Use Cases

for

WITFIT

Version 1.0

Prepared by WomenInTech

WomenInTech (CZ2006 BS3)

1 Apr 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
WomenInTech	01/04/2022	Added use cases to due new requirements churn	1.0

Use Case ID:	UC001		
Use Case Name:	Create main app account		
Created By:	Joshua Khoo	Last Updated By:	Gladys loh
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

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 UC003 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 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. UnregisteredUser selects “I don’t have an account” 2. LoginUI brings Users to SignUpUI. 3. UnregisteredUser enters First Name, Last Name, Email, Birthday, Gender, Username, Password, Confirm Password and clicks the sign up button. 4. UnregisteredUser is prompted for further information which invokes UC003 Login to the main app account.
Alternative Flows:	AF-S3: If the user’s credentials are already in use. 1. App displays the error message “Email already in use“. 2. The app returns to step 2.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	The user’s email address is valid. The cloud service used to host our UserDatabase is operational. The user has a stable internet connection.
Notes and Issues:	-

Use Case ID:	UC002		
Use Case Name:	Enter user details		
Created By:	Joshua Khoo	Last Updated By:	Gladys Loh
Date Created:	13/2/2022	Date Last Updated:	01/04/2022

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.
Postconditions:	1. User has an account created with a username and password. 2. BoardingUI 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. UnregisteredUser presses continue account creation. 2. UserDetailsUI will prompt the UnregisteredUser 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 UnregisteredUser for any physical injuries or pre-existing health conditions. 6. User answers “No” to both questions. 7. UserDetailsUI will prompt the UnregisteredUser to enter their fitness goals. 8. User enters fitness goals and selects next to confirm. 9. UserDetailsManager sends height, weight, fitness goals and menstruation details to UserDatabase. 10. BoardingUI is displayed and displays welcome messages.
Alternative Flows:	AF-S4: If the UnregisteredUser’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 2. AF-S6: If the UnregisteredUser answers “Yes” they have physical injuries. 1. UserDetailsUI will prompt the UnregisteredUser for their physical injuries. 2. UnregisteredUser enter physical injury details. 3. The app will proceed to Step 6.

	AF-S6: If the UnregisteredUser answers “Yes” they have pre-existing medical conditions. 1. UserDetailsUI will prompt the UnregisteredUser 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 UnregisteredUser for their physical injuries. 2. UnregisteredUser will be prompted to enter their type of injury. 3. UserDetailsUI will prompt the UnregisteredUser 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 UnregisteredUser is a female. 1. UserDetailsUI will prompt the UnregisteredUser 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 9.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	The user has a stable internet connection. The included use case UC001 Create main app account is functional.
Notes and Issues:	-

Use Case ID:	UC003		
Use Case Name:	Login to main app account		
Created By:	Joshua Khoo	Last Updated By:	Gladys Loh
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

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:	Critical. Users will not be able to use any feature unless they log 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. RegisteredUser opens the app. 2. LoginUI displays the login page and prompts the user for login details. 3. RegisteredUser enters username and password and presses the login button. 4. LoginUI send username and password to the UserDetailsManager 5. UserDetailsManager validates the username and password using AuthenticationService and logs the RegisteredUser in. 6. BoardingUI is displayed and displays a welcome message
Alternative Flows:	AF-S5: If RegisteredUser'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. The cloud service used to host our UserDatabase is operational.
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:	Gladys Loh
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (Initiating Actor), UserDatabase
Description:	User logs into the app and update personal information
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account. 2. User wants to create a new workout
Postconditions:	1. App stores updated user information in the UserDatabase
Priority:	Low. The user can still use all other functionalities and proceed creating a workout without updating personal information.
Frequency of Use:	High. This function is invoked when the user wants to create a new workout.
Flow of Events:	1. RegisteredUser selects ‘Create Workout’ on WorkoutUI 2. WorkoutUI brings RegisteredUser to UserDetailsUpdaterUI popover 3. UserDetailsUpdaterUI displays prefilled RegisteredUser information 4. UserDetailsUpdaterUI will prompt the RegisteredUser to edit their personal information. 5. RegisteredUsers are allowed to update their height, weight, fitness goals, injuries. 6. Female RegisteredUsers are able to update their last period date. 7. The RegisteredUser selects ‘Next’ after all fields are filled. 8. UserDetailsUpdaterUI calls ValidateAndUpdate to check if there is updated information 9. If true, UserDetailsUpdaterUI calls updateUser to save RegisteredUser information 10. RegisteredUser will be navigated to WorkoutCreatorUI
Alternative Flows:	AF-S6: RegisteredUser selects ‘Next’ 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:	UC003 Login to main app account
Special Requirements:	-
Assumptions:	The included use case UC003 Login to main app account is functional. The cloud service used to host our UserDatabase is operational. The user has a stable internet connection.

