**Use Cases**

**for**

**WITFIT**

**Version 0.3**

**Prepared by WomenInTech**

**WomenInTech (CZ2006 BS3)**

**13 Feb 2022**

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| WomenInTech | 31/01/2022 | Initial documentation | 0.1 |
| WomenInTech | 06/02/2022 | Merging and Completion of Initial Draft | 0.2 |
| WomenInTech | 13/02/2022 | Change some flows and added new use case | 0.3 |

[**UC001**](#_heading=h.ptlupy9m4ibb) **5**

[**Create main app account**](#_heading=h.ov74vj8zpzsm) **5**

[**UC002**](#_heading=h.s05ih2ido9y3) **6**

[**Enter user details**](#_heading=h.i50xjqkwk4yu) **6**

[**UC003**](#_heading=h.b19js4rcjfqp) **8**

[**Login to main app account**](#_heading=h.76aoi5b4tq0) **8**

[**UC004**](#_heading=h.9zrmddo5px3) **9**

[**Update personal information**](#_heading=h.mjmb66azrmbz) **9**

[**UC005**](#_heading=h.sr5yfzwhad2u) **10**

[**Select new fitness goals**](#_heading=h.a1213hdhxakh) **10**

[**UC006**](#_heading=h.vke0jqniznov) **11**

[**Recommend personalised workout**](#_heading=h.o2ks597gczpl) **11**

[**UC007**](#_heading=h.ren0t5fgb99d) **12**

[**Search for exercises**](#_heading=h.1ii3unvz54ut) **12**

[**UC008**](#_heading=h.3bj13he13ywb) **13**

[**Conduct workout**](#_heading=h.46xpgzkws2c9) **13**

[**UC009**](#_heading=h.6eh0zkppppxz) **15**

[**Edit workout routine**](#_heading=h.qyjle2dnymfw) **15**

[**UC010**](#_heading=h.6a4u6ftz9nuo) **16**

[**Manage music**](#_heading=h.uw2jqsu9oorq) **16**

[**UC011**](#_heading=h.v4z1hksxfdnv) **17**

[**Display workout summary**](#_heading=h.bfr0fapukszf) **17**

[**UC012**](#_heading=h.7ovlvzzfczdg) **18**

[**Workout Progress Calendar**](#_heading=h.chgqtqvwdw31) **18**

[**UC013**](#_heading=h.3fwu6wvvxvx8) **19**

[**Display workout history**](#_heading=h.8jruzj325lgp) **19**

[**UC015**](#_heading=h.58q5znefo1bz) **21**

[**Login to gym buddy account**](#_heading=h.pnu93k61e6bq) **21**

[**UC018**](#_heading=h.rm784jfosqko) **26**

[**View specific buddy details**](#_heading=h.q1l6cur9e246) **26**

[**UC019**](#_heading=h.lvq8ox5y9a8l) **27**

[**Request to pair up with buddy**](#_heading=h.qx4ffa6yexmm) **27**

[**UC020**](#_heading=h.es24foleve8u) **29**

[**View chat with buddy**](#_heading=h.z0hzsf115cl1) **29**

[**UC021**](#_heading=h.50z2v29saxvl) **31**

[**Send message**](#_heading=h.h1y3jqju3wzg) **31**

[**UC022**](#_heading=h.amuztgppa6w) **33**

[**Remove buddy**](#_heading=h.lv74jb1opa8e) **33**

[**UC023**](#_heading=h.9133zt7yae5q) **34**

[**View all chats**](#_heading=h.dx1funtvpjz6) **34**

| Use Case ID: | UC001 | | |
| --- | --- | --- | --- |
| Use Case Name: | Create main app account | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 31/1/2022 | Date Last Updated: | 6/2/2022 |

| Actor: | UnregisteredUser (Initiating actor), UserDatabase |
| --- | --- |
| Description: | First time users are to create a WITFIT account to use the application. |
| Preconditions: | 1. This use case extends UC002 Login to the main app account. 2. User does not have an existing account. 3. Users must have an email. 4. An internet connection is required. |
| Postconditions: | 1. User has an account created with a username. 2. Users will be able to see the homepage of their account. 3. The account information will be stored in the app database. |
| Priority: | High.  Users will not be able to login to access any of the application functionalities without an account. |
| Frequency of Use: | Rarely.  The first time the user uses the app. |
| Flow of Events: | 1. UnregisteredUser selects “I don’t have an account” 2. LoginUI brings User to SignUpUI. 3. UnregisteredUser enters First Name, Last Name, Email, Birthday, Gender, Username, Password, Confirm Password and clicks sign up button. 4. SignUpUI send username to the LoginAccessSystem 5. LoginAccessSystem validates the username using UserDatabase and logs the user in. 6. UnregisteredUser is prompted for further information which invokes UC002. |
| Alternative Flows: | AF-S5: If the user’s credentials are already in use.   1. App displays the error message “Username already in use“. 2. The app returns to step 2. |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | The user’s email address is valid. |
| Notes and Issues: | - |

