Objective:

Develop a simplified Ticket Management System similar to Jira that allows users to manage various types of tickets such as stories, epics, and on-call tasks. Implement functionalities to create, update, and manage these tickets, including sprint management for stories.

Requirements:

1. **Ticket Types:**
    - Implement three types of tickets: Story, Epic, and On-call.
    - Each ticket type should have a unique flow of statuses.
2. **Ticket Flows:**
    - **Story:** `Open` -> `In Progress` -> `Testing` -> `In Review` -> `Deployed`
    - **Epic:** `Open` -> `In Progress` -> `Completed`
    - **On-call:** `Open` -> `In Progress` -> `Resolved`
3. **Sprint Management:**
    - Only tickets of type `Story` can be part of a sprint.
    - Users should be able to add and remove stories from a sprint.
    - Maintain a list of stories in the current sprint.
4. **Sub-tasks:**
    - Each ticket can have multiple sub-tasks associated with it.
    - Sub-tasks should have the same status flow as their parent ticket.
    - Users should be able to create, update, and delete sub-tasks.
    - Story should not be closed if the sub tasks are not marked complete.
5. **Functional Requirements:**
    - **Create Ticket:** Allow users to create tickets of any type.
    - **Update Ticket Status:** Allow users to update the status of any ticket.
    - **Sprint Management:** Allow users to add/remove stories from the current sprint.
    - **Sub-task Management:** Allow users to add/remove sub-tasks for any ticket.
6. **Data Storage:**
    - Use an in-memory data structure to store tickets, sub-tasks, and sprint information.
    - Ensure the data structure supports efficient lookup and modification.

Good to Have Requirements:

1. Users should be able to add description, comments on the ticket
2. Handle scenarios where multiple people are modifying the same ticket

Example Scenario:

1. **Create Tickets:**
    - User creates a Story ticket with the title "Implement login feature".
    - User creates an Epic ticket with the title "User authentication".

    - User creates an On-call ticket with the title "Fix production bug".
2. **Update Ticket Status:**
    - User updates the status of "Implement login feature" from `Open` to `In Progress`.
3. **Sprint Management:**
    - User adds "Implement login feature" to the current sprint.
    - User removes "Implement login feature" from the current sprint.
4. **Sub-task Management:**
    - User creates a sub-task for "Implement login feature" with the title "Design login UI".
    - User updates the status of "Design login UI" from `Open` to `In Progress`.
    - User deletes the sub-task "Design login UI".

Evaluation Criteria:

- **Correctness:** The solution should correctly implement the required functionalities.
- **Code Quality:** The code should be clean, well-organized, and easy to understand. Follow OO principles.
- **Code Extensibility**: Code should be modular & extensible
- **Efficiency:** The solution should handle operations efficiently, considering edge cases and potential large data sets.
- **Completeness:** The solution should cover all aspects of the requirements, including ticket creation, status updates, sub-task management, and sprint management.

Notes:

- You can use any programming language of your choice.
- Focus on the core functionality first before adding any additional features.
- Write unit tests/utility to demonstrate the correctness of your implementation.