Notes and Issues:	-

Use Case ID:	UC005		
Use Case Name:	Recommend personalised workout		
Created By:	Gabriel Tang	Last Updated By:	Gabriel Tang
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	UserDatabase. Initiating actor: NULL
Description:	The app generates a recommended personalised workout for the user.
Preconditions:	1. Invocation as an inclusion of use case UC003 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 creates a workout, a default personalised workout should be recommended to the user.
Flow of Events:	1. On successful login, the RegisteredUser selects to add a workout. 2. WorkoutUI invokes the included Use Case UC004 Update Personal Information to update RegisteredUser's details 3. After returning with the updated RegisteredUser details, WorkoutUI will create the workout via WorkoutCreatorUI. 4. WorkoutCreatorUI calls the WorkoutRecommender to get the recommended workout. 5. WorkoutRecommender retrieves the RegisteredUser's injuries from the UserDatabase. 6. WorkoutRecommender retrieves the RegisteredUser 's goals from the UserDatabase. 7. WorkoutRecommender returns the list of exercises to the WorkoutCreatorUI. 8. 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:	UC003 Login to main app account
Special Requirements:	-
Assumptions:	The included use case UC003 Login to main app account is functional. The cloud service used to host our UserDatabase, and the Exercise API, is operational.

	The user has a stable internet connection.
Notes and Issues:	-

Use Case ID:	UC006		
Use Case Name:	Conduct workout		
Created By:	Bryan Leow	Last Updated By:	Gladys Loh
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (initiating actor), UserDatabase
Description:	Facilitates the overall flow of a workout routine.
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account. 2. User has already created a workout with valid exercises. 3. This use case extends UC011 View completed workouts
Postconditions:	1. 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. RegisteredUser will select a workout from their list of workouts and click “Start”. 2. WorkoutUI receives the RegisteredUser selection and passes the Workout ID to the WorkoutManager to get the workout and check the status. 3. WorkoutConductorUI will display the exercise based on workout status and workout history. 4. WorkoutConductorUI displays a timer. 5. WarmUpUI will call YoutubeService for a warm up video. 6. Once warm up is completed, RegisteredUser can proceed to do exercises. 7. While there is an exercise for the RegisteredUser to currently perform, the WorkoutConductorUI will retrieve the exercise from the WorkoutManager. 8. RegisteredUsers are able to add or delete sets before moving on to the next exercise. 9. WorkoutConductorUI will update the exercise progress for the current exercise via the Workout Manager. 10. WorkoutManager will mark the exercise as done in the Exercise and move on to the next exercise. 11. RegisteredUser 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 display CoolDownUI. 14. CoolDownUI will call YoutubeService for a cool down video.

	15. WorkoutManager will invoke UC008 Display workout summary to display workout summary.
Alternative Flows:	AF-S1: When the RegisteredUser adds more sets 1. RegisteredUser selects ‘Add more sets’ on the WorkoutConductorUI. 2. WorkoutConductorUI updates the workout via the Workout Manager. 3. Workout Manager updates the WorkoutConductorUI, and adds set to the Exercise. 4. The app returns to step 9. AF-S2: If RegisteredUser presses ‘Pause workout’ 1. WorkoutConductorUI pauses the current workout via Workout Manager. 2. RegisteredUsers will be prompted if they want to continue the workout. 3. If the RegisteredUser presses ‘Resume workout’, WorkoutConductorUI resumes the workout and returns to step 11. 4. Otherwise, there will be another prompt for the RegisteredUser to stop the workout. 5. If the RegisteredUser presses ‘Stop workout’, the WorkoutConductorUI stops the workout and returns to WorkoutUI 6. WorkoutConductorUI resumes the workout and returns to step 11.
Exceptions:	EX 1: The Youtube API is down, or the API key is invalid 1. YoutubeService will return a default placeholder warm up video. 2. The app returns to step 5. EX 1: The Youtube API is down, or the API key is invalid 1. YoutubeService will return a default placeholder cool down video. 2. The app returns to step 14.
Includes:	UC003 Login to main app account, UC008 Display workout summary
Special Requirements:	-
Assumptions:	The included use cases UC003 Login to main app account, and UC008 Display workout summary are functional. The cloud service used to host our UserDatabase is operational. The user has a stable internet connection.
Notes and Issues:	-