| Use Case ID: | UC002 | | |
| --- | --- | --- | --- |
| Use Case Name: | Enter user details | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 13/2/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | UnregisteredUser (Initiating actor), UserDatabase |
| --- | --- |
| Description: | First time users are to enter their credentials as part of account creation. |
| Preconditions: | 1. Invocation as an inclusion of use case UC001 Create main app account. 2. User does not have an existing account. 3. Users must have an email. 4. An internet connection is required. |
| Postconditions: | 1. User has an account created with a username and password. 2. HomeUI will be displayed. 3. User details will be stored in the UserDatabase. |
| Priority: | High.  Users will not be able to login to access any of the application functionalities without an account. |
| Frequency of Use: | Rarely.  The first time the user uses the app. |
| Flow of Events: | 1. User presses continue account creation. 2. UserDetailsUI will prompt the user for their height and weight. 3. UserDetailsUI sends height and weight to UserDetailsValidator. 4. UserDetailsValidator validates the height and weight. 5. UserDetailsUI will prompt the user for any physical injuries or pre-existing health conditions. 6. User answers “No” to both questions. 7. UserDetailsUI will prompt the user to enter their fitness goals. 8. User enters fitness goals and selects next to confirm. 9. LoginAccessSystem sends height,weight, fitness goals and menstruation details to UserDatabase. 10. HomeUI is displayed and displays a welcome message |
| Alternative Flows: | AF-S4: If the user’s height or weight is invalid   1. UserDetailsUI displays the error message “Please enter a valid height and weight” 2. The app returns to step 1.   AF-S6: If the user answers “Yes” they have physical injuries.   1. UserDetailsUI will prompt the user for their physical injuries. 2. UnregisteredUser enter physical injury details. 3. The app will proceed to Step 6.   AF-S6: If the user answers “Yes” they have pre-existing medical conditions.   1. UserDetailsUI will prompt the user for their medical condition. 2. UnregisteredUser enter their medical condition. 3. The app will proceed to Step 6.   AF-S6: If the user answers “Yes” to both questions.   1. UserDetailsUI will prompt the user for their physical injuries. 2. UnregisteredUser will be prompted to enter their type of injury. 3. UserDetailsUI will prompt the user for their medical condition. 4. UnregisteredUser will be prompted to enter their medical condition.. 5. Upon clicking next, the app will proceed to step 6.   AF-S8: Upon clicking next, if the user is a female.   1. UserDetailsUI will prompt the user for their menstruation details. 2. UnregisteredUser will be prompted to enter the date of their last period. 3. UnregisteredUser selects “Not sure” or “I am pregnant” or enters the date of last period. 4. The app will proceed to step 8. |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | The user’s email address is valid. |
| Notes and Issues: | - |

| Use Case ID: | UC003 | | |
| --- | --- | --- | --- |
| Use Case Name: | Login to main app account | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | RegisteredUser(Initiating Actor), UserDatabase |
| --- | --- |
| Description: | User logs into app using their WITFIT account |
| Preconditions: | 1. Users must have an existing account for successful login. 2. An internet connection is required. |
| Postconditions: | 1. User will be logged in to the app 2. HomePage will be displayed |
| Priority: | High.  Use cases UC003 View gym map, UC006 View gym details, UC008 Submit review, UC009 Update personal information, UC014 Conduct workout, UC015 Create customised workout routine, UC018 Display workout summary and UC021 Create gym buddy account will not be available without logging in. |
| Frequency of Use: | Rarely.  The first time the user uses the app.  The only instance other than this is when the user chooses to sign out. |
| Flow of Events: | 1. User opens the app. 2. LoginUI displays login page. 3. User enters username and password and presses the login button. 4. LoginUI send username and password to the LoginAccessSystem 5. LoginAccessSystem validates the username and password using UserDatabase and logs the user in. 6. HomeUI is displayed and displays a welcome message |
| Alternative Flows: | AF-S5: If user’s credentials are invalid   1. LoginUI will display a message indicating invalid password/username. 2. The app returns to step 2. |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | The user has a valid email address. |
| Notes and Issues: | The app does not limit the number of attempts the user has to input their credentials. |

| Use Case ID: | UC004 | | |
| --- | --- | --- | --- |
| Use Case Name: | Update personal information | | |
| Created By: | Gabriel Tang | Last Updated By: | Gabriel Tang |
| Date Created: | 31/1/2022 | Date Last Updated: | 20/2/2022 |

