## Use Case Specifications

**Use case name:** Register countries as teams and individuals as athletes

**Use case purpose:** This use case provides the Winter Olympics with a set of countries that will compete as well as the set of athletes that will represent each team

**Actor:** Official Registrant of Winter Olympics

**Optimistic flow:**

1. Master file with countries, their respective individuals, and the individual’s data has been given to the Winter Olympics by the pre-Olympics committee
2. Actor logs into the system
3. Actor is given the option to read the master file
4. Actor reads master file
   1. Actor checks the file and ensures the following are present
      1. The twelve teams competing
      2. Each athlete in their respective team
         1. First and last name
         2. Gender
         3. Events in which to participate
   2. Actor confirms that the file they have selected is the correct file for the database
   3. Actor has database populated with the contents of the file
5. Actor given choice to modify and/or add information into the database that is not currently in the master file
6. If insertion was successful and no further modification needed, actor’s job is complete
7. Actor logs out of the system

**Pragmatic flow:**

**Condition triggering alternate flow:**

**Condition 1:** The information stored does not accurately correspond to the actual teams and/or individuals to be competing in the Winter Olympics

1. Master file that has been provided via the official qualification rounds has been updated by the appropriate pre-Olympics committee
2. Actor logs into the system
3. Actor is given the option to read the newly updated master file
4. Actor reads master file
   1. Actor assesses the file
      1. Locates and recognizes the differences
      2. Ensures the correction(s) are valid
   2. Actor confirms that the file they have selected is the correct file for the database
   3. Actor has the database overridden with the newly updated master file
5. Actor given choice to modify and/or add information into the database that is not currently in the master file
   1. If modification needed, actor modifies database directly
   2. Actor confirms that the newly modified database is correct
   3. If more modification is needed, go to step 4
6. If insertion was not successful, go to step 2
7. If insertion was successful, and no further modification needed, actor’s job is complete
8. Actor logs out of the system

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use case name:** Allocate events to a specific rink with time and day

**Use case purpose:** This use case takes the predetermined events of the Winter Olympics and assigns each event to an available rink throughout the two-week duration

**Actor:** Official Scheduler of Winter Olympics

**Optimistic flow:**

1. Actor logs into the system
2. Actor views the list of events to be held on a particular day
3. Actor views list of rinks to ensure no rinks have been assigned an event
   1. Allocate a specific event to a specific rink
   2. Allocate a specific time and day for said rink
   3. Actor confirms that each event is allocated to the appropriate rink
   4. Actor confirms that each event contains the correct time
   5. If more events need allocated, go to step C-a
   6. Actor confirms the complete allocation is correct
4. If allocation was successful, actor’s job is complete
5. Actor logs out of the system

**Pragmatic flow:**

**Conditions triggering alternate flow:**

**Condition 1:** Two or more events are held on the same rink at the same time

1. Start at step A and proceed until step C
   1. One or more rinks contain overlapping events
      1. Actor changes the allocation by moving one of the events to an open rink
      2. Actor is prompted to confirm change
      3. If additional modification is needed, go to step 1-a
2. If no additional changes are required, actor’s job is complete
3. Actor logs out of the system

**Condition 2:** Event is held on the wrong date

1. Start at step A and proceed until step C
   1. One or more events are held on the wrong date
      1. Actor changes the allocation by swapping the problem event with the event that is supposed to occur on the date at hand
      2. Actor is prompted to confirm change
      3. If additional modification is needed, go to step 1-a
2. If no additional changes are required, actor’s job is complete
3. Actor logs out of the system

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use case name:** Allocate athletes to each event

**Use case purpose:** This use case takes the registered athletes of the Winter Olympics and assigns each athlete to their assigned event on a particular day

**Actor:** Official Scheduler of Winter Olympics

**Optimistic flow:**

1. Actor logs into the system
2. Actor views list of organized events held on a particular day
3. Actor also views list of athletes to be participating in the events taking place on the particular day
   1. Actor allocates an athlete to their respective event
   2. Confirmation is required and actor is prompted to confirm
   3. If more athletes need to be assigned to an event, go to step B
   4. Actor confirms allocation of all athletes for the particular day is correct
4. If allocation was successful, actor’s job is complete
5. Actor logs out of the system

