# Use Cases

for

# <Sportify>

**Version 1.0 approved**

**Prepared by <Ding Ren>**

**<Haagen Daz>**

**<8/02/24>**

## Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| | | | |
| | | | |

# 1. Guidance for Use Case Template

Document each use case using the template shown in the Appendix. This section provides a description of each section in the use case template.

# 1. Use Case Identification

## 1.1. Use Case ID

Give each use case a unique numeric identifier, in hierarchical form: X.Y. Related use cases can be grouped in the hierarchy. Functional requirements can be traced back to a labeled use case.

## 1.2. Use Case Name

State a concise, results-oriented name for the use case. These reflect the tasks the user needs to be able to accomplish using the system. Include an action verb and a noun. Some examples:

- View part number information.
- Manually mark hypertext source and establish link to target.
- Place an order for a CD with the updated software version.

## 1.3. Use Case History

### 1.3.1 Created By

Supply the name of the person who initially documented this use case.

### 1.3.2 Date Created

Enter the date on which the use case was initially documented.

### 1.3.3 Last Updated By

Supply the name of the person who performed the most recent update to the use case description.

### 1.3.4 Date Last Updated

Enter the date on which the use case was most recently updated.

# 2. Use Case Definition

## 2.1. Actor

An actor is a person or other entity external to the software system being specified who interacts with the system and performs use cases to accomplish tasks. Different actors often correspond to different user classes, or roles, identified from the customer community that will use the product. Name the actor(s) that will be performing this use case.

## 2.2. Description

Provide a brief description of the reason for and outcome of this use case, or a high-level description of the sequence of actions and the outcome of executing the use case.

## 2.3. Preconditions

List any activities that must take place, or any conditions that must be true, before the use case can be started. Number each precondition. Examples:

2. User's identity has been authenticated.
3. User's computer has sufficient free memory available to launch task.

## 2.4. Postconditions

Describe the state of the system at the conclusion of the use case execution. Number each postcondition. Examples:

1. Document contains only valid SGML tags.
2. Price of item in database has been updated with new value.

## 2.5. Priority

Indicate the relative priority of implementing the functionality required to allow this use case to be executed. The priority scheme used must be the same as that used in the software requirements specification.

## 2.6. Frequency of Use

Estimate the number of times this use case will be performed by the actors per some appropriate unit of time.

## 2.7. Flow of Events

Provide a detailed description of the user actions and system responses that will take place during execution of the use case under normal, expected conditions. This dialog sequence will ultimately lead to accomplishing the goal stated in the use case name and description. This description may be written as an answer to the hypothetical question, "How do I <accomplish the task stated in the use case name>?" This is best done as a numbered list of actions performed by the actor, alternating with responses provided by the system.

## 2.8. Alternative Flows

Document other, legitimate usage scenarios that can take place within this use case separately in this section. State the alternative course, and describe any differences in the sequence of steps that take place. Number each alternative course using the Use Case ID as a prefix, followed by "AC" to indicate "Alternative Course". Example: X.Y.AC.1.

## 2.9. Exceptions

Describe any anticipated error conditions that could occur during execution of the use case, and define how the system is to respond to those conditions. Also, describe how the system is to respond if the use case execution fails for some unanticipated reason. Number each exception using the Use Case ID as a prefix, followed by “EX” to indicate “Exception”. Example: X.Y.EX.1.

## 2.10. Includes

List any other use cases that are included (“called”) by this use case. Common functionality that appears in multiple use cases can be split out into a separate use case that is included by the ones that need that common functionality.

## 2.11. Special Requirements

Identify any additional requirements, such as nonfunctional requirements, for the use case that may need to be addressed during design or implementation. These may include performance requirements or other quality attributes.

## 2.12. Assumptions

List any assumptions that were made in the analysis that led to accepting this use case into the product description and writing the use case description.

## 2.13. Notes and Issues

List any additional comments about this use case or any remaining open issues or TBDs (To Be Determineds) that must be resolved. Identify who will resolve each issue, the due date, and what the resolution ultimately is.