| Actor: | User (Initiating Actor), UserDatabase |
| --- | --- |
| Description: | User logs into the app and update personal information |
| Preconditions: | 1. Invocation as an inclusion of use case UC002 Login to main app account. 2. An internet connection is required. |
| Postconditions: | App stores update name and profile picture in the App Database |
| Priority: | Low.  The user can still use all other functionalities without updating personal information. |
| Frequency of Use: | Low.  This function will only be invoked if the user wants to edit their personal information. |
| Flow of Events: | 1. User selects ‘Edit Profile’ on WorkoutUI 2. WorkoutUI brings User to UserDetailsUpdaterUI 3. UserDetailsUpdaterUI displays pre-filled User information 4. UserDetailsUpdaterUI will prompt the User to edit their personal information. 5. If the User wishes to changes their fitness goals, the app invokes the extended use case UC005 Select new fitness goals 6. The User selects ‘Confirm Settings’ after all fields are filled. 7. UserDetailsUpdaterUI calls ValidateAndUpdate to check if there is updated information 8. If true, UserDetailsUpdaterUI calls updateFitnessGoals to send updated information to UsersDetailValidator 9. UsersDetailValidator will then update fitness goal 10. WorkoutUI displays an 'Update Successful’ message. |
| Alternative Flows: | AF-S6: User selects ‘Confirm Settings’ when not all fields are filled   1. WorkoutUI displays a red ‘Please fill in all fields’ warning and outlines the required fields in red. 2. The app will return to step 4. |
| Exceptions: | - |
| Includes: | UC002 Login to main app account |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| Use Case ID:⠀ | UC005 | | |
| --- | --- | --- | --- |
| Use Case Name:⠀ | Select new fitness goals | | |
| Created By:⠀ | Gabriel Tang | Last Updated By: | Gabriel Tang |
| Date Created:⠀ | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | User (Initiating Actor), UserDatabase |
| --- | --- |
| Description: | User uses the app and selects his/her fitness goals |
| Preconditions: | 1. Invocation as an inclusion of use case UC002 Login to main app account. 2. This use case extends UC004 Update personal information 3. An internet connection is required |
| Postconditions: | The app shall update the user’s selected fitness goals into the user database. |
| Priority: | Medium.  This functionality allows users to define and work towards their own objectives, an important factor in fitness ensuring continued motivation and discipline in achieving one’s fitness goal. |
| Frequency of Use: | Low.  This function will only be invoked if the user chooses to change their fitness goals. Users will not change their fitness goals frequently. |
| Flow of Events: | 1. User selects select fitness goals 2. UserDetailsUpdaterUI calls LoadUser to retrieve User information on their pre-selected fitness goal. 3. UserDetailsUpdaterUI extends LoadFromDb to display a list of available fitness goals. 4. UserDetailsUpdaterUI prompts the User to select a new fitness goal. 5. If the User has selected a new fitness goal, the user can click on confirm settings. 6. The app displays a confirmation message at the user profile, indicating that the update of fitness goals is successful. |
| Alternative Flows: | AF-S1: If the User does not have a preselected fitness goal   1. WorkoutUI will display ‘No fitness goal selected’ 2. WorkoutUI will return to step 2.   AF-S4: If the User did not select a new fitness goal   1. WorkoutUI will display an option to return back to the previous step. 2. If the User selects the option to return back to the previous step, the app will return to UC004 Update Personal Information. |
| Exceptions: |  |
| Includes: | UC002 Login to main app account |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| Use Case ID: | UC006 | | |
| --- | --- | --- | --- |
| Use Case Name: | Recommend personalised workout | | |
| Created By: | Gabriel Tang | Last Updated By: | Bryan Leow |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | User, UserDatabase |
| --- | --- |
| Description: | User selects a recommended personalised workout that will be generated by the app |
| Preconditions: | 1. Invocation as an inclusion of use case UC002 Login to main app account. 2. An internet connection is required |
| Postconditions: | 1. User will see their overall workout selected (Duration + exercises) |
| Priority: | High.  User selects workout based on their personal information input hence it is important for the app to recommend a workout that is suitable for the user |
| Frequency of Use: | Medium.  Everytime the user uses the app. |
| Flow of Events: | 1. On successful login, the user selects to add a workout. 2. WorkoutUI invokes the included Use Case UC004 Update Personal Information to update user’s details 3. After returning with the updated user details, WorkoutUI will create the workout via WorkoutCreatorUI. 4. WorkoutCreatorUI calls the WorkoutRecommender to get the recommended workout. 5. WorkoutRecommender retrieves the user’s injuries from the UserDatabase. 6. WorkoutRecommender retrieves the user’s goals from the UserDatabase. 7. WorkoutRecommender generates the list of recommendations using included Use Case UC007 Search for exercises. 8. Iterate through the list of recommendations to retrieve the exercises via the ExerciseSearcher. 9. WorkoutRecommender returns the list of exercises to the WorkoutCreatorUI. 10. WorkoutCreatorUI shows the workout that was created. |
| Alternative Flows: | AF-S1: If the operation fails   1. A toast message will be displayed indicating the error 2. App returns to step 1 |
| Exceptions: | - |
| Includes: | UC002 Login to main app account |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| Use Case ID: | UC007 | | |
| --- | --- | --- | --- |
| Use Case Name: | Search for exercises | | |
| Created By: | Bryan Leow | Last Updated By: | Bryan Leow |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | User (initiating actor), Exercise API |
| --- | --- |
| Description: | Filter our database of exercises based on a search criteria. |
| Preconditions: | 1. Invocation as inclusion from UC006 Recommend personalised workout or extension from UC009 Edit workout routine. 2. An internet connection is required |
| Postconditions: | A filtered list of exercises matching the search criteria is returned to the calling use case. |
| Priority: | High.  Being able to search our database for exercises is crucial as it enables users to select the exercises they desire, whether due to personal preference in-line with their training goals or due to previous injuries preventing some exercises from being carried out. |
| Frequency of Use: | High.  It is invoked as an extension of two other highly used use cases UC006 Recommend personalised workout or UC009 Edit workout routine |
| Flow of Events: | 1. ExerciseSearcher invokes this use case as an inclusion from UC006 Recommend personalised workout or extension of UC009 Edit workout routine, from WorkoutUI 2. ExerciseSearcher gets the list of exercises matching that body part from the Exercise API. 3. ExerciseSearcher parses and validates data received from Exercise API. 4. ExerciseSearcher returns the list of exercises to the calling class. |
| Alternative Flows: | AF-S3: If the search criteria cannot be parsed:   1. Exercise Searcher displays the message ‘The app does not understand your search’. 2. Exercise Searcher sets the filtered list to be empty. 3. Exercise Searcher returns to step 4. |
| Exceptions: | EX1: The Exercise API cannot be accessed:   1. Exercise Searcher displays the message ‘The exercise database is currently unavailable, please try again later’ |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| Use Case ID: | UC008 | | |
| --- | --- | --- | --- |
| Use Case Name: | Conduct workout | | |
| Created By: | Bryan Leow | Last Updated By: | Bryan Leow |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | User (initiating actor), UserDatabase |
| --- | --- |
| Description: | Facilitates the overall flow of a workout routine. |
| Preconditions: | 1. Invocation as an inclusion of use case UC002 Login to main app account. 2. User has already created a workout with valid exercises. |
| Postconditions: | A workout routine is completed. |
| Priority: | High.  This is one of the key functionalities of the app to facilitate users to exercise and is thus extremely important. |
| Frequency of Use: | High.  Users will almost always use this functionality when wanting to exercise, which ranges from 3-7 times per week per user. |
| Flow of Events: | 1. On successful login, user selects the desired workout they wish to start. 2. WorkoutUI receives the user selection and passes the Workout ID to the WorkoutConductorUI to start the workout. 3. WorkoutConductorUI displays a timer. 4. If user presses any music controls, the WorkoutConductorUI invokes Use Case UC010 Manage music to manage the music. 5. While there is an exercise for the user to currently perform, the WorkoutConductorUI will retrieve the workout status from the WorkoutManager. 6. For exercise by sets, while the sets completed is less than the sets required to do, users will click ‘Complete Set’ on WorkoutConductorUI. 7. WorkoutConductorUI will update the exercise progress for the current exercise via the Workout Manager. 8. Workout Manager will mark the set as done in the Exercise. 9. When sets completed is equal to sets required to do, Workout Manager will mark the exercise as completed. 10. Workout Manager will mark the exercise as done in the Exercise. 11. User will select the next exercise on WorkoutConductorUI. 12. WorkoutConductorUI retrieves the next exercise in the workout from the WorkoutManager. 13. When there are no more exercises left, the WorkoutConductorUI will stop the workout via the WorkoutManager. 14. WorkoutManager will invoke UC011 Display workout summary to display workout summary. |
| Alternative Flows: | AF-S6: For exercise by timer   1. WorkoutUI runs down the exercise timer via the Workout Manager. 2. When the timer runs out, the app returns to step 8.   AF-S9: When the user adds more sets   1. User selects ‘Add more sets’ on the WorkoutConductorUI. 2. WorkoutConductorUI updates the workout via the Workout Manager. 3. Workout Manager updates the conductor UI, and adds set to the Exercise. 4. The app returns to step 9.   AF-S11: If user presses ‘Pause workout’   1. WorkoutConductorUI pauses the current workout via Workout Manager. 2. If the user presses ‘Resume workout’, WorkoutConductorUI resumes the workout and returns to step 11. 3. Otherwise, if the user presses ‘Stop workout’, the WorkoutConductorUI stops the workout and returns to step 13. |
| Exceptions: | - |
| Includes: | UC002 Login to main app account, UC010 Manage music, UC011 Display workout summary |
| Special Requirements: | - |
| Assumptions: | The included use cases UC002 Login to main app account, UC010 Manage music, and UC011 Display workout summary are functional. |
| Notes and Issues: | - |

