|  |  |
| --- | --- |
| Use Case Name | OfflineActionQueue |
| Participating Actors | User |
| Goal | System saves actions |
| Trigger | User commits action while offline |
| Precondition | System has no connection  User has committed action via other Use Case |
| Postcondition | On commit, user is informed action was queued |
| Basic Flow | 1. User commits action 2. System detects lack of connection 3. System sends alert box of impossibility and requests permission to queue (auto-execute) 4. User grants permission 5. System adds action to queue |
| Exceptions | 4 If user doesn’t grant permission  4.1.1 User rejects notification  4.1.2 Action is not queued  4.1.3 Action is saved for User’s later consideration |
| Qualities | Require minimal but notable input |
| Constraint | Queue should be able to hold any number of actions |
| Includes | Creating an inventory item use case  Proposing trade UC |
| Extends |  |
| Related Artifacts | US09.01.01, US09.02.01 |
| Notes | Saved(4.1.3) actions are similar to counter proposal files |
| Open Issues | US01.01.01, US03.01.01, will be answered by use cases referenced by this UC |
|  |  |

Glossary

Action refers to sequences of events detailed by other use cases, which result in changes. (creating an inventory, proposing trades)

Execute refers to actions be done

Commit refers to end of other use cases where the action, would execute if the app had a connection.

Test

User user = new User(“userman”);

system.setConnect(Boolean.False);

Action addItem = new Item(“Starbucks”, 20);

user.addTestItem(addItem);

assertTrue(user.peekQueue==addItem);