IUBAT—INTERNATIONAL UNIVERSITY OF BUSINESS AGRICULTURE AND TECHNOLOGY

Founded 1991 by Md. Alimullah Miyan

Assignment On DFA

Topic Name: A Smart Building Access Control System

Course Name: Theory of Computation

Course Code: CSC397

Submitted To

Md Nazir Ahmed

Lecturer

Department of Computer Science & Engineering

Submitted By

Shuvo Chakrobortty

ID:22203229

Section: J

Submitted Date: 25/09/2025

A Smart Building Access Control System

Introduction to DFA and the access control system:

A **Deterministic Finite Automaton (DFA)** is a state machine where each input leads to exactly one next state, ensuring predictable outcomes. In a smart building access control system, DFA models authentication as a sequence of events such as card swipes, fingerprints, retina scans, or PIN entries. Each zone (Lobby, Server Room, Vault, etc.) has a unique multi-step policy represented by a DFA path. Correct sequences reach an accepting state (access granted), while any wrong or out-of-order input leads to a trap state (access denied). This makes DFA a secure and transparent way to enforce access rules.

Problem Analysis & Constraint Resolution Approach:

1. **Requirement-Level Analysis**
   - **Unique Sequences per Zone:** Each access zone must have its own distinct sequence of at least four authentication events. To address this, the DFA design assigns separate accepting states for each zone (e.g., q4 for Lobby, q6 for Laboratory, etc.), ensuring no overlap in authentication paths.
   - **Multi-Factor Authentication:** Since the minimum sequence length is four, each zone's path integrates at least four distinct authentication methods (e.g., Card → PIN → Face Recognition → Voice for the Lobby). This enforces strong security.
   - **Immediate Rejection of Invalid Sequences:** Any wrong or unexpected input symbol at any step transitions the system directly to the trap state (`qt`), which represents denial of access. This guarantees that partially correct or tampered sequences cannot bypass security.
2. **Technical-Level Analysis**
   - **Determinism:** The DFA ensures one and only one transition for each input symbol in every state. This is enforced in both the transition table and the Java implementation, where the `nextState` method maps each input to a single next state.
   - **Variable-Length Sequences:** The system supports sequences of varying lengths across different zones (but not fewer than four). The DFA design accommodates this by defining separate state chains for each zone's required sequence while still maintaining determinism.
   - **Scalability:** The modular structure of the transition table allows easy extension to new zones or authentication methods. Adding a new zone simply involves appending new states and transitions without affecting existing logic.

Resolution Approach:

- To resolve **conflicts between zones**, the DFA explicitly separates their sequences, preventing ambiguity.
- To handle **out-of-order inputs**, transitions are designed to route invalid attempts directly into the trap state.

- To balance **security and usability**, each authentication sequence is deterministic and strictly defined, minimizing user confusion while upholding strong protection.
- The Java program implementation reinforces these constraints by checking each input step in real time and providing immediate feedback (Access Granted / Access Denied).

Thus, the approach ensures that all requirement-level and technical-level constraints are systematically addressed, resulting in a secure, deterministic, and extensible access control DFA.