**Pragmatic flow:**

**Conditions triggering alternate flow:**

**Condition 1:** Athlete is placed into the wrong event

1. Start at step A and proceed until step C
   1. Actor views current allocation of athletes to events
   2. Extract athlete from wrong event and place into correct event
   3. Actor confirms change
   4. If more athletes need arranged, proceed to step 1-a
2. If allocation was successful, actor’s job is complete
3. Actor logs out of the system

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use case name:** Record scores/times from each event

**Use case purpose:** This use case takes the qualified judge’s recorded scores/times and those scores/times get validated and recorded by the head judge into the database

**Actor:** Official Head Judge of Winter Olympics

**Optimistic flow:**

1. Actor logs into the system
2. Actor views a particular event
   1. Actor views the judge’s scores/times for the particular event
   2. Actor verifies that the scores are valid
      1. For Figure Skating event, actor disregards highest and lowest score
         1. Actor averages the middle five scores and records the average
      2. For Speed Skating event, actor records the time taken to complete the race
   3. Actor confirms the entry of each score/time
   4. If more scores/times need validated and entered, proceed to step B
3. Actor confirms every recorded score/time as a whole
4. If no more validation or recording is required, actor’s job is complete
5. Actor logs out of the system

**Pragmatic flow:**

**Conditions triggering alternate flow:**

**Condition 1:** Wrong scores/times are entered into the database

1. Actors logs into the system
2. Actor views the database with the scores/times that have been recorded
   1. Actor locates the event with the incorrect datum/data
      1. Review of judge’s official scores for the particular event is checked by the actor
         1. Recalculation of official scores done by actor to see if different result occurs
      2. If calculation error, actor enters in corrected datum/data to database
      3. Actor confirms any change
      4. If more revision is desired, proceed to step 2
3. Actor confirms every correction/change as a whole
4. If no more corrections/changes needed, actor’s job is complete
5. Actor logs out of the system

**Condition 2:** Wrong scores/times are entered via an official judge

1. Start at step A and proceed until step B-b
   1. If scores/times are invalid, have official judges re-enter their scores/times
   2. Actor confirms and validates the new scores/times
   3. If more validation is needed, proceed to step B-b
2. If no more invalid scores/times, Proceed to step B-i

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use case name:** Display event information, scores, current standings, and medal count

**Use case purpose:** This use case utilizes the data stored into the database and displays the events being held on what rink and when, athletes who are competing in said events, current scores and current standings

**Actor:** Official of Winter Olympics

**Optimistic flow:**

1. Actor logs into system
2. Actor accesses database
   1. Actor looks over all information in database
      1. Allocation of events
      2. Allocation of athletes
      3. Assignment of judges
      4. Scores/times for each event
   2. Actor confirms that all information to be displayed is present
3. Actor makes information, scores, and current standings available for public view
4. If no other information needs displayed, actor’s job is complete
5. Actor logs out of the system

**Pragmatic flow:**

**Conditions triggering alternate flow:**

**Condition 1:** Display information is not in concord with the official information

1. Start at step A and proceed until step B-a-iv
   1. Actor locates a discrepancy
      1. Actor confers with Official Scheduler if wrong scheduling information is displayed
         1. Actor waits until Official Scheduler has corrected the issue in the database
         2. Actor confirms the change in the database
         3. Actor displays the corrected information
      2. Actor confers with Official Head Judge if wrong scoring/timing information is displayed
         1. Actor wais until Official Head Judge has corrected the issue in the database
         2. Actor confirms the change in the database
         3. Actor displays the corrected information
2. If there are any more discrepancies, proceed to step 1-a
3. Actor confirms all changes made as a whole
4. If no more information needs corrected, actor’s job is complete
5. Actor logs out of the system