| Use Case ID: | UC009 | | |
| --- | --- | --- | --- |
| Use Case Name: | Edit workout routine | | |
| Created By: | Gladys Loh | Last Updated By: | Gladys Loh |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | User (Initiating actor), Exercise API, UserDatabase |
| --- | --- |
| Description: | Users are able to edit their recommended routine. |
| Preconditions: | 1. Invocation as an inclusion of use case UC002 Login to main app account. |
| Postconditions: | The app will update the workout routine according to what the user specified to the app database. |
| Priority: | Medium.  This is a main functionality - that is to allow users to have creativity in customisation to suit their needs. |
| Frequency of Use: | Medium.  This function will only be invoked if users want to edit their customised routine. |
| Flow of Events: | 1. On successful login, the user can select WorkoutApp. 2. User enters WorkoutApp 3. User press workout to be edited 4. WorkoutEditorUI is shown and user can edit workout title and description, add exercises, edit reps, add set, delete set. 5. User confirm changes to be updated and WorkoutPlanner will save information to the UserDatabase. 6. WorkoutApp will display their updated workout. |
| Alternative Flows: | AF-S1: If user press “Add exercise”:   1. UC007 Search for Exercises will be invoked for users to search for exercises to be added.   AF-S2: If user press “Update workout”:   1. WorkoutEditorUI will prompt confirmation from the user. 2. If the user presses “confirm” and information is complete, WorkoutApp will be navigated and WorkoutPlanner will update their information. 3. If the user presses “cancel”, the user will remain on WorkoutEditorUI. 4. If the user presses “confirm” and information is incomplete, WorkoutEditorUI will prompt the user that information entered is not valid and will allow the user to re-enter. |
| Exceptions: | - |
| Includes: | UC002 Login to main app account |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| Use Case ID: | UC010 | | |
| --- | --- | --- | --- |
| Use Case Name: | Manage music | | |
| Created By: | Gladys Loh | Last Updated By: | Gabriel Tang |
| Date Created: | 31/1/2022 | Date Last Updated: | 20/2/2022 |

| Actor: | User (Initiating actor), Music API |
| --- | --- |
| Description: | Users are recommended a workout music playlist but are allowed to change if they want. |
| Preconditions: | 1. This use case extends UC006 Recommend personalised workout 2. The user must be connected to their spotify account. 3. An internet connection is required. |
| Postconditions: | The app must allow the user to select music. |
| Priority: | Low.  This feature is merely to boost user experience. |
| Frequency of Use: | Medium.  High likelihood that users would want to play music during their workout. |
| Flow of Events: | 1. Workout Planner sends relevant information from the customised workout created from UC006 Recommend Personalised workout to Music Recommendation 2. Music Recommendation parses workout information to generate a range of bpms and genres suitable for the workout. 3. WorkoutUI prompt user for choice of intensity for workout 4. Music Recommendation retrieves a filtered list of song choices from the SpotifyAPI and put them into a playlist. 5. Music Recommendation returns a playlist to Music Manager 6. Music Recommendation send recommended playlist to WorkoutUI for user selection 7. Workout Planner will prompt the user if they wish to conduct the workout with the recommended music playlist. 8. If Yes, the app will return to UC008 Conduct workout along with the recommended music playlist. |
| Alternative Flows: | AF-S4: If the user selects No,   1. Workout Planner will prompt the user to edit their recommended music playlist and call Music Manager. 2. Music Manager will allow User to make edits to the playlist via the Music API. 3. If the User chooses not to edit, the app will return to UC008 Conduct workout without any music playlist selected. |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | User has a valid spotify account. |
| Notes and Issues: | - |