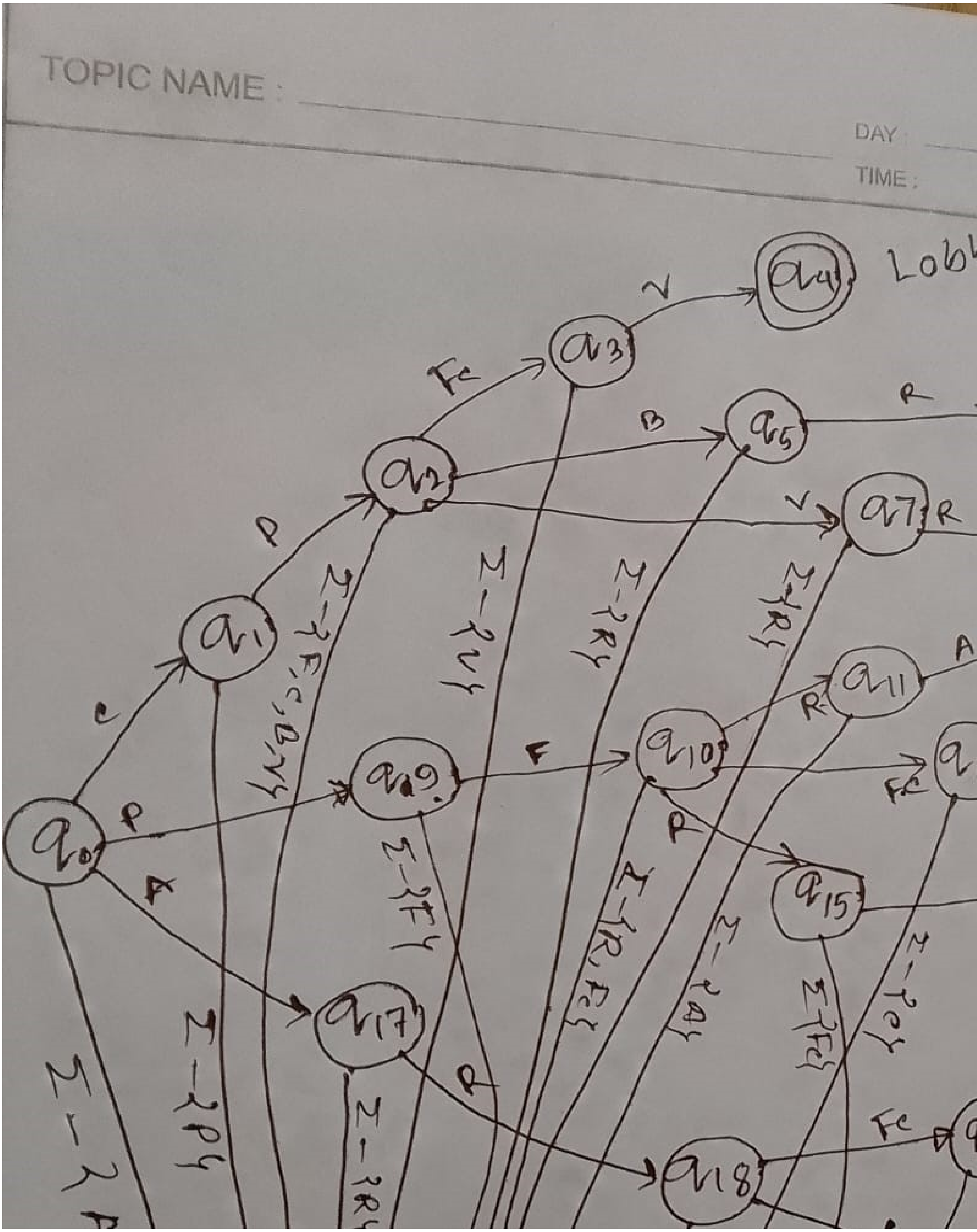
1. **Alphabets (Authentication Symbols):**

C	Card Swipe
F	Fingerprint Scan
R	Retina Scan
P	PIN Entry
V	Voice Recognition
F_c	Face Recognition
B	Biometric Combination
A	Admin Override

2. **Zones & Unique Authentication Sequences:**

Zone	Authentication Sequence (Inputs)	Accepting State
Lobby	C → P → Fc → V	q4
Laboratory	C → P → B → R	q6
Research Wing	C → P → V → R	q8
Server Room	P → F → R → A	q12
Executive Room	P → F → Fc → C	q14
Data Center	P → F → R2 → Fc	q16
Admin Room	A → R → Fc → B	q20
Vault	A → R → C → B	q22

DFA State Diagram:



DFA State Diagram:

