Cheat Sheet: Use Case (UC) Guidelines

#	Guideline	Description
1	**Global Entry Point for UC Model**	Create a global summary use case which serves as an entry point to the use case model. Organize the use case model as a hierarchical story that can either be unfolded to get more detail or folded up to hide detail and show more context.
2	**Avoid Duplication**	Avoid duplication and redundant information by factoring out common behavior into sub use cases using «includes».
3	**Group Similar Extensions**	Combine as many extensions as possible without sacrificing completeness. Strive for a layer of abstraction that will integrate similar alternative flows.
4	**Consistency**	Use consistent actor, object and action names throughout the use cases.
5	**Separation of Concerns**	Use references whenever possible. Define details about actors, domain objects, business rules, user messages, data descriptions, non-functional requirements (ex. performance) outside the use cases.
6	**Change History**	Track and document the evolution of the use cases through versioning, annotations or tool support.
7	**Goal Orientation**	Ensure that the use case (with the exception of the Global Entry Point) addresses *one* well-defined goal of the primary actor. The use case is named with an active verb phrase representing that goal.
8	**Two Endings**	Every use case has two possible endings: Success and Failure. Ensure that Failures of «includes» use cases are properly handled through an extension in the base use case.
9	**Main Success Scenario**	The normal flow of the use case is a simple scenario that leads to the fulfillment of the use case goal. It does not include failures or alternative interactions.
10	**Scenario Length**	The main success scenario (or any other scenario) consists of three to nine steps.
11	**Detectable by System**	All actions (performed by actor) and conditions are detectable by the system.
12	**Validation Failures**	Validation steps performed by the system should be complete by capturing both positive and negative outcomes.
13	**Explicitly State Actor or System**	Each use case step clearly shows who is performing the step (e.g., the primary actor, a secondary actor or the system).
14	**Leave UI Out**	Each use case step is written in a UI- and technology-independent manner.
15	**Writing style**	Use simple language. Write from a bird's eye view using present tense and active voice. Avoid negations, adverbs, adjectives, and synonyms.
16	**Keep away implementation details**	Do not mention design or implementation details in the use case (ex. function calls or database queries).
17	**Diagram complex flows and nontrivial cases**	If the flow is complicated, prepare a system sequence or activity diagram to make it easier to follow the text.
18	**Validate Conditions**	Have the system validate conditions. Do not use 'if' statements.