| Use Case ID: | UC011 | | |
| --- | --- | --- | --- |
| Use Case Name: | Display workout summary | | |
| Created By: | Foo Zhi Kai | Last Updated By: | Gabriel Tang |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | User (Initiating actor), UserDatabase |
| --- | --- |
| Description: | Display summary of completed workout |
| Preconditions: | 1. Invocation as an inclusion of use case UC002 Login to main app account. 2. Invocation as an inclusion of use case UC008 Conduct workout. 3. An internet connection is required. |
| Postconditions: | The app must display a summary of the workout |
| Priority: | Medium.  A key component of the user experience. |
| Frequency of Use: | Medium.  The summary will be displayed every time the user completes a workout. |
| Flow of Events: | 1. The user uses the included use case UC008 Conduct workout to start conducting a workout. 2. The user completes the workout. 3. On successful completion, WorkoutSummaryUI displays a summary of the workout. 4. WorkoutSummaryUI will display the time and date the workout was completed. 5. WorkoutSummary will calculate the total calories, fitness and strength. 6. WorkoutSummaryUI will also display some basic statistics like total sets, total calories, duration of workout, total fitness and strength used. |
| Alternative Flows: |  |
| Exceptions: | - |
| Includes: | UC002 Login to main app account, UC008 Conduct workout |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| Use Case ID: | UC012 | | |
| --- | --- | --- | --- |
| Use Case Name: | Workout Progress Calendar | | |
| Created By: | Foo Zhi Kai | Last Updated By: | Foo Zhi Kai |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | User (Initiating Actor), UserDatabase |
| --- | --- |
| Description: | User adds workout to their progress calendar |
| Preconditions: | 1. This use case extends UC011 Display workout summary. 2. An internet connection is required. |
| Postconditions: | 1. App stores workout summary into user’s progress calendar 2. App displays an updated progress calendar |
| Priority: | Medium.  A key component of the user experience and motivates the user to continue using the app. |
| Frequency of Use: | Low  This functionality will only be invoked when the user chooses to add their workout to their progress calendar whenever the user completes a workout. |
| Flow of Events: | 1. HomeUI will display the user's daily workout progress. 2. User press on WorkoutProgressCalendar 3. WorkoutProgressCalendar will display daily workout based on date 4. User can select to view monthly workout progress and WorkoutProgress will calculate the monthly statistics. 5. WorkoutProgressCalendar will display the workout summary displayed in use case UC011 Display Workout Summary that was completed on that date. |
| Alternative Flows: | AF-S1: If the user is a female,   1. WorkoutProgressCalendar will retrieve information regarding the user’s menstruation cycle from the UserDatabase 2. WorkoutProgressCalendar will display this information   AF-S2: If the user does not have workout on month or date selected,   1. WorkoutProgressCalendar will display no information |
| Exceptions: | - |
| Includes: |  |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| Use Case ID: | UC013 | | |
| --- | --- | --- | --- |
| Use Case Name: | Display workout history | | |
| Created By: | Joshua Khoo | Last Updated By: | Gabriel Tang |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | User (Initiating Actor), UserDatabase |
| --- | --- |
| Description: | User is able to access and view past workouts |
| Preconditions: | 1. This use case extends UC012 Workout progress calendar 2. The user has done at least a workout |
| Postconditions: | 1. The app must display a summary of past workouts 2. The app must display a history of all previous workouts |
| Priority: | Medium.  A key component for users to track their progress. This functions as an important motivating force for them to continue using the app. |
| Frequency of Use: | Low.  The workout history will only be displayed when the user selects this option on the progress calendar. Users are unlikely to check their history often. |
| Flow of Events: | 1. WorkoutApp will retrieve information regarding the user’s previous workouts from UserDatabase using their userID. 2. WorkoutApp will display a summary of past workouts, such as title, description and whether it is completed or not. 3. User is able to sort and filter their workouts based on date, tile and completion. |
| Alternative Flows: | - |
| Exceptions: | EX1 User has not done any workouts   1. The app will display an error message “You have not performed any workout”. 2. The app will prompt them to conduct a workout. 3. If the user selects the option to conduct a workout, the app will return to UC008 Conduct Workout. |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| Use Case ID: | UC014 | | |
| --- | --- | --- | --- |
| Use Case Name: | Create gym buddy account | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | RegisteredUser (Initiating Actor), UserDatabase |
| --- | --- |
| Description. | Create user profile for gym buddy application function |
| Preconditions: | 1. Invocation as an inclusion of use case UC003 Login to main app account. 2. This use case extends UC015 3. The user must have an internet connection. |
| Postconditions: | The app will store the user's gym buddy data in the UserDatabase.  The user will be interfaced to the homepage, indicating successful account creation. |
| Priority: | High.  Users will not be able to login to access any of the “find a gym buddy” functionalities without an account. |
| Frequency of Use: | Once.  The first time the user tries to use the “find a gym buddy” functionality. |
| Flow of Events: | 1. GymBuddyLoginUI prompts user to enter a brief introduction, their preferred workout time, preferred buddy gender, their goals, areas of expertise, style of training, preference of training location and photo of themself. 2. The user fills up all information fields and clicks on create buddy profile. 3. GymBuddyLoginUI sends new gym buddy info to GymBuddyAccessSystem 4. GymBuddyAccessSystem creates and sends a new GymBuddyProfile to UserDatabase. 5. GymBuddyAccessSystem sends a signal and GymBuddyProfile to GymBuddy. 6. GymBuddyUI displays the user's gym buddy profile, indicating successful login. |
| Alternative Flows: | AF-S3: User clicks on create buddy profile when not all fields are filled.   1. GymBuddyLoginUI will display a red text, please fill up all fields. 2. The app returns to step 2. |
| Exceptions: | - |
| Includes: | UC002 Login to main app account. |
| Special Requirements: | - |
| Assumptions: | The user has unlimited tries to fill in all the fields. |
| Notes and Issues: | - |