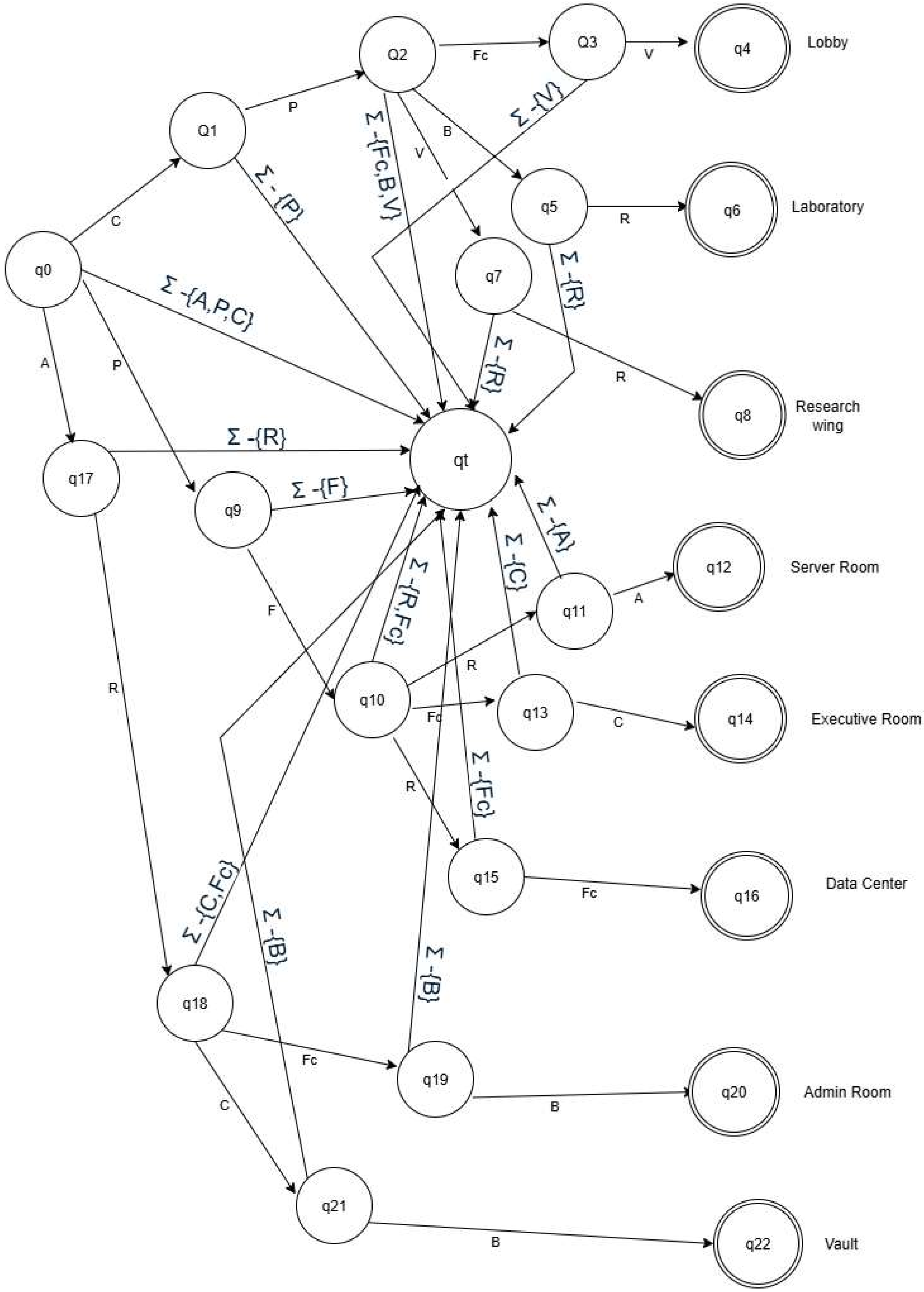


Figure: DFA Diagram for Smart Building Access Control System

1. Transition Table of DFA:

Current State	Input Symbol	Next State	Description
q0	C	q1	Start of Lobby sequence
q1	P	q2	Lobby authentication continues
q2	Fc	q3	Lobby sequence
q3	V	q4	Lobby → Accepting State
q2	B	q5	Laboratory path
q5	R	q6	Laboratory → Accepting State
q2	V	q7	Research Wing path
q7	R	q8	Research Wing → Accepting State
q0	P	q9	Server Room path
q9	F	q10	Server Room authentication continues
q10	R	q11	Server Room authentication continues
q11	A	q12	Server Room → Accepting State
q10	Fc	q13	Executive Room path
q13	C	q14	Executive Room → Accepting State
q10	R2	q15	Data Center path
q15	Fc	q16	Data Center → Accepting State
q0	A	q17	Admin Room path
q17	R	q18	Admin authentication continues
q18	Fc	q19	Admin authentication continues
q19	B	q20	Admin Room → Accepting State
q18	C	q21	Vault path
q21	B	q22	Vault → Accepting State
Any	Invalid	qt	Trap state (Access Denied)

- **Another way** Transition Table of DFA:

State	C	P	Fc	V	B	F	R	R2	A
q0	q1	q9	—	—	—	—	—	—	q17
q1	—	q2	—	—	—	—	—	—	—
q2	—	—	q3	q7	q5	—	—	—	—
q3	—	—	—	q4	—	—	—	—	—
q4	—	—	—	—	—	—	—	—	—
q5	—	—	—	—	—	—	q6	—	—
q6	—	—	—	—	—	—	—	—	—
q7	—	—	—	—	—	—	q8	—	—
q8	—	—	—	—	—	—	—	—	—
q9	—	—	—	—	—	q10	—	—	—
q10	—	—	q13	—	—	—	q11	q15	—
q11	—	—	—	—	—	—	—	—	q12
q12	—	—	—	—	—	—	—	—	—
q13	q14	—	—	—	—	—	—	—	—
q14	—	—	—	—	—	—	—	—	—
q15	—	—	q16	—	—	—	—	—	—
q16	—	—	—	—	—	—	—	—	—
q17	—	—	—	—	—	—	q18	—	—
q18	q21	—	q19	—	—	—	—	—	—
q19	—	—	—	—	q20	—	—	—	—
q20	—	—	—	—	—	—	—	—	—
q21	—	—	—	—	q22	—	—	—	—
q22	—	—	—	—	—	—	—	—	—
qt	qt	qt	qt	qt	qt	qt	qt	qt	qt