Use Case ID:	UC007		
Use Case Name:	Edit workout routine		
Created By:	Gladys Loh	Last Updated By:	Gabriel Tang
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (Initiating actor), Exercise API, UserDatabase
Description:	Users are able to edit their recommended routine.
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account.
Postconditions:	1. The app will update the workout routine according to what the user specified to the UserDatabase.
Priority:	Medium. This is a main functionality - allowing users to edit their sets and reps based on their physical capabilities when performing the workout.
Frequency of Use:	Medium. This function will only be invoked if users want to edit their customised routine.
Flow of Events:	1. RegisteredUser selects workout to be edited. 2. WorkoutEditorUI is shown and RegisteredUser can edit reps and set. 3. RegisteredUser confirms changes to be updated and WorkoutPlanner will save information to the UserDatabase. 4. WorkoutUI will display their updated workout and the app will be navigated to UC0101 Display Created Workouts.
Alternative Flows:	AF-S3: If information is incomplete 1. The WorkoutEditorUI will prompt the RegisteredUser that information entered is not valid and will allow the RegisteredUser to re-enter. 2. Once information is complete, Workout Planner will save information to the UserDatabase. 3. The app returns to step 4. AF-S3: If the RegisteredUser presses “cancel”. 1. The RegisteredUser will remain on WorkoutEditorUI. 2. The app returns to step 2.
Exceptions:	-
Includes:	UC003 Login to main app account
Special Requirements:	-
Assumptions:	The included use case UC003 Login to main app account is functional. The cloud service used to host our UserDatabase is operational.
Notes and Issues:	-

Use Case ID:	UC008		
Use Case Name:	Display workout summary		
Created By:	Foo Zhi Kai	Last Updated By:	Gladys Loh
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	UserDatabase. Initiating actor: NULL
Description:	Display summary of completed workout
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account. 2. Invocation as an inclusion of use case UC006 Conduct workout.
Postconditions:	1. 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 RegisteredUser uses the included use case UC006 Conduct workout to start conducting a workout. 2. The RegisteredUser 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 burnt, fitness and strength. 6. WorkoutSummaryUI will also display some basic statistics like total sets and reps.
Alternative Flows:	-
Exceptions:	-
Includes:	UC003 Login to main app account, UC006 Conduct workout
Special Requirements:	-
Assumptions:	1. The included use case UC003 Login to main app account, and UC006 Conduct workout, are functional. 2. The cloud service used to host our UserDatabase is operational. 3. The user has a stable internet connection.
Notes and Issues:	The timestamps are based on Singapore Time (GMT +8).

Use Case ID:	UC009		
Use Case Name:	Display weekly statistics		
Created By:	Foo Zhi Kai	Last Updated By:	Bryan Leow
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	UserDatabase, RegisteredUser. Initiating actor: NULL
Description:	Displays summary statistics of the workouts completed for the week.
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account.
Postconditions:	1. App displays total calories burnt and total duration spent exercising 2. App displays a line chart showing the duration spent exercising over the past week.
Priority:	High. This functionality will live on a prominent location, such as the Homepage, where it is regularly exposed to users.
Frequency of Use:	High. A key component of the user experience and motivates the user to continue using the app.
Flow of Events:	1. On successful login, RegisteredUser is brought to the app's HomepageUI. 2. HomepageManager retrieves information about RegisteredUser's completedWorkouts from UserService. 3. HomepageManager filters the list of completedWorkouts down to those completed in the past week. 4. HomepageManager calculates the total calories burnt and duration spent exercising across the week. 5. HomepageUI displays total calories and duration on-screen as numbers. 6. HomepageUI displays duration across the week as a line chart. 7. HomepageUI toggles between these two displays automatically.
Alternative Flows:	AF-S7: If RegisteredUser swipes 1. HomepageUI toggles between these two displays manually. The automatic display stops. AF-S8: RegisteredUser clicks on 'See More' 1. UC011 View completed workouts is invoked.
Exceptions:	-
Includes:	UC003 Login to main app account,
Special Requirements:	-
Assumptions:	1. The included use case UC003 Login to main app account is functional.

	2. The cloud service used to host our UserDatabase is operational. 3. The user has a stable internet connection.
Notes and Issues:	The timestamps are based on Singapore Time (GMT +8).

