**Subject-based project**

**Theory of automata and formal languages**

**Project Title:**

Simple chat filter(remove offensive words from chat)

**Abstract:**

This project presents the design of a finite automaton (FA) that identifies and filters offensive words from chat messages. The automaton recognizes specific forbidden words using deterministic transitions and halts in an accepting state when an offensive word is detected. This project demonstrates the real-world application of finite automata in moderation systems, content filtering, and text validation tasks.

**Purpose:**

The purpose of this project is to use a formal model (Finite Automaton) to recognize certain offensive words from a chat input. This strengthens the understanding of DFA design while showing its practical use in automatic moderation and real-time content checking systems.

**Applications:**

* **Online Chat Moderation**: Auto-detect offensive terms in chats or comments
* **Social Media Platforms**: Content filtering to maintain community guidelines
* **Gaming Chats**: Detect and block abusive words
* **SMS/Email Filters**: Block inappropriate language
* **Customer Support Chatbots**: Reject messages with bad language

**Description:**

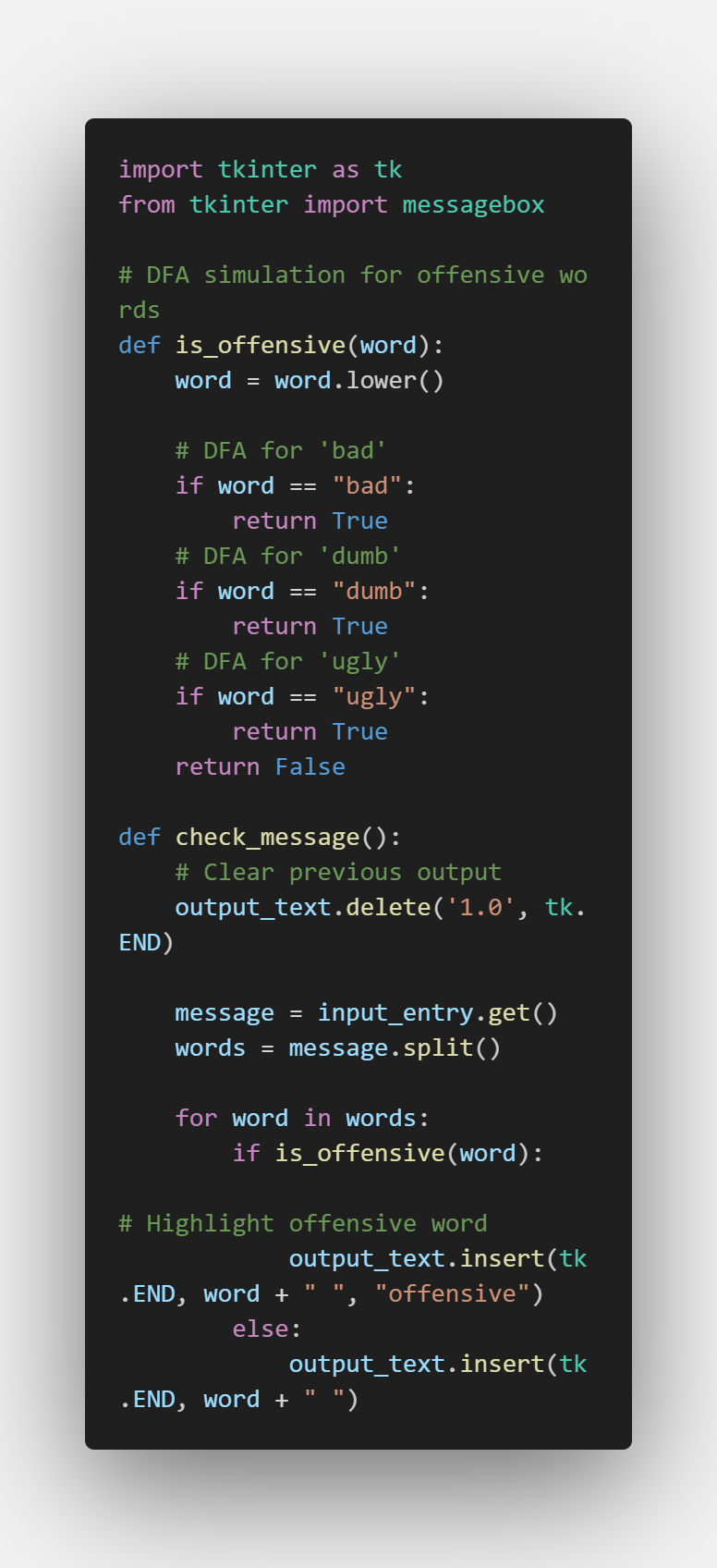
* **Input alphabet (Σ)**: {a–z}
* **Language (L)**: Set of strings that match exactly one of the following offensive words:  
  bad, mad, sad, dumb, fool, lame, hate, ugly
* **FA Type**: Deterministic Finite Automaton (DFA)
* **Working**:
  + Start at initial state (q0).
  + Read characters sequentially.
  + Follow specific transitions for each offensive word.
  + Reach accepting states if a full offensive word is matched.
  + Any wrong character causes transition to a dead state (q\_dead).

**DFA Simulation:**

**Steps to Simulate a DFA:**

1. **Start** at the **initial state** (usually called q0).
2. **Read the input string one symbol at a time.**
3. **Follow the transition** corresponding to the current symbol from the current state.
4. **Move to the next state** as defined by the transition.
5. **Repeat** this for all symbols in the input string.
6. **After the input ends:**
   * **If** you are in an **accepting state** → the input is **accepted**.
   * **If not** → the input is **rejected**.

**code:**

****