Accepting States:

- **q4** — Lobby
- **q6** — Laboratory
- **q8** — Research Wing
- **q12** — Server Room
- **q14** — Executive Room
- **q16** — Data Center
- **q20** — Admin Room
- **q22** — Vault

Trap State: qt *(Any transition not listed above can be treated as going to the trap state).*

1. Step-by-step Authentication By Java Code:

```
import java.util.*;

public class SmartBuildingDFA_Interactive {
    static String startState = "q0";
    static String trapState = "qt";

    static Map<String, Map<String, String>> transitions = new HashMap<>();
    static Map<String, String> zoneAcceptingStates = new HashMap<>();

    public static void main(String[] args) {
        buildTransitionTable();

        Scanner sc = new Scanner(System.in);
        String currentState = startState;

        int step = 1;
        boolean trap = false;

        while (step <= 4) {
            System.out.print("Enter authentication step " + step + ": ");
            String symbol = sc.nextLine().trim();

            currentState = nextState(currentState, symbol);
            if (currentState.equals(trapState)) {
                System.out.println("Wrong authentication! Access Denied.");
                trap = true;
                break;
            } else {
                System.out.println("Step " + step + " authentication successful.");
            }
            step++;
        }
```

```java
        if (!trap && zoneAcceptingStates.containsKey(currentState)) {
            System.out.println("Access Granted → Zone: " + zoneAcceptingStates.get(currentState));
        } else if (!trap) {
            System.out.println("Authentication sequence incomplete or invalid. Access Denied.");
        }
    }

    private static void buildTransitionTable() {
        // Lobby
        add("q0", "C", "q1");
        add("q1", "P", "q2");
        add("q2", "Fc", "q3");
        add("q3", "V", "q4");
        zoneAcceptingStates.put("q4", "Lobby");

        // Laboratory
        add("q2", "B", "q5");
        add("q5", "R", "q6");
        zoneAcceptingStates.put("q6", "Laboratory");

        // Research Wing
        add("q2", "V", "q7");
        add("q7", "R", "q8");
        zoneAcceptingStates.put("q8", "Research Wing");

        // Server Room
        add("q0", "P", "q9");
        add("q9", "F", "q10");
        add("q10", "R", "q11");
        add("q11", "A", "q12");
        zoneAcceptingStates.put("q12", "Server Room");

        // Executive Room
        add("q10", "Fc", "q13");
        add("q13", "C", "q14");
        zoneAcceptingStates.put("q14", "Executive Room");

        // Data Center
        add("q10", "R2", "q15");
        add("q15", "Fc", "q16");
        zoneAcceptingStates.put("q16", "Data Center");

        // Admin Room
        add("q0", "A", "q17");
        add("q17", "R", "q18");
        add("q18", "Fc", "q19");
        add("q19", "B", "q20");
        zoneAcceptingStates.put("q20", "Admin Room");

        // Vault
        add("q18", "C", "q21");
```

```
        add("q21", "B", "q22");
        zoneAcceptingStates.put("q22", "Vault");
    }

    private static void add(String from, String input, String to) {
        transitions.putIfAbsent(from, new HashMap<>());
        transitions.get(from).put(input, to);
    }
  private static String nextState(String current, String input) {
        return transitions.getOrDefault(current, new HashMap<>())
                     .getOrDefault(input, trapState);
    }
}
```

2. Output:

```
Output                                                          Clear
Enter authentication step 1: C
? Step 1 authentication successful.
Enter authentication step 2: P
? Step 2 authentication successful.
Enter authentication step 3: Fc
? Step 3 authentication successful.
Enter authentication step 4: V
? Step 4 authentication successful.
? Access Granted ? Zone: Lobby

=== Code Execution Successful ===
```

Figure: Lobby Access (Input: C → P → Fc → V→ Access Granted)

```
Output                                                          Clear
Enter authentication step 1: P
? Step 1 authentication successful.
Enter authentication step 2: F
? Step 2 authentication successful.
Enter authentication step 3: R
? Step 3 authentication successful.
Enter authentication step 4: A
? Step 4 authentication successful.
? Access Granted ? Zone: Server Room

=== Code Execution Successful ===
```