Use Case ID:	UC010		
Use Case Name:	Display created workouts		
Created By:	Joshua Khoo	Last Updated By:	Gladys Loh
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (Initiating Actor), UserDatabase
Description:	User is able to access and view past workouts
Preconditions:	1. This use case extends UC007 Edit workout routine and UC006 Conduct workout
Postconditions:	1. The app must display a history of all previous workouts 2. The app shall allow the user to filter between the different display options.
Priority:	Medium. A key component for users to track and view all their workouts that they have created. This function is an important motivating force for them to continue using the app.
Frequency of Use:	High. The created workouts will be displayed whenever the user has created workouts. Users will come to this page when they want to redo their workouts, create a new workout, delete a workout or edit a workout.
Flow of Events:	1. RegisteredUser navigates to the WorkoutUI. 2. WorkoutManager will retrieve information regarding the RegisteredUser's created workouts from UserDatabase using their UserID. 3. WorkoutManager will display a list of all workouts, with their title, description and status of their workout. 4. RegisteredUser is able to filter their workouts based on completion, in progress or shared by a buddy.
Alternative Flows:	-
Exceptions:	EX1 RegisteredUser has not done any workouts 1. The app will display a message "No workouts created". 2. The app will prompt them to create a workout. 3. If the RegisteredUser selects the option to create a workout, the app will return to UC005 Recommend personalised workout
Includes:	UC003 Login to main app account
Special Requirements:	-
Assumptions:	The included use case UC003 Login to main app account is functional. The cloud service used to host our UserDatabase is operational. The user has a stable internet connection.
Notes and Issues:	The timestamps are based on Singapore Time (GMT +8).

Use Case ID:	UC011		
Use Case Name:	View completed workouts		
Created By:	Bryan Leow	Last Updated By:	Bryan Leow
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (initiating actor), UserDatabase
Description:	User is able to view past workouts
Preconditions:	1. This use case extends UC009 Display weekly statistics.
Postconditions:	1. App displays list of workouts completed in the past week and their details
Priority:	Medium. This allows users to track their past workouts in a higher level of detail than what the Homepage statistics already provide.
Frequency of Use:	Medium. This functionality is directly accessible from the Homepage, but only if users choose to click the button that navigates to this page.
Flow of Events:	1. RegisteredUser navigates to the CompletedWorkoutsUI. 2. CompletedWorkoutsManager retrieves the array of workouts completed this week from the HomepageManager. 3. CompletedWorkoutsManager parses the information in the array. 4. CompletedWorkoutsUI displays this parsed information on-screen.
Alternative Flows:	AF-S5: RegisteredUser presses back button 1. App goes back to UC009 Display weekly statistics
Exceptions:	EX1 RegisteredUser has not done any workouts 1. CompletedWorkoutsUI displays a ‘No completed workouts’ message. 2. CompletedWorkoutsUI displays a button to navigate to the Workouts page, invoking UC006 Conduct workout.
Includes:	UC003 Login to main app account
Special Requirements:	-
Assumptions:	1. The included use case UC003 Login to main app account is functional. 2. The cloud service used to host our UserDatabase is operational. 3. The user has a stable internet connection.
Notes and Issues:	The timestamps are based on Singapore Time (GMT +8).

Use Case ID:	UC012		
Use Case Name:	Display recommended videos		
Created By:	Bryan Leow	Last Updated By:	Bryan Leow
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (initiating actor), UserDatabase
Description.	User is able to view a list of Youtube videos recommended to them.
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account. 2. An internet connection is required. 3. An established connection with the Youtube API endpoint is required.
Postconditions:	1. App displays the recommended videos on the homepage as a carousel.
Priority:	Medium. This functionality will live on a prominent location, such as the Homepage, where it is regularly exposed to users.
Frequency of Use:	Medium. Allows users to improve their fitness knowledge, helping them in their fitness journey.
Flow of Events:	1. On successful login, RegisteredUser is brought to the app's HomepageUI. 2. HomepageManager retrieves information about RegisteredUser's latest completedWorkout from UserDatabase. 3. HomepageManager extracts completedWorkout's workout name, exercise names, and the RegisteredUser's gender. 4. HomepageManager calls YoutubeService to search Youtube for these exercise names + gender. 5. YoutubeService compiles the recommended videos and returns them to HomepageManager. 6. HomepageUI displays this information on-screen.
Alternative Flows:	AF-S3: RegisteredUser does not have any completed workout 1. HomepageManager will extract RegisteredUser's workout goal, gender, and area of injury. 2. If gender is 'Others', HomepageManager will call YoutubeService to search Youtube for 'exercise for beginners', and 'exercise for ' + RegisteredUser's workout goal. 3. Else, HomepageManager will call YoutubeService to search Youtube for 'exercise for beginners'+gender, and 'exercise for'+ RegisteredUser's workout goal+gender. 4. If RegisteredUser has an area of injury, HomepageManager will call YoutubeService to search Youtube for 'exercise for injury'+injury. 5. The app returns to step 5. AF-S4: RegisteredUser's gender is 'Other'