# Use Cases

| Use Case ID: | UC01 | | |
|---|---|---|---|
| Use Case Name: | Calculate Distances | | |
| Created By: | Ding Ren | Last Updated By: | Ding Ren |
| Date Created: | 8/2/24 | Date Last Updated: | 8/2/24 |

| Actor: | Google Maps API |
|---|---|
| Description: | Calculates distances from user-specified location to every sports location in a certain circular radius using a specific mode of transport, which are set by the user. |
| Preconditions: | 1. The circular radius for which sport facilities are considered must be set.<br>2. Mode of transport must be set. |
| Postconditions: | |
| Priority: | |
| Frequency of Use: | 1. Everytime a new query is made.<br>2. Everytime the search radius changes.<br>3. Everytime the user-specified location changes.<br>4. Everytime mode of transport changes. |
| Flow of Events: | 1. Based on departure location, Sportify sets the circular radius of xkm.<br>2. Sportify adds every sports facility within the radius into a list.<br>3. Sportify calls the Google Maps API.<br>4. The Google Maps API calculates the distance based on the selected mode of transport between user departure location and every sports location in the list.<br>5. The Google Maps API returns all the calculated information to the System.<br>6. Sportify returns the list of results for each sports location. |
| Alternative Flows: | |
| Exceptions: | |
| Includes: | |
| Special Requirements: | API response time should be 0.1s which can be counted as immediate response where users would not feel any interruption. |

| | |
|---|---|
| Assumptions: | There is a way to flag out whether sports facilities are in a particular radius. |
| Notes and Issues: | |

| Use Case ID: | UC02 | | |
|---|---|---|---|
| Use Case Name: | Get Weather Conditions | | |
| Created By: | Ding Ren | Last Updated By: | Ding Ren |
| Date Created: | 8/2/24 | Date Last Updated: | 8/2/24 |

| Actor: | 1. Rainfall API<br>2. Weather Forecast API<br>3. PSI API<br>4. UV API<br>5. Air temperature API |
|---|---|
| Description: | Obtains and collates weather conditions for the area of Singapore which the user-specified search radius is located. |
| Preconditions: | 1. User has entered location.<br>2. User has specified search-radius. |
| Postconditions: | |
| Priority: | |
| Frequency of Use: | 1. Everytime a new query is made.<br>2. Everytime the search radius changes.<br>3. Everytime the user-specified location changes. |
| Flow of Events: | 1. Sportify queries the APIs for each location to get the weather conditions.<br>2. The rainfall API returns rainfall amount in mm of that location.<br>3. The weather forecast API returns the temperature in degrees of that location.<br>4. The weather forecast API returns wind speed in km/h of that location.<br>5. The PSI API returns the PSI readings of that location.<br>6. The UV API returns the UV levels of that location. |
| Alternative Flows: | |
| Exceptions: | EX1: If any of the APIs does not return a result<br>1. Replace any of the weather quantities with NaN if that API does not return a result. |
| Includes: | |
| Special Requirements: | API response time should be 0.1s which can be counted as immediate response where users would not feel any interruption. |
| Assumptions: | |
| Notes and Issues: | |

| Use Case ID: | UC03 | | |
|---|---|---|---|
| Use Case Name: | Provide Court Vacancy Levels | | |
| Created By: | Ding Ren | Last Updated By: | Ding Ren |
| Date Created: | 8/2/24 | Date Last Updated: | 18/2/24 |

| Actor: | |
|---|---|
| Description: | Gives the vacancy levels of each sport location identified in the search radius, based on number of people checked-in and pre checked-in. |
| Preconditions: | |
| Postconditions: | |
| Priority: | |
| Frequency of Use: | 1. Everytime a new query is made.<br>2. Everytime the search radius changes.<br>3. Everytime the user-specified location change.<br>4. Periodically, every 1 sec. |
| Flow of Events: | 1. Based on departure location , Sportify sets search radius of xkm.<br>2. Sportify add every sports location within the radius into a list.<br>3. Sportify gets the live-count of number of people pre checked-in, checked-in and playing<br>4. Returns the list of results for each sports location |
| Alternative Flows: | |
| Exceptions: | |
| Includes: | |
| Special Requirements: | |
| Assumptions: | |
| Notes and Issues: | Vacancy levels are not used to calculate the scores used in ranking the various sports in the search radius. |