Figure: Server Room Access (Input: P → F → R → A→ Access Granted)

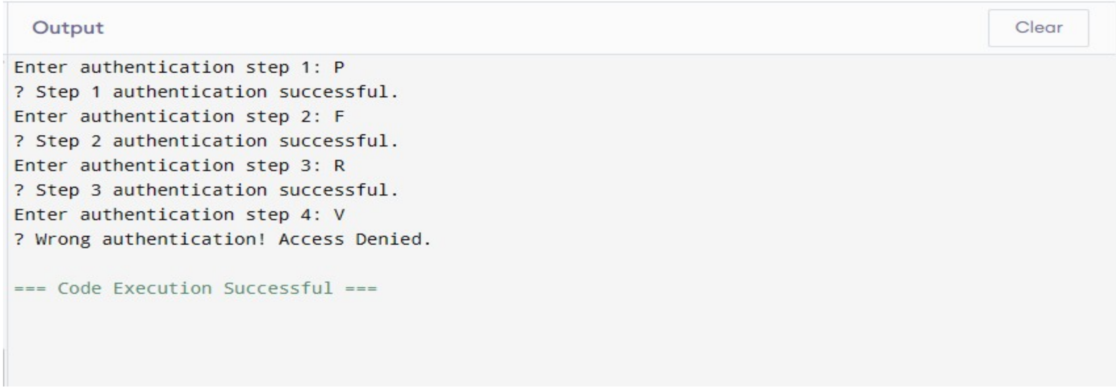


Figure: Server Room Access (Input: P → F → R → V→ Access Denied)

3. Testing and Result:

Test Case	Input Sequence	Expected Output	Actual Output	Result
1	C → P → Fc → V	Access Granted → Lobby	Access Granted → Lobby	Pass
2	P → F → R → A	Access Granted → Server Room	Access Granted → Server Room	Pass
3	P → F → R → V	Access Denied	Access Denied	Pass
4	A → R → Fc → B	Access Granted → Admin Room	Access Granted → Admin Room	Pass
5	A → R → C → B	Access Granted → Vault	Access Granted → Vault	Pass

4. Conclusion:

This DFA-based access control system effectively controls building access through multiple steps authentication. The customized version demonstrates how easily the system can be scaled to include new zones and authentication sequences, making it adaptable for future use cases.

CEP Justification:

Complex Engineering Criteria	Knowledge Profile	Justification by mentioning Section number/ line number of code/chapter number
	K2	I applied DFA theory while defining the input symbols (like C, F, R, etc.) and the states **(q0–q22).** These were taken directly from the course concepts and then used in my design and transition table.
P1	**K3**	I designed the DFA transitions in a way that ensures each zone has its own unique authentication sequence. Any wrong or out-of-order input immediately goes to the trap state **qt**. In my Java code, this part is handled through the next State method.
	K4	I tested the DFA thoroughly to ensure all valid sequences reach their correct zones and any wrong input immediately goes to the trap state. This confirms that the system works as intended and handles all cases reliably.
P2		I designed the system logic to handle multiple sequences efficiently, ensuring each step follows the required order. All transitions are clearly defined to prevent errors and maintain smooth operation.
P3		I verified that all components work together as intended and that errors are handled appropriately. Testing confirmed that the system behaves consistently in all scenarios.

Assignment Assessment Rubric (PO(b) – Problem Analysis)

Criteria	Excellent	Proficient	Adequate	Limited	Insufficient
Application of 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 awarded if insufficient evidence is presented or irrelevant discussion is included.
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 abstract thinking and in-depth analysis.	Proposed solution reflects very little abstract thinking and in-depth analysis.	—
Addressing Conflicting Requirements (20%)	Proposed solution addresses all conflicting requirements.	Proposed solution addresses most conflicting requirements.	Proposed solution addresses some conflicting requirements.	Proposed solution addresses very few conflicting requirements.	—
DFA Solution Design (30%)	DFA solution design satisfies all mentioned constraints.	DFA solution design satisfies most mentioned constraints.	DFA solution design satisfies some mentioned constraints.	DFA solution design satisfies very few mentioned constraints.	—
Documentation / Demonstration (20%)	Student conveys knowledge and analytical skills with exceptional approach; logically structured.	Student conveys knowledge and analytical skills with moderate approach; logically structured.	Student conveys knowledge and analytical skills with average approach; logic/flow sometimes unclear.	Student conveys knowledge and analytical skills with poor approach; logic/flow largely unrecoverable.	—