	1. HomepageManager will not include user’s gender in the search term when calling YoutubeService. 2. The app returns to step 5.
Exceptions:	EX 1: The Youtube API is down, or the API key is invalid 3. YoutubeService will return a default placeholder recommendation video. 4. The app returns to step 6.
Includes:	UC003 Login to main app account.
Special Requirements:	-
Assumptions:	1. The included use case UC003 Login to main app account is functional. 2. The cloud service used to host our UserDatabase is operational. 3. The user has a stable internet connection.The user has a stable internet connection.
Notes and Issues:	-

Use Case ID:	UC013		
Use Case Name:	Create gym buddy account		
Created By:	Joshua Khoo	Last Updated By:	Foo Zhi Kai
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (Initiating Actor), ImageCollection, 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.
Postconditions:	1. The app will store the user's gym buddy data in the UserDatabase. 2. 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. GymBuddySignUpPageUI prompts RegisteredUser 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. On selecting the browse button, GymBuddySignUpPageUI sends a signal to GymBuddySignUpPageManager to allow registeredUsers the ability to select an image from their device. 3. GymBuddySignUpPageManager converts the image and begins uploading the image to the ImageCollection. 4. GymBuddySignUpPageManager sends a signal about the progress of the upload to the GymBuddySignUpPageUI to be displayed. 5. Upon successful upload, GymBuddySignUpPageManager updates the UserDatabase with the link of the image stored in the Image Collection. 6. GymBuddySignUpPageManager sends a signal to the GymBuddySignUpPageUI along with the file size of the image. 7. The RegisteredUser fills up the remaining information fields and clicks on submit. 8. GymBuddySignUpPageUI sends new gym buddy info to GymBuddySignUpPageManager

	9. GymBuddySignUpPageManager sends new gym buddy info to GymBuddyService 10. GymBuddyService creates and sends new GymBuddyDetails to UserDatabase. 11. GymBuddyService sends a signal and GymBuddySignUpPageManager. 12. GymBuddySignUpPageManager calls UC014 Display gym buddy home page, indicating successful login.
Alternative Flows:	AF-S2: RegisteredUser does not choose to select the browse button to upload an image. 1. GymBuddySignUpPageUI sends a signal to GymBuddySignUpPageManager. 2. GymBuddySignUpPageManager sends a signal to the GymBuddyService along with the UserID 3. GymBuddyService updates the UserDatabase with a default image link based on the gender of the RegisteredUser. 4. The app returns to step 7. AF-S7: RegisteredUser clicks on create buddy profile when not all fields are filled. 1. GymBuddySignUpPageUI will not allow RegisteredUser to select the submit button. 2. The app returns to step 7.
Exceptions:	-
Includes:	UC003 Login to main app account.
Special Requirements:	-
Assumptions:	1. The RegisteredUser has unlimited tries to fill in all the fields. 2. The included use case UC003 Login to main app account is functional. 3. The cloud service used to host our UserDatabase is operational. 4. The user has a stable internet connection.
Notes and Issues:	-

Use Case ID:	UC014		
Use Case Name:	Display gym buddy home page		
Created By:	Joshua Khoo	Last Updated By:	Joshua Khoo
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (Initiating Actor), UserDatabase
Description.	Display the gym buddy home page with options to explore gym buddy functionalities.
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account. 2. The user must have signed up for a gym buddy account.
Postconditions:	1. The app will display the user's gym buddy profile with options to navigate.
Priority:	High. Use cases UC015 Update gym buddy preferences, UC016 View suggested buddy Details, UC017 Request to pair up with buddy, UC018 View all chats, UC019 View chat with buddy, UC020 Send message, UC021 Remove buddy, UC022 Share Workout will not be available without implementing this functionality.
Frequency of Use:	High. This use case will always be used as a starting point before the user can navigate to access all gym buddy functionalities.
Flow of Events:	1. RegisteredUser selects the "Gym Buddy" tab. 2. GymBuddyHomePageUi sends a signal to GymBuddyHomePageController to get RegisteredUser details. 3. GymBuddyHomePageController retrieves current RegisteredUser details from the DbRetrieveService which fetches information from the UserDatabase. 4. The GymBuddyHomePageController checks if the RegisteredUser has created a gym buddy account. 5. If the RegisteredUser has created a gym buddy account, the gym buddy home page controller sets the RegisteredUser details and sends the details to GymBuddyHomePageUi. 6. GymBuddyHomePageUi displays user details to the RegisteredUser .
Alternative Flows:	AF-S5: GymBuddyHomePageController checks that the RegisteredUser has not created a gym buddy account. 1. The GymBuddyHomePageController will redirect the RegisteredUser to UC013 Create gym buddy account
Exceptions:	-
Includes:	UC003 Login to main app account
Special Requirements:	-

