Rules for Drawing UML Sequence Diagram

1. **Define the Scope and Context**:
   - Clearly specify the scenario or process being modeled (e.g., "User Login Process").
   - Identify the specific goal or interaction to keep the diagram focused on a single use case or functionality.
2. **Identify Participants (Objects/Actors)**:
   - Represent participants as **stick figures** at the top of the diagram, labeled with their role or object name (e.g., "User," "System," "Database").
   - Place participants in a logical order, typically from left to right, based on interaction flow.
3. **Draw Lifelines**:
   - Extend a **dashed vertical line** (lifeline) downward from each participant to represent their existence over time.
   - Ensure lifelines span the duration of the interaction being modeled.
4. **Show Message Exchanges**:
   - Represent messages as **arrows** between lifelines, indicating interactions:
     - **Solid arrow with filled head**: Synchronous message (caller waits for response, e.g., function call).
     - **Solid arrow with open head**: Asynchronous message (caller does not wait, e.g., notification).
     - **Dashed arrow**: Return message (response to a synchronous call, optional for clarity).
   - Label messages with clear, concise descriptions of the action or data exchanged (e.g., "submitCredentials()", "returnAuthToken").
5. **Use Activation Bars**:
   - Draw **thin rectangles** (activation bars) on lifelines to show when a participant is actively processing a message.
   - Ensure activation bars align with the start and end of synchronous message processing.
6. **Incorporate Time Progression**:
   - Arrange messages from **top to bottom** to reflect the chronological order of interactions.
   - Ensure the vertical axis represents time, with earlier events at the top and later ones below.

7. **Handle Control Structures (Optional)**:
    - Use **combined fragments** (rectangles with labels) to represent control flows:
        - **alt**: Alternatives (e.g., "if-else" conditions, with guards like "[login successful]").
        - **opt**: Optional interactions (executed if a condition is met).
        - **loop**: Repeated interactions (e.g., "loop [until valid input]").
    - Label conditions clearly within square brackets (e.g., "[valid credentials]").
8. **Include Exceptions and Alternatives**:
    - Model exceptions (e.g., "login fails") using **alt** fragments with conditions (e.g., "[invalid credentials]").
    - Show alternative paths or error handling as separate message flows within the fragment.
9. **Maintain Simplicity**:
    - Focus on **key interactions** for the scenario, avoiding excessive details like internal processing or minor messages.
10. **Ensure Consistency with Requirements**:
    - Align the diagram with functional requirements from the requirements engineering process (e.g., services like "Authenticate User").
    - Verify that actors and objects match stakeholders and system components identified in elicitation.