| Use Case ID: | UC04 | | |
|---|---|---|---|
| Use Case Name: | Provides Sports Facilities | | |
| Created By: | Ding Ren | Last Updated By: | Ding Ren |
| Date Created: | 8/2/24 | Date Last Updated: | 8/2/24 |

| Actor: | Google Maps API |
|---|---|
| Description: | Obtains a list of sports facilities that are within search-radius. |
| Preconditions: | 1. User specify location.<br>2. User specify search radius. |
| Postconditions: | |
| Priority: | |
| Frequency of Use: | 1. Everytime a new query is made.<br>2. Everytime the search radius changes.<br>3. Everytime the user-specified location change. |
| Flow of Events: | 1. Based on departure location, Sportify sets search radius of xkm.<br>2. Sportify call Google Maps API to find the list of sports facilities within the search radius<br>3. The Google Maps API returns the list of results for each sports location |
| Alternative Flows: | AF-S2: If no sports location within search radius, display error message<br>1. Sportify displays the message “ No sports location! Increase your radius size or change location!”<br>2. Sportify prompts the user to enter new location or search radius. |
| Exceptions: | |
| Includes: | |
| Special Requirements: | API response time should be 0.1s which can be counted as immediate response where users would not feel any interruption. |
| Assumptions: | Google Maps API must be up to date. |
| Notes and Issues: | |

| Use Case ID: | UC05 | | |
|---|---|---|---|
| Use Case Name: | Calculate Final Score | | |
| Created By: | Ding Ren | Last Updated By: | Ding Ren |
| Date Created: | 8/2/24 | Date Last Updated: | 8/2/24 |

| Actor: | |
|---|---|
| Description: | Calculates score for each sport in every sport location for Sportify to rank . |
| Preconditions: | |
| Postconditions: | |
| Priority: | |
| Frequency of Use: | 1. Everytime a new query is made.<br>2. Everytime the search radius changes.<br>3. Everytime the user-specified location change.<br>4. Everytime the mode of transport changes. |
| Flow of Events: | 1. Sportify uses the included use case Get Weather Conditions to take in weather conditions from each API.<br>2. Sportify uses the included use case Calculate Distances to get distances and busy levels from Google maps API.<br>3. Sportify generates a score based on the variables obtained for each sport facility location.<br>4. Sportify ranks each sport facility location based on the scores calculated. |
| Alternative Flows: | |
| Exceptions: | |
| Includes: | UC01, UC02 |
| Special Requirements: | |
| Assumptions: | |
| Notes and Issues: | |

| Use Case ID: | UC06 | | |
|---|---|---|---|
| Use Case Name: | Specify Search Criteria | | |
| Created By: | Randall | Last Updated By: | Randall |
| Date Created: | 8/2/24 | Date Last Updated: | 8/2/24 |

| Actor: | User, Google Maps API |
|---|---|
| Description: | User specifies the search radius and preferred mode of transport for Sportify to take in account. |
| Preconditions: | |
| Postconditions: | |
| Priority: | |
| Frequency of Use: | 1. Everytime a new query is made.<br>2. Everytime the search radius changes.<br>3. Everytime the user-specified location change.<br>4. Everytime the mode of transport changes. |
| Flow of Events: | 1. Sportify requests the input of a departure location, a specified radius and the preferred mode of transport.<br>2. User clicks the “select current location” button.<br>3. User enters a specified radius and a preferred mode of transport.<br>4. User chooses a location as the departure location. |
| Alternative Flows: | AF-S2: If user chooses to type in a departure location<br>1. The Google Maps API will give a drop-down list of search suggestions containing relevant locations.<br>2. User chooses a location as the departure location.<br>3. Sportify returns to step 3.<br>AF-S2: If user chooses to type in a departure location<br>1. No result is given by the Google API.<br>1. User should report the issue or start another search.<br>2. Sportify returns to step 1. |
| Exceptions: | |
| Includes: | |
| Special Requirements: | |
| Assumptions: | |
| Notes and Issues: | |