Assumptions:	1. The user has a stable internet connection throughout. 2. The included use case UC003 Login to main app account. 3. The cloud service used to host our UserDatabase is operational.
Notes and Issues:	-

Use Case ID:	UC015		
Use Case Name:	Update gym buddy preferences		
Created By:	Joshua Khoo	Last Updated By:	Foo Zhi Kai
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (Initiating Actor), ImageCollection, UserDatabase
Description.	Update user's account preference
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account 2. This use case extends UC014 Display gym buddy home page. 3. The user must have signed up for a gym buddy account.
Postconditions:	1. The app will store the user's gym buddy data in the UserDatabase. 2. The user will be interfaced to the homepage, indicating successful update.
Priority:	Medium. Users will be able to update their gym buddy details, allowing the algorithm to provide them with a more accurate match.
Frequency of Use:	Low. Unlikely that the user will update their account details often.
Flow of Events:	1. GymBuddyUpdateAccountPreferenceUI prompts RegisteredUser 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. On selecting the browse button, GymBuddyUpdateAccountPreferenceUI sends a signal to GymBuddyUpdateAccountPreferenceManager to allow RegisteredUsers the ability to select an image from their device. 3. GymBuddyUpdateAccountPreferenceManager converts images and begins uploading the image to the ImageCollection. 4. GymBuddyUpdateAccountPreferenceManager sends a signal about the progress of the upload to the GymBuddyUpdateAccountPreferenceUI to be displayed. 5. Upon successful upload, GymBuddyUpdateAccountPreferenceManager updates the UserDatabase with the link of the image stored in the Image Collection. 6. GymBuddyUpdateAccountPreferenceManager sends a signal to the GymBuddyUpdateAccountPreferenceUI along with the file size of the image.

	7. The RegisteredUser fills up the remaining information fields and clicks on submit. 8. GymBuddyUpdateAccountPreferenceUI sends new gym buddy info to GymBuddyUpdateAccountPreferenceManager 9. GymBuddyUpdateAccountPreferenceManager sends new gym buddy info to GymBuddyService 10. GymBuddyService creates and sends new GymBuddyDetails to UserDatabase. 11. GymBuddyService sends a signal and GymBuddyUpdateAccountPreferenceManager . 12. GymBuddyUpdateAccountPreferenceManager calls UC014 Display gym buddy home page, indicating successful update.
Alternative Flows:	AF-S2: RegisteredUser does not choose to select the browse button to upload an image. 5. GymBuddyUpdateAccountPreferenceUI sends a signal to GymBuddyUpdateAccountPreferenceManager . 6. GymBuddyUpdateAccountPreferenceManager sends a signal to the GymBuddyService along with the UserID 7. GymBuddyService updates the UserDatabase with a default image link based on the gender of the RegisteredUser. 8. The app returns to step 7. AF-S7: RegisteredUser clicks on create buddy profile when not all fields are filled. 3. GymBuddyUpdateAccountPreferenceUI will not allow RegisteredUser to select the submit button. 4. The app returns to step 7.
Exceptions:	-
Includes:	UC003 Login to main app account
Special Requirements:	-
Assumptions:	The user must have an internet connection. The included use case UC003 Login to main app account is functional. The cloud service used to host our UserDatabase is operational.
Notes and Issues:	-

Use Case ID:	UC016		
Use Case Name:	View suggested buddy details		
Created By:	Joshua Khoo	Last Updated By:	Joshua Khoo
Date Created:	13/2/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (Initiating Actor), UserDatabase
Description.	View one suggested buddy
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account 2. This use case extends UC014 Display gym buddy home page. 3. The user must have signed up for a gym buddy account.
Postconditions:	1. 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 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:	Medium. This function will only be invoked if the user chooses to utilise the gym buddy functionality to search for buddies.
Flow of Events:	1. RegisteredUser clicks find a buddy. 2. The FindBuddyBoardingPageUi sends a signal of RegisteredUser click to the FindBuddyBoardingPageController. 3. The FindBuddyBoardingPageController retrieves current user information from the DbRetrieveService. 4. The FindBuddyBoardingPageController retrieves a dictionary of gym buddy profiles filtered by the FindBuddyQuery which fetches all the profiles from the UserDatabase. 5. The FindBuddyBoardingPageController sends a signal to the RecommendationEngine to get all potential matches. 6. The RecommendationEngine sends a signal to the MatchmakingAlgo which calculates the matching score between the RegisteredUser and all other profiles. 7. The RecommendationEngine then polls the score of all users and arranges the scores in decreasing order (highest score first) in an array of potential matches. 8. The RecommendationEngine sends the array of potential matches to the GymBuddyService which stores the information.

