Q1 [22 points]. Imagine adding the following functionality to the **SlidingPuzzleApp**, the case study completed in the Individual Project module.

*Upon solving the puzzle, a proud user can print out the sequence of moves that led to the solution*

Write a use case that captures this functionality using the structured template shown in class.

(+2) Name: Print Solution (+2)

(+2) Participating Actor : Initiated by user (+2)

(+2) Entry Condition: User has solved the puzzle (+2)

(+2) Exit Condition: None or Result Printed (+1)

(+2) Flow Of Events:

1. (+1) user requests to (+1) print solution (+1 numbered)
2. (+1) SlidingPuzzleApp (+1) prints the solution to the console

Grade is recorded as

2,2 Name/UseCase Print Solution (accept as 4 points if clear this is the name)

2,2 Participating Actor Initialed by User (only +1 if they use Player)

2,2 Entry Condition User has Solved Puzzle (only +1 if use Game instead of Puzzle)

2,1 Exit Condition None or Solution Printed

2 Flow of Events

3 Numbering (not bullets) +1 [user requests] +1 [to print solution]

2 +1 SlidingPuzzleApp (note: can’t just be System or App or It)  
 +1 [prints the solution to the console]

Note: Only take off 1 point for PLAYER instead of user in the participating actor. Do not take off multiple times.

Q2 Sample graded for this question regarding UML Class diagrams

Classes  
------  
+1 Server (or Discord or DiscordServer)  
+1 Channel

+1 Participant

+1 Post

Attributes

----------

+1 Server.name

+1 Server.participants

+1 Server.channels

+1 Channel.name

+1 Channel.posts

+1 Participant.name

+1 Participant.description

+1 [0..1] for optional

+1 Post.message

+1 Post.participant

+1 Post.date

Arrows

-------

+1 Server -> Participant

+1 (1 .. \*)

+1 Post -> Participant

+1 Server -> Channel

+1 (1 .. \*)

+1 Channel -> Post

+1 (1 .. \*)

==================================

total of 22 points

Q3. Sample EBC resolution

A picture containing text

Description automatically generated

Q4. Party API design

There should be three resource entry points. Note names are subjective and should align

with the use case names.

**Create PartyList**

/create POST { "items" : "comma,separated,list,of,things"}

Alternatively, could have added structure for the input items

/create POST { "item" : [ { "item" : "large pizze" }, {"item" : "folding table"}} ] }

200 SUCCESS You could return {full-list} as I show below, or return nothing. Either is fine

400 FAILURE

**Show PartyList**

/show GET // no need for payload

200 SUCCESS You must return {full-list} as below:

400 FAILURE

full-list: { "items": [

{ "item" : "1. Two Large Cheese pizzas" },

{ "item" : "2. Two large Veggie pizzas", "person" : "Taylor Swift"}

...

]}

**Claim Item**

/claim POST { "item" : "1. Two Large Cheese pizzs", "name" : "George Heineman" }

200 SUCCESS You could return {full-list} or nothing

400 FAILURE