tables (Summary)

Complete transition tables for all zones are included as a separate CSV file (transition_tables.csv).

| ZONE | STATE | CARD | FINGER | RETINA | FACE | VOICE | PIN | ADMIN | OTP |
|---|---|---|---|---|---|---|---|---|---|
| **Lobby** | q0 | q1 | REJECT | REJECT | q2 | REJECT | q3 | REJECT | q4 |
| **Laboratory** | q0 | q5 | q6 | REJECT | REJECT | REJECT | q7 | REJECT | q8 |
| **ServerRoom** | q0 | q9 | q10 | q11 | REJECT | REJECT | q12 | REJECT | REJECT |
| **ExecutiveLounge** | q0 | q15 | REJECT | REJECT | q13 | q14 | q16 | REJECT | REJECT |
| **ResearchWing** | q0 | q17 | q19 | REJECT | REJECT | q18 | q20 | REJECT | REJECT |
| **DataCenterVault** | q0 | q21 | q23 | q22 | REJECT | REJECT | q24 | REJECT | q25 |
| **MaintenanceWorkshop** | q0 | q26 | q28 | REJECT | q29 | REJECT | q27 | REJECT | REJECT |
| **SecurityOpsRoom** | q0 | q31 | REJECT | q32 | REJECT | REJECT | q33 | q30 | REJECT |

Each table enumerates states q0..qN and REJECT; for each symbol in Σ, δ(state, symbol) is defined.
From REJECT, δ(REJECT, symbol) = REJECT for all symbols (sink). From the accept state qN, all inputs transition to REJECT, so extra inputs cause denial, preventing ambiguous behavior.

## 6. Code & Implementation Overview

```
# DFA-based Access Control System (Online Compiler Version)

ALPHABET = ["CARD", "FINGER", "RETINA", "FACE", "VOICE", "PIN",
"ADMIN", "OTP"]

ZONE_POLICIES = {
    "Lobby": ["CARD", "FACE", "PIN", "OTP"],
    "Laboratory": ["CARD", "FINGER", "PIN", "OTP"],
    "ServerRoom": ["CARD", "FINGER", "RETINA", "PIN"],
    "ExecutiveLounge": ["FACE", "VOICE", "CARD", "PIN"],
    "ResearchWing": ["CARD", "VOICE", "FINGER", "PIN"],
    "DataCenterVault": ["CARD", "RETINA", "FINGER", "PIN",
"OTP"],
    "MaintenanceWorkshop": ["CARD", "PIN", "FINGER", "FACE"],
    "SecurityOpsRoom": ["ADMIN", "CARD", "RETINA", "PIN"],
}

class ZoneDFA:
    def __init__(self, zone, seq):
        self.zone = zone
        self.seq = seq
        self.start = "q0"
        self.accept = f"q{len(seq)}"
        self.reject = "REJECT"
        self.delta = {}
```

```
        # Build DFA transitions
        for i in range(len(seq)):
            expected = seq[i]
            for sym in ALPHABET:
                if sym == expected:
                    self.delta[(f"q{i}", sym)] = f"q{i+1}"
                else:
                    self.delta[(f"q{i}", sym)] = self.reject

        for sym in ALPHABET:
            self.delta[(self.accept, sym)] = self.reject
            self.delta[(self.reject, sym)] = self.reject

    def process(self, inputs):
        current = self.start
        path = [current]
        for idx, sym in enumerate(inputs, start=1):
            if sym not in ALPHABET:
                return False, path + [self.reject], f"Invalid
symbol '{sym}' at position {idx}."
            current = self.delta[(current, sym)]
            path.append(current)

        if current == self.accept:
            return True, path, "Access Granted."
        else:
            if current.startswith("q"):
```

```
                k = int(current[1:])
                expected_next = self.seq[k] if k < len(self.seq)
else None
                return False, path, f"Incomplete/incorrect
sequence. Expected next: {expected_next}."
            return False, path, "Wrong or out-of-order
authentication."


# ---------------- Main Section -----------------
print("Available Zones:", list(ZONE_POLICIES.keys()))
zone = input("Enter Zone Name: ").strip()
if zone not in ZONE_POLICIES:
    print("❌ Unknown Zone")
else:
    events = input("Enter authentication sequence (space-
separated): ").strip().split()
    dfa = ZoneDFA(zone, ZONE_POLICIES[zone])
    granted, path, reason = dfa.process(events)


    print("\n--- Result ---")
    print(f"Zone: {zone}")
    print(f"Input: {events}")
    print(f"State path: {' -> '.join(path)}")
    print("Decision:", "GRANTED ☑" if granted else "DENIED ❌")
    print("Reason:", reason)
```

The Python program implements a ZoneDFA class that constructs a deterministic transition function for a given zone policy. The application takes a zone name and a space-separated list of events as input, then runs the DFA. It prints the state path, a GRANTED/DENIED decision, and a reason.

Key correctness properties addressed:

• Determinism: For every (state, symbol), δ is total and single-valued.
• Immediate rejection: unexpected symbols go directly to REJECT.
• Exact acceptance: Acceptance only if the input ends in qN with no extra symbols.
• Invalid symbol handling: non-alphabet tokens are caught and explained.