	9. The GymBuddyService then sends a signal to the FindBuddyBoardingPageController of completion of the matching algorithm. 10. The FindBuddyBoardingPageController sends a signal to the FindBuddyBoardingPageUi which displays a confirmation message to the RegisteredUser. 11. The RegisteredUser clicks “Lets get swiping” and is routed to the FindBuddyPageUi. 12. The FindBuddyPageUi sends a signal to FindBuddyPageController which retrieves the potential matches array from the GymBuddyService. 13. The GymBuddyService sends the array of potential matches to the GymBuddyPageController. 14. The GymBuddyPageUi then displays the suggested buddy profile with the highest score to the RegisteredUser.
Alternative Flows:	AF-14: If there are no more suggested profiles to display. 1. The app will display a blank screen indicating that there are no more profiles to suggest.
Exceptions:	-
Includes:	UC003 Login to main app account
Special Requirements:	There must be at least 1 other user registered for gym buddy in order for this functionality to work.
Assumptions:	1. The user must have an internet connection. 2. The included use case UC003 Login to main app account is functional. 3. The cloud service used to host our UserDatabase is operational.
Notes and Issues:	1. The FindBuddyDisplayUi will continuously update to recommend the next potential match when users swipe either left or right as in UC017 Request to pair up with buddy until there are no more matches to recommend to the RegisteredUser. 2. PreferredGender of the RegisteredUser is a key field, users will only be displayed matches of their preferred gender. Example: If the RegisteredUser preferred gender is Male, you will only be shown Male profiles, likewise only Males will be able to see the profile of the RegisteredUser.

Use Case ID:	UC017		
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:	01/04/2022

Actor:	RegisteredUser (Initiating Actor), UserDatabase, ChatDatabase
Description.	The user selects to pair up with the suggested buddy.
Preconditions:	1. This use case extends UC016 View suggested buddy details 2. Invocation as an inclusion of use case UC003 Login to main app account 3. The user must have signed up for a gym buddy account.
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 UC016 View suggested buddy details.
Flow of Events:	1. The RegisteredUser swipes to the right, indicating a match. 2. FindBuddyUI sends a signal to FindBuddyManager . 3. FindBuddyManager checks if the recommended RegisteredUser had previously matched with the current RegisteredUser. 4. FindBuddyManager validates the match. 5. FindBuddyManager sends a signal to FindBuddyQueryManager to update the current RegisteredUser information and to create a new chat. 6. FindBuddyQueryManager sends a signal to DBRetrieveService to update the current RegisteredUser information and create a new chat. 7. DBRetrieveService sends a signal to ChatDatabase to create a chat. 8. DBRetrieveService sends a signal to UserDatabase to update current RegisteredUser and recommended RegisteredUser information. 9. FindBuddyManager sends a signal to the FindBuddyUI. 10. FindBuddyUI displays the next available buddy.

Alternative Flows:	AF-S4 If the recommender RegisteredUser has yet to match with the current RegisteredUser. 1. BuddyValidationManager sends a signal to the FindBuddyUI. 2. FindBuddyUI displays the next available buddy.
Exceptions:	-
Includes:	UC003 Login to main app account
Special Requirements:	-
Assumptions:	1. The user has a stable internet connection. 2. The included use case UC003 Login to main app account is functional. 3. The cloud service used to host our UserDatabase and ChatDatabase is operational.
Notes and Issues:	-

Use Case ID:	UC018		
Use Case Name:	View all chats		
Created By:	Joshua Khoo	Last Updated By:	Joshua Khoo
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (Initiating Actor), ChatDatabase, UserDatabase
Description.	The user can view all existing chats with the user's matches.
Preconditions:	1. This use case extends UC014 Display gym buddy home page. 2. The user must have signed up for a gym buddy account. 3. Invocation as an inclusion of use case UC003 Login to main app account
Postconditions:	1. The app shall display a list of all chats with all existing matches the user has. For each chat, the user must be able to see the buddy's profile picture, name, last message and a timestamp indicating the time elapsed since the last message.
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 Send message and UC019 View chat with buddy
Flow of Events:	1. The RegisteredUser selects 'View Buddy List'. 2. The BuddyListHomePageUi sends signal of user click to BuddyListHomePageController. 3. The BuddyListHomePageController retrieves current chat user, all chat information from the ChatService which gets information from the ChatDatabase. 4. The BuddyListHomePageUi then retrieves the full name, profile picture, last message and timestamp from the BuddyListHomePageController. 5. The BuddyListHomePageUi then displays all the chats to the RegisteredUser.
Alternative Flows:	-
Exceptions:	-
Includes:	UC013 Create gym buddy account
Special Requirements:	-
Assumptions:	The RegisteredUser must have a stable internet connection. The included use case UC003 Login to main app account is functional.