| Use Case ID: | UC07 | | |
|---|---|---|---|
| Use Case Name: | Display Top Recommendations | | |
| Created By: | Steven | Last Updated By: | Steven |
| Date Created: | 8/2/24 | Date Last Updated: | 8/2/24 |

| Actor: | User |
|---|---|
| Description: | Collates all sports facilities with their respective sports according to the final scores. Displays all sports facilities sorted by the final scores in descending order. |
| Preconditions: | Final score is calculated and ready for all search facilities. |
| Postconditions: | |
| Priority: | |
| Frequency of Use: | Everytime new final score is calculated. |
| Flow of Events: | 1. Sportify uses the included use case Calculate Final Score to get final score for all sports facilities.<br>2. Sportify ranks them in descending order.<br>3. Sportify displays them according to their ranks.<br>4. Spotify shows all available locations in the search radius including:<br>a. Name of location<br>b. Address<br>c. Type of activity<br>d. Distance based of the mode of travel<br>e. Type of Sports Facility (Free, Booking & Paid)<br>5. Sportify uses the included use case Provide vacancy level to obtain the vacancy level for each court location.<br>6. Sportify uses the included use case Get weather conditions to get the weather conditions for each court location.<br>7. User presses find out more button shows:<br>a. Playing count<br>b. Check-In count<br>c. Pre-check-in count<br>d. Temperature<br>e. Thunderstorm<br>f. Rain<br>g. PSI<br>h. UVI |
| Alternative Flows: | |

| | |
|---|---|
| Exceptions: | EX1: If the final score is negative, out of range, zero, NaN<br>1. Sportify prompts user to restart app. |
| Includes: | UC02, UC03, UC05 |
| Special Requirements: | Requires data from other use cases. |
| Assumptions: | All previous use cases are correct and provide accurate information. |
| Notes and Issues: | |

| | | | |
|---|---|---|---|
| Use Case ID: | UC08 | | |
| Use Case Name: | Pre check-in | | |
| Created By: | Ding Ren | Last Updated By: | Ding Ren |
| Date Created: | 18/2/24 | Date Last Updated: | 19/2/24 |

| | |
|---|---|
| Actor: | User |
| Description: | Allows user to indicate their intent of heading to a location before doing so. This adds on to a live-count of the number of people interested |
| Preconditions: | 1. Status of user must be “Default”<br>2. User not in any of the following statuses: “Pre-checked-in”, “Checked-in”, “Playing” |
| Postconditions: | 1. Changes user status to “Pre-checked-in”<br>2. Increases the counter for pre-check-in at that sport facility |
| Priority: | |
| Frequency of Use: | Everytime the “pre-check-in” button on the subpage of a sport facility is pressed |
| Flow of Events: | 1. User presses “pre-check-in” button<br>2. Sportify checks if the status of user is “Default”<br>3. If yes, Sportify changes user status to “Pre-checked-in”<br>4. Sportify increments the counter for pre-check-in at that sport facility by 1<br>5. Sportify starts timer of 90 minutes for user |
| Alternative Flows: | AF-S2: User status is not “Default”<br>1. Sportify displays error message “ Duplicate Pre-check-ins are not allowed!”<br>AF-S5: User presses “Pre-check-out” button<br>1. Sportify changes user status to “Default” |

| | |
|---|---|
| | 2. Sportify decrements the counter for pre-check-in at that sport facility by 1<br><br>AF-S5: Timer of 90 minutes is up<br>1. Sporify displays error message " Pre-check-in has expired!!"<br>2. Sportify changes user status to "Default"<br>3. Sportify decrements the counter for pre-check-in at that sport facility by 1 |
| Exceptions: | |
| Includes: | |
| Special Requirements: | |
| Assumptions: | All previous use cases are correct and provide accurate information. |
| Notes and Issues: | |

| | | | |
|---|---|---|---|
| Use Case ID: | UC09 | | |
| Use Case Name: | Check-in | | |
| Created By: | Ding Ren | Last Updated By: | Ding Ren |
| Date Created: | 19/2/24 | Date Last Updated: | 19/2/24 |

