**National University of Computer and Emerging Sciences, Karachi**  
**FAST School of Computing – Spring 2025**  
**Course: Theory of Automata (CS3005)**

**Project Proposal: NFA Simulator**

**1. Project Overview**

**Project Title:** NFA Simulator  
**Course:** Theory of Automata (CS3005)  
**Instructor:** *Miss Bakhtawer Abbasi*

**2. Group Members**

* **Mohid Raheel Khan (23k-3000)**
* **Aaqib Shivji (23k-0625)**
* **Hassan Nafees (23k-0769)**

**3. Project Description**

We are developing a Non-deterministic Finite Automaton (NFA) simulator using C++. The simulator will allow users to define NFA components such as states, input symbols, transitions (including epsilon transitions), start state, and accepting states. The core functionality will determine whether an input string is accepted by the given NFA.

The simulator handles non-determinism and epsilon-closure computation and supports step-by-step simulation to visualize how the NFA processes an input string.

**Objectives**

* Build a fully functional NFA simulator from scratch using C++.
* Allow manual or code-defined NFA creation.
* Simulate and trace NFA transitions for given input strings, including epsilon transitions.

**Scope of the Project**

**Concepts Covered:**

* NFA construction
* Epsilon transitions and closure
* Simulation of multiple paths (non-determinism)
* Acceptance state checking

**Expected Outcomes:**

* Input-driven NFA simulation with clear output
* Terminal feedback on whether a string is accepted
* A tool that aids students in learning NFA behavior

**Functional Features**

* **Feature 1:** User-defined or code-initialized NFA with all required elements
* **Feature 2:** Support for non-deterministic transitions and epsilon transitions
* **Feature 3:** Input simulation with result: accepted or rejected

**4. Methodology**

We will build the entire simulator from scratch using C++, focusing on basic data structures such as hash maps, vectors/Arrays, graphs and sets. We will implement the core logic for transitions and simulate the processing of input strings via epsilon closure and non-deterministic state tracking.

**Tools and Technologies Used**

* Visual Studio Code (VSC)
* C++
* Standard Template Library (STL)

**5. Team Contributions**

**Aaqib Shivji:**

* Focused on optimizing transition handling and edge-case testing.
* Contributed to refining the NFA logic and improving code readability.
* Assisted in verifying correctness across various input strings.

**Mohid Raheel Khan:**

* Handled input simulation and acceptance checking logic.
* Helped with debugging and refining the core NFA flow.

**Hassan Nafees:**

* Designed and integrated the complete simulator structure.
* Managed testing, main program, and final integration.

*All members contributed equally. Whoever identified a problem implemented their solution directly to ensure shared understanding and productivity.*

**6. Challenges and Risks**

* **Handling state explosion** due to non-determinism.
* **Correctly implementing epsilon closures** for multiple states.
* **Ensuring robust input handling** and avoiding crashes or incorrect transitions.

To mitigate these risks, we relied on efficient data structures from STL, modular coding practices, and continuous testing after each implementation phase.

**7. References**

* Hopcroft, Motwani, Ullman – *Introduction to Automata Theory, Languages, and Computation*
* Introduction-to-computer-theory-by-Cohen
* C++ STL Documentation
* Online resources and tutorials on NFAs and epsilon closures

**8. Conclusion**

This project enhances our understanding of theoretical automata by translating NFA principles into a functional simulator. It provides a practical approach to exploring the behavior of NFAs, including their non-deterministic and epsilon-transition handling. In the future, we plan to extend this project to support regular expression parsing, graphical UI, and conversion from NFA to DFA.