| Use Case ID: | UC015 | | |
| --- | --- | --- | --- |
| Use Case Name: | Login to gym buddy account | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | RegisteredUser (Initiating Actor), UserDatabase |
| --- | --- |
| Description. | Login to gym buddy to access gym buddy functionalities |
| Preconditions: | 1. Invocation as an inclusion of use case UC003 Login to main app account. 2. The user must have an internet connection. 3. The user must have a gym buddy account. |
| Postconditions: | The app will store user login information to UserDatabase.  GymBuddyUI displays gym buddy profile. |
| Priority: | High.  Use cases UC016 Update account preferences, UC017 View suggested buddies, UC018 View specific buddy details, UC019 Request to pair up with buddy, UC020 View chat with buddy, UC021 Send message UC022 Remove buddy and UC023 View all chats will not be available without logging in. |
| Frequency of Use: | Rarely.  This function will only be used if the user logs out from the “find a gym buddy” functionality. |
| Flow of Events: | 1. User selects “Gym Buddy”. 2. GymBuddyLoginUI sends username and signal to GymBuddyAccessSystem. 3. GymBuddyAccessSystem validates entry and retrieves gym buddy info using UserDatabase. 4. GymBuddyAccessSystem creates GymBuddyProfile. 5. GymBuddyAccessSystem sends a signal and GymBuddyProfile to GymBuddy. 6. GymBuddyHomeUI displays the user's gym buddy profile, indicating successful login. |
| Alternative Flows: | AF-S3: GymBuddyAccessSystem does not validate the entry   1. The GymBuddyLoginUI will be redirected to UC014 Create Gym Buddy Account |
| Exceptions: | - |
| Includes: | UC002 Login to main app account |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| Use Case ID: | UC016 | | |
| --- | --- | --- | --- |
| Use Case Name: | Update account preferences | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | RegisteredUser (Initiating Actor), UserDatabase |
| --- | --- |
| Description. | This functionality allows the user to update their account preferences with regards to their gym buddy account. |
| Preconditions: | 1. Invocation as an inclusion of use case UC015 Login to gym buddy account. 2. The user must have an internet connection. |
| Postconditions: | The app will update the user data that was edited in the UserDatabase. |
| Priority: | Low.  The user can still use all other functionalities in the “find a gym buddy” interface even without updating account preferences. |
| Frequency of Use: | Low.  This function will only be invoked if the user wants to edit their preferences. |
| Flow of Events: | 1. The user clicks “Edit Profile” 2. UpdateGymBuddyUI sends a signal to UpdateGymBuddyManager. 3. UpdateGymBuddyManager retrieve 4. UpdateGymBuddyUI will display pre-filled information. 5. UpdateGymBuddyUI prompts users to enter a brief introduction, their preferred workout time, preferred buddy gender, their goals, areas of expertise, style of training, preference of training location and photo of themself. 6. The user fields up all information fields and clicks on confirm changes. 7. UpdateGymBuddyUI sends updated gym buddy info to UpdateGymBuddyManager 8. UpdateGymBuddyManager updates and send GymBuddyProfile to UserDatabase. 9. UpdateGymBuddyManager sends a signal and GymBuddyProfile to UpdateGymBuddyUI. 10. UpdateGymBuddyUI displays a confirmation message at the user profile, indicating that the update of preferences is successful. |
| Alternative Flows: | AF-S4: User clicks on confirm changes when all fields are not filled   1. UpdateGymBuddyUI will display a red text and prompt the user to fill up all fields. 2. The app will return to step 3. |
| Exceptions: | - |
| Includes: | UC015 Login to gym buddy account |
| Special Requirements: | - |
| Assumptions: | The user’s location is kept on throughout. |
| Notes and Issues: | - |

| Use Case ID: | UC017 | | |
| --- | --- | --- | --- |
| Use Case Name: | View suggested buddy | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 13/2/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | RegisteredUser (Initiating Actor), UserDatabase |
| --- | --- |
| Description. | View one suggested buddy |
| Preconditions: | 1. Invocation as an inclusion of use case UC015 Login to gym buddy account. 2. The user must have an internet connection. 3. The user must have accepted the app permission to access the user’s location. |
| Postconditions: | The app displays one profile of a suggested buddy. |
| Priority: | Medium.  This functionality will enable users to find buddies to match. It is a core feature of the gymm buddy functional requirement without which it will not work to fulfil its purpose. Having a workout partner is important to motivate users to continue working out and keep each other accountable for their fitness goals. |
| Frequency of Use: | Medium.  This function will only be invoked if the user chooses to utilise the gym buddy functionality. |
| Flow of Events: | 1. User selects find a buddy. 2. FindBuddyUI displays “Finding matches” and sends username and signal to RecommendationEngine. 3. RecommendationEngine sends user’s username to MatchmakingAlgo. 4. MatchmakingAlgo retrieves user’s PreferredGymBuddyInformation from UserDatabase. 5. MatchmakingAlgo retrieves OtherUserInfo from UserDatabase. 6. MatchmakingAlgo returns matching scores to RecommendationEngine. 7. RecommendationEngine retrieves OtherBuddyProfile from UserDatabase based on score and sends signal to FindBuddyUI. 8. FindBuddyUI displays suggested OtherBuddyProfile. 9. User selects cross. 10. RecommendationEngine sends OtherBuddyProfile to UserDatabase and marks the OtherBuddyProfile as “Do not recommend”. 11. OtherBuddyProfile is removed from the recommendation engine. 12. RecommendationEngine returns to step 7 if there are more matches |
| Alternative Flows: | AF-S9: User clicks on the profile   1. The app invokes the extended use case UC018 View specific buddy details. 2. The app will return to step 8.   AF-S9: User selects Tick   1. The app invokes the extended use case UC019 Request to pair up with buddy. 2. The app will return to step 11.   AF-S12: There are currently no more matches.   1. FindBuddyUI displays “There are currently no more matches”. 2. GymBuddyHomeUI will be displayed. |
| Exceptions: | - |
| Includes: | UC015 Login to gym buddy account |
| Special Requirements: | - |
| Assumptions: | The user’s location is kept on throughout. |
| Notes and Issues: | - |

