User Use Cases

Use Case: Create Snippet

Participating Actors: Initiated by *creator*
Entry Criteria: None
Exit Criteria: New Snippet created
Flow of Events:

1. *Creator* requests to create a new snippet.
2. *SnippetSystem* creates snippet and updates display with a new empty snippet.

Use Case: View Snippet

Participating Actors: Initiated by *viewer*
Entry Criteria: There is an existing Snippet
Exit Criteria: *Viewer* is now observing Snippet
Flow of Events:

1. *Viewer* requests to view an existing snippet (via its unique ID).
2. *SnippetSystem* updates display to show existing snippet with all comments.

Use Case: Update Snippet INFO

Participating Actors: Initiated by *creator*; observed by viewers
Entry Criteria: Snippet exists that was created by creator
Exit Criteria: That snippet's INFO has been updated
Flow of Events:

1. *Creator* requests to update the INFO associated with a snippet.
2. *SnippetSystem* updates snippet INFO and all viewers see the result.

Use Case: Update Snippet TEXT

Participating Actors: Initiated by a *viewer* or *creator*; observed by viewers
Entry Criteria: Snippet exists
Exit Criteria: Snippet TEXT is updated
Flow of Events:

1. *User* requests to update the text for given snippet.
2. *SnippetSystem* updates TEXT and all viewers see the result.

Use Case: Delete Snippet

Participating Actors: Initiated by *creator*; observed by viewers
Entry Criteria: Snippet exists that was created by creator
Exit Criteria: Snippet is removed together with all comments
Flow of Events:

1. *Creator* requests to delete a given snippet.
2. *SnippetSystem* removes snippet and alerts all viewers.

Use Case: Create Comment

Participating Actors: Initiated by a *viewer* or *creator*; observed by viewers
Entry Criteria: Snippet exists
Exit Criteria: A new comment is created for the snippet
Flow of Events:

1. *User* requests to create a comment for given snippet.
2. *SnippetSystem* creates comment for snippet and all viewers see the result.

Use Case: Delete Comment

Participating Actors: Initiated by a *viewer* or *creator*; observed by viewers
Entry Criteria: Snippet exists with at least one comment
Exit Criteria: Desired comment is removed from the snippet
Flow of Events:

1. *User* requests to delete a comment for given snippet.
2. *SnippetSystem* deletes comment for snippet and all viewers see the result.

Administrator Use Cases

Use Case: List Snippets

Participating Actors: Initiated by an *administrator*
Entry Criteria: None
Exit Criteria: None
Flow of Events:

1. *Administrator* requests to list all existing snippets.
2. *SnippetSystem* presents a listing of snippets with information organized in reverse order of creation

Use Case: Delete Snippet

Participating Actors: Initiated by an *administrator*
Entry Criteria: Desired snippet exists
Exit Criteria: Snippet removed from the system
Flow of Events:

1. *Administrator* requests to delete given snippet by its ID.
2. *SnippetSystem* deletes snippet and all viewers see the result.

Use Case: Remove Stale Snippets

Participating Actors: Initiated by an *administrator*
Entry Criteria: None
Exit Criteria: Snippets older than N days old are removed
Flow of Events:

1. *Administrator* requests to delete all snippets older than N days old.
2. *SnippetSystem* presents a listing of remaining snippets; all viewers see the result.

End Notes

1. When a creator or viewer has a snippet in front of them, is there a list of all concurrent users that are observing that code snippet? This *could* be done but I am not yet ready to confirm this as a requirement. This would affect **View Snippet** use case.
2. Note that viewers discover snippets in a way that is external to the system. There is no way to discover a snipper for regular viewers except by being given the snippets unique ID (from the creator).

3. Note that the **Delete Snippet** use case appears both for *creator* and for *administrator* so there is some duplication. The difference is that the creator can only self-delete the snippet that he created, whereas the administrator can delete any snippet in the system.
4. Each student team is responsible for crafting a storyboard for the expected visual behavior of the system (from the points of view of creator, viewer and administrator).

CHANGE LOG

1. Initial version September 22 2020 2:30 PM