| | |
|---|---|
| Actor: | User |
| Description: | Allows user to inform others that they have arrived and is currently at the sport facility. This adds on to a live-count of the number of people present at the sport facility but has yet started playing or has yet to find a playing group or has yet to find a playing partner. |
| Preconditions: | 1. Status of user must be "Pre-checked-in"<br>2. User not in any of the following statuses: "Default", "Checked-in", "Playing" |
| Postconditions: | 1. Changes user status to "Checked-in"<br>2. Increases the counter for Check-in at that sport facility<br>3. Decreases the counter for Pre-check-in at that sport facility |
| Priority: | |
| Frequency of Use: | Everytime the "Check-in" button on the subpage of a sport facility is pressed |
| Flow of Events: | 1. User presses "Check-in" button<br>2. Sportify checks if the status of user is "Pre- |

| | |
|---|---|
| | checked-in”<br>3. If yes, Sportify changes user status to “Checked-in”<br>4. Sportify increments the counter for check-in at that sport facility by 1<br>5. Sportify decrements the counter for pre-check-in at that sport facility by 1<br>6. Sportify starts timer of 180 minutes for user |
| Alternative Flows: | AF-S2: User status is not “pre-checked-in”<br>1. Sportify displays error message “ Please perform pre-check-in first!”<br>2. Sportify prompts user to restart app.<br>AF-S6: User presses “Check-out” button<br>1. Sportify changes user status to “Default”<br>2. Sportify decrements the counter for “Checked-in” at that sport facility by 1<br><br>AF-S6: Timer of 180 minutes is up<br>1. Sporify displays error message “ Check-in has expired! Playing time is up!”<br>2. Sportify changes user status to “Default”<br>3. Sportify decrements the counter for “Checked-in” at that sport facility by 1 |
| Exceptions: | |
| Includes: | |
| Special Requirements: | |
| Assumptions: | All previous use cases are correct and provide accurate information. |
| Notes and Issues: | |

| | | | |
|---|---|---|---|
| Use Case ID: | UC10 | | |
| Use Case Name: | Playing | | |
| Created By: | Ding Ren | Last Updated By: | Ding Ren |
| Date Created: | 19/2/24 | Date Last Updated: | 19/2/24 |

| | |
|---|---|
| Actor: | User |
| Description: | Allows user to inform others that they have found a playing group at the sport facility and have begun |

| | |
|---|---|
| | playing. This adds on to a live-count of the number of people playing at the sport facility. |
| Preconditions: | 1. Status of user must be “Checked-in”<br>2. User not in any of the following statuses: “Default”, “Pre-checked-in”, “Playing”<br>3. Timer of 180 minutes is running |
| Postconditions: | 1. Changes user status to “Playing”<br>2. Increases the counter for “Playing” at that sport facility<br>4. Decreases the counter for “Checked-in” at that sport facility |
| Priority: | |
| Frequency of Use: | Everytime the “Playing” button on the subpage of a sport facility is pressed |
| Flow of Events: | 1. User presses “Playing” button<br>2. Sportify checks if the status of user is “Checked-in”<br>3. If yes, Sportify changes user status to “Playing”<br>4. Sportify increments the counter for “Playing” at that sport facility by 1<br>5. Sportify decrements the counter for “Checked-in” at that sport facility by 1 |
| Alternative Flows: | AF-S1: User presses “Check-out”<br>1. Sportify changes user status to “Default”<br>2. Sportify decrements the counter for check-in at that sport facility by 1<br><br>AF-S2: User status is not “Checked-in”<br>1. Sportify displays error message “ Please perform check-in first!”<br>2. Sportify prompts user to restart app.<br><br>AF-S5: User presses “Check-out” button<br>1. Sportify changes user status to “Default”<br>2. Sportify decrements the counter for “Playing” at that sport facility by 1 |
| Exceptions: | EX1: Timer of 180 minutes is up<br>1. Sporify displays error message “Check-in has expired! Playing time is up!”<br>2. Sportify changes user status to “Default”<br>3. Sportify decrements the counter for “Playing” at that sport facility by 1 |
| Includes: | |

| | |
|---|---|
| Special Requirements: | |
| Assumptions: | All previous use cases are correct and provide accurate information. |
| Notes and Issues: | |