| Use Case ID: | UC018 | | |
| --- | --- | --- | --- |
| Use Case Name: | View specific buddy details | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | RegisteredUser (Initiating Actor), UserDatabase |
| --- | --- |
| Description. | User clicks into a specific buddy and is shown more details of the selected buddy. |
| Preconditions: | 1. Invocation as an inclusion of use case UC015 Login to gym buddy account. 2. This use case extends UC017 View suggested buddy |
| Postconditions: | The app shall display detailed information about the buddy selected, such as their style of training and traits. |
| Priority: | Low.  This functionality will allow users to view more specific details about a buddy should they choose to do so. It is not required for users to match with buddies. |
| Frequency of Use: | Low.  This function will only be invoked if the user chooses to utilise the gym buddy functionality, and chooses to view the details of a specific buddy while in use case UC017 View suggested buddy. |
| Flow of Events: | 1. User clicks on the suggested gym buddy profile 2. FindBuddyUI sends signal to ProfileInfoRetriever. 3. ProfileInfoRetriever retrieves DetailedGymBuddyProfile from UserDatabase. 4. ProfileInfoRetriever sends DetailedGymBuddyProfile to FindBuddyUI . 5. FindBuddyUI displays DetailedGymBuddyProfile . |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | UC015 Login to gym buddy account |
| Special Requirements: | - |
| Assumptions: | The user’s location is kept on throughout. |
| Notes and Issues: | - |

| Use Case ID: | UC019 | | |
| --- | --- | --- | --- |
| Use Case Name: | Request to pair up with buddy | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | RegisteredUser (Initiating Actor), UserDatabase, ChatDatabase |
| --- | --- |
| Description. | The user selects to pair up with the suggested buddy. |
| Preconditions: | 1. This use case extends UC018 View Specific Buddy Details 2. Invocation as an inclusion of use case UC015 Login to gym buddy account. 3. The user must have an internet connection. 4. The user must have accepted the app permission to access the user’s location. |
| Postconditions: | The app will update pairing information in the app database.  The buddy will appear in the user’s buddy list, the next time the user navigates there. |
| Priority: | Medium.  This functionality will allow users to request to pair up with a buddy. It is a core feature of the gym buddy functional requirement without which it will not work to fulfil its purpose. Having a workout partner is important to motivate users to continue working out and keep each other accountable for their fitness goals. |
| Frequency of Use: | Low.  This function will only be invoked if the user chooses to utilise the gym buddy functionality, and chooses to view the details of a specific buddy while in use case UC024 View suggested buddies. |
| Flow of Events: | 1. The user selects Tick. 2. FindBuddyUI sends username and signal to BuddyValidationManager. 3. BuddyValidationManager retrieves GymBuddyInformation from UserDatabase. 4. BuddyValidationManager validates the match. 5. BuddyValidationManager sends a message to UserDatabase to update the match. 6. BuddyValidationManager sends a message to ChatDatabase to create a chat. 7. BuddyValidationManager sends signal and username to the FindBuddyUI. 8. FindBuddyUI displays a message “You have successfully matched with <The other user’s name>”. |
| Alternative Flows: | AF-S4 If the other buddy has yet to click the green tick or the other buddy has selected the “red cross”.   1. BuddyValidationManager sends a signal to the FindBuddyUI. 2. FindBuddyUI displays a message “Pending Match”. |
| Exceptions: | - |
| Includes: | UC015 Login to gym buddy account |
| Special Requirements: | - |
| Assumptions: | The user’s location is kept on throughout. |
| Notes and Issues: | - |

| Use Case ID: | UC020 | | |
| --- | --- | --- | --- |
| Use Case Name: | View chat with buddy | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | RegisteredUser (Initiating Actor), ChatDatabase, Buddy |
| --- | --- |
| Description. | The user chooses to start a chat with a buddy after a match is found. |
| Preconditions: | 1. Invocation as an inclusion of use case UC019 Request to pair up with buddy. 2. Invocation as an inclusion of use case UC015 Login to gym buddy account. 3. There must be a match between the user and the suggested buddy. 4. There must be an existing internet connection. 5. This use case extends UC023 View all chats. |
| Postconditions: | The app will consistently update chat information into the app database.  The app must display chat messages sent by the user onto the buddy’s chat display.  The app must display chat messages sent by the buddy onto the user’s chat display. |
| Priority: | Medium.  This functionality will allow users to chat with a matched buddy. It is a core feature of the gym buddy functional requirement without which it will not work to fulfil its purpose. Having a workout partner is important to motivate users to continue working out and keep each other accountable for their fitness goals. |
| Frequency of Use: | Very Low.  This function will only be invoked if the user chooses to utilise the gym buddy functionality, chooses to request to pair up with buddy and the buddy has done the same, and one of the users wants to send a message. |
| Flow of Events: | 1. User selects a buddy to start a chat. 2. ChatWithBuddyUI sends chatid to ChatManager. 3. ChatManager retrieves ChatInformation from ChatDatabase. 4. ChatManager returns ChatInformation to ChatWithBuddyUI. 5. ChatWithBuddyUI displays conversation. 6. If the user chooses to send message UC021 Send message will be invoked. |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | UC015 Login to gym buddy account, UC019 Request to pair up with buddy |
| Special Requirements: | - |
| Assumptions: | The user’s location is kept on throughout.  For messages to be received and sent concurrently, both the user and the buddy must have a stable internet connection. |
| Notes and Issues: | The app shall display the message sent on the buddy’s chat within 1 one second.  The app shall display the message sent by the buddy on the user’s chat within one second. |