## 7. Testing — Cases & Outcomes (Pass/Fail)

| Input Sequence | Expected | Actual | Pass/Fail | Reason/Notes | State Path |
|---|---|---|---|---|---|
| CARD FINGER RETINA PIN | GRANTED | GRANTED | PASS | All required factors provided in correct order; no extra inputs. | q0 -> q9 -> q10 -> q11 -> q12 |
| CARD RETINA FINGER PIN | DENIED | DENIED | PASS | Wrong or out-of-order authentication detected; transitioned to REJECT. | q0 -> q1 -> REJECT -> REJECT -> REJECT |
| CARD FINGER RETINA OTP | DENIED | DENIED | PASS | Wrong or out-of-order authentication detected; transitioned to REJECT. | q0 -> q1 -> q2 -> q3 -> REJECT |
| FACE VOICE CARD PIN OTP | DENIED | DENIED | PASS | Wrong or out-of-order authentication detected; transitioned to REJECT. | q0 -> q1 -> q2 -> q3 -> q4 -> REJECT |
| CARD FINGER PIN QR | DENIED | DENIED | PASS | Invalid symbol 'QR' encountered at position 4. Allowed: ['CARD', 'FINGER', 'RETINA', 'FACE', 'VOICE', 'PIN', 'ADMIN', 'OTP']. | q0 -> q1 -> q2 -> q3 -> REJECT |
| CARD FINGER PIN OTP | DENIED | DENIED | PASS | Wrong or out-of-order authentication detected; transitioned to REJECT. | q0 -> q1 -> REJECT -> REJECT -> REJECT |
| CARD FACE PIN OTP | GRANTED | GRANTED | PASS | All required factors provided in correct order; no extra inputs. | q0 -> q1 -> q2 -> q3 -> q4 |
| CARD PIN | DENIED | DENIED | PASS | Incomplete sequence. Expected next: FINGER. | q0 -> q1 -> q2 |
| CARD RETINA FINGER PIN OTP | GRANTED | GRANTED | PASS | All required factors provided in correct order; no extra inputs. | q0 -> q21 -> q22 -> q23 -> q24 -> q25 |
| ADMIN CARD RETINA PIN | GRANTED | GRANTED | PASS | All required factors provided in correct order; no extra inputs. | q0 -> q30 -> q31 -> q32 -> q33 |

Complete test results, which contains each test case with the expected output, actual output, and pass/fail status.
The tests cover the following scenarios:

• Correct sequence input for a zone
• Incorrect sequences (out-of-order or missing symbols)
• Extra inputs after a correct sequence
• Invalid symbols in the sequence

## 8. Conclusion & Justification of CEP

**Conclusion:**

This DFA-based access control design achieves secure, deterministic, and auditable authentication workflows for eight smart-building zones with unique multi-factor sequences. The sink REJECT state ensures immediate denial for invalid/out-of-order events, while exact acceptance prevents replay or overrun exploits.

**CEP Justification:**

| **Complex Engineering Criteria** | **Knowledge Profile** | **Knowledge Profile** |
|---|---|---|
| Problem Analysis | K2, K3, K4 | Applied computer science principles to analyze constraints and design appropriate DFA structure |
| DFA Design | P1, P2, P3 | Engineering fundamentals applied to create minimal, efficient state machine with optimal state sharing |
| Implementation | P1, P2, P3 | Specialist knowledge demonstrated through robust code implementation with complete error handling |
| Testing & Validation | K4, P3 | Comprehensive test cases validate all requirements and edge cases |
| Documentation | K2, K3 | Complete technical documentation with clear explanations and justifications |

Assignment Assessment Rubric (Especially for PO(b) Problem Analysis)

| Criteria | Excellent 910 | Proficient 7-8 | Adequate 06 | Limited 4-5 | Insufficient |
|---|---|---|---|---|---|
| Application of the Engineering Knowledge. (10%) | All required engineering fundamental ideas & knowledge are present. Access enough resources to understand the problem including mathematical approaches. | Most required engineering fundamental ideas & knowledge are present. Access enough resources to understand the problem including mathematical approaches. | Some required fundamental ideas and topics are present. Access not enough resources to understand the problem. | Few required fundamental ideas and topics are present. Access mostly irrelevant resources to understand the problem. | No score will be awarded if there is insufficient evidence of student performance identified and irrelevant discussion is presented in the document |
| | Excellent 18-20 | Proficient 15-17 | Adequate 12-14 | Limited 8-11 | |
| Depth of analysis. (20%) | Proposed Solution reflects sufficient abstract thinking and in depth analysis. | Proposed Solution reflects reasonable abstract thinking and in depth analysis. | Proposed Solution reflects some form of abstract thinking and in depth analysis. | Proposed Solution reflects very few abstract thinking and in depth analysis. | |
| | Excellent 18-20 | Proficient 15-17 | Adequate 12-14 | Limited 8-11 | |
| Addressing the issues of conflicting requirements (20%) | Proposed Solution addresses all the issues of conflicting requirements. | Proposed Solution addresses most of the issues of conflicting requirements. | Proposed Solution addresses some of the issues of conflicting requirements. | Proposed Solution addresses very few issues of conflicting requirements. | |
| | Excellent 26-30 | Proficient 20-25 | Adequate 15-20 | Limited 10-15 | |
| The DFA solution design (30%) | The DFA solution design satisfies all the mentioned constraints. | The DFA solution design satisfies most of the mentioned constraints. | The DFA solution design satisfies some of the mentioned constraints. | The DFA solution design satisfies very few of the mentioned constraints. | |
| | Excellent 18-20 | Proficient 15-17 | Adequate 12-14 | Limited 8-11 | |

| | | | | | |
|---|---|---|---|---|---|
| The documentation / demonstration of the work with a proper and clear explanation. (20%) | Student coveys their knowledge and analytical skills with exceptional approach and logically bound. | Student coveys their knowledge and analytical skills with moderate approach and logically bound. | Student coveys their knowledge and analytical skills with average approach and in some areas, the logic or flow of ideas is difficult to follow. | Student coveys their knowledge and analytical skills with poor approach and in some areas, the logic or flow of ideas is unrecoverable. | |