| Use Case ID: | UC021 | | |
| --- | --- | --- | --- |
| Use Case Name: | Send message | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | RegisteredUser (Initiating Actor), ChatDatabase, Buddy |
| --- | --- |
| Description. | The user chooses to start a chat with a buddy after a match is found. |
| Preconditions: | 1. Invocation as an inclusion of use case UC019 Request to pair up with buddy. 2. Invocation as an inclusion of use case UC015 Login to gym buddy account. 3. There must be a match between the user and the suggested buddy. 4. There must be an existing internet connection. 5. This use case extends UC023 View all chats. |
| Postconditions: | The app will consistently update chat information into the app database.  The app must display chat messages sent by the user onto the buddy’s chat display.  The app must display chat messages sent by the buddy onto the user’s chat display. |
| Priority: | Medium.  This functionality will allow users to chat with a matched buddy. It is a core feature of the gym buddy functional requirement without which it will not work to fulfil its purpose. Having a workout partner is important to motivate users to continue working out and keep each other accountable for their fitness goals. |
| Frequency of Use: | Very Low.  This function will only be invoked if the user chooses to utilise the gym buddy functionality, chooses to request to pair up with buddy and the buddy has done the same, and one of the users wants to send a message. |
| Flow of Events: | 1. User sends message. 2. ChatWithBuddyUI display user’s message. 3. ChatManager sends user’s message and chatid to ChatDatabase. 4. ChatManager continuously pings ChatDatabase for any new message. 5. ChatManager sends message to ChatWithBuddyUI. 6. ChatWithBuddyUI displays Buddy’s message. 7. The use case terminates. |
| Alternative Flows: | AF-S4 No message is pinged   1. The app returns to Step 7. |
| Exceptions: | - |
| Includes: | UC015 Login to gym buddy account, UC019 Request to pair up with buddy |
| Special Requirements: | - |
| Assumptions: | The user’s location is kept on throughout.  For messages to be received and sent concurrently, both the user and the buddy must have a stable internet connection. |
| Notes and Issues: | The app shall display the message sent on the buddy’s chat within 1 one second.  The app shall display the message sent by the buddy on the user’s chat within one second. |

| Use Case ID: | UC022 | | |
| --- | --- | --- | --- |
| Use Case Name: | Remove buddy | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | RegisteredUser (Initiating Actor), UserDatabase, ChatDatabase |
| --- | --- |
| Description. | Remove gym buddy from buddy list |
| Preconditions: | 1. Invocation as an inclusion of use case UC015 Login to gym buddy account. 2. Invocation as an inclusion of use case UC019 request to pair up with buddy. 3. There must be an existing match between the user and the suggested buddy. |
| Postconditions: | 1. The app will remove the match from the user’s buddy list and the app database will be updated. 2. The app will also remove the match from the buddy’s user list and the app database will be updated accordingly. 3. The app shall display a message indicating successful removal of buddy. |
| Priority: | Medium.  This function is necessary to enhance overall user experience, allowing users to remove buddies that they do not desire, or feel uncomfortable chatting with. |
| Frequency of Use: | Very Low.  This function will only be invoked if the user chooses to utilise the gym buddy functionality, chooses to request to pair up with buddy and the buddy has done the same, and one of the users wants to remove gym buddy |
| Flow of Events: | 1. The user selects remove buddy. 2. RemoveBuddyUI displays a confirmation message “Are you sure?” 3. The user selects Yes. 4. RemoveBuddyUI sends a message and username to RemoveBuddyManager. 5. RemoveBuddyManager sends a message and username to UserDatabase and ChatDatabase. 6. RemoveBuddyUI displays a message “Buddy has been removed”. |
| Alternative Flows: | AF-S3: The user selects No   1. The use case terminates. |
| Exceptions: | - |
| Includes: | UC015 Login to gym buddy account, UC019 Request to pair up with buddy |
| Special Requirements: | - |
| Assumptions: | The user’s location is kept on throughout. |
| Notes and Issues: | - |

| Use Case ID: | UC023 | | |
| --- | --- | --- | --- |
| Use Case Name: | View all chats | | |
| Created By: | Joshua Khoo | Last Updated By: | Joshua Khoo |
| Date Created: | 31/1/2022 | Date Last Updated: | 13/2/2022 |

| Actor: | RegisteredUser (Initiating Actor), ChatDatabase |
| --- | --- |
| Description. | The user can view all existing chats with the user’s matches. |
| Preconditions: | 1. Invocation as an inclusion of use case UC022 Login to gym buddy account. 2. The user must have an internet connection. |
| Postconditions: | The app shall display a list of all chats with all existing matches the user has. |
| Priority: | Medium.  This function represents the main platform for users to interact with their buddies. |
| Frequency of Use: | High.  This function will be invoked whenever the user wants to chat with other buddies in UC020 or view their buddy list. |
| Flow of Events: | 1. User selects “Buddy List”. 2. BuddyListUI sends Username and signal to ChatManager. 3. ChatManager retrieves chat information from ChatDatabase. 4. BuddyListUI is displayed with a list of chats with all buddies that have been matched.. 5. If the user selects a specific buddy to chat with, the app invokes the extended use case UC020 View chat with buddy |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | UC015 Login to gym buddy account |
| Special Requirements: | - |
| Assumptions: | The user’s location is kept on throughout. |
| Notes and Issues: | - |