	The cloud service used to host our UserDatabase and ChatDatabase is operational.
Notes and Issues:	The timestamps are based on Singapore Time (GMT +8).

Use Case ID:	UC019		
Use Case Name:	View chat with buddy		
Created By:	Joshua Khoo	Last Updated By:	Joshua Khoo
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (Initiating Actor), ChatDatabase, UserDatabase
Description.	The user chooses to click into a specific chat from the list of all chats.
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account 2. There must be an existing match between the user and the suggested buddy. 3. This use case extends UC018 View all chats. 4. The user must have already signed up for a gym buddy account.
Postconditions:	1. The app will consistently update chat information into the ChatDatabase. 2. The app must display chat messages sent by the user onto the buddy’s chat display. 3. 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:	Medium. 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. RegisteredUser clicks view chat. 2. The ChatUi sends a signal of user click to the ChatController. 3. The ChatUi subscribes to the ChatController and constantly gets updated messages from the ChatController.

	4. The Chat Controller subscribes to the ChatService to constantly get updated chat messages from the ChatDatabase. 5. The ChatService returns all chat messages to the ChatController which sends the chat messages to the ChatUi which displays the chat messages to the RegisteredUser. 6. The ChatService notifies the ChatController whenever there is a chat message written to the database which updates the ChatUi which then displays the latest chat message to the RegisteredUser.
Alternative Flows:	-
Exceptions:	-
Includes:	UC003 Login to main app account
Special Requirements:	-
Assumptions:	1. The users of the chat must have a stable and fast internet connection. 2. The included use case UC003 Login to main app account, is functional. 3. The cloud service used to host our ChatDatabase and UserDatabase is operational.
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:	UC020		
Use Case Name:	Send message		
Created By:	Joshua Khoo	Last Updated By:	Joshua Khoo
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (PrimaryUser), ChatDatabase, SecondaryUser
Description.	The user chooses to send a message in a chat room with another buddy.
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account 2. There must be an existing match between the user and the suggested buddy. 3. This use case extends UC019 View chat with buddy. 4. The user must have already signed up for a gym buddy account.
Postconditions:	1. The app will consistently update chat information into the ChatDatabase. 2. The app must display chat messages sent by the user onto the buddy's chat display. 3. 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. This use case is necessary, else it defeats the purpose of having the chat functionality altogether.
Frequency of Use:	Medium. 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. The RegisteredUser sends a chat message to the secondary user. 2. The ChatUi notifies the ChatController of the message sent. 3. The ChatService adds the chat message to the ChatDatabase. 4. The ChatUi subscribes to any update in chat messages and displays the message sent by the PrimaryUser. 5. The ChatService constantly listens for writes to database and returns an observable to both the PrimaryUser ChatUi and SecondaryUser ChatUi. 6. The SecondaryUser's ChatUi which subscribes to the ChatService constantly listens for updates and displays the new chat message sent by the PrimaryUser.

Alternative Flows:	-
Exceptions:	-
Includes:	UC003 Login to main app account
Special Requirements:	-
Assumptions:	1. For messages to be received and sent concurrently, both the user and the buddy must have a stable internet connection. 2. The included use case UC003 Login to main app account, is functional. 3. The cloud service used to host our ChatDatabase is operational.
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:	Remove buddy		
Created By:	Joshua Khoo	Last Updated By:	Joshua Khoo
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (Initiating Actor), UserDatabase, ChatDatabase
Description.	The user can remove a buddy from their match list.
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account 2. There must be an existing match between the user and the suggested buddy. 3. This use case extends UC019 View chat with buddy. 4. The user must have already signed up for a gym buddy account.
Postconditions:	1. The app will remove the match from the user's buddy list and the UserDatabase will be updated. 2. The app will also remove the match from the buddy's user list and the UserDatabase will be updated accordingly. 3. The Chat between both users will also be deleted and the ChatDatabase will be updated accordingly. 4. The app shall return to the UC018 View all chats with the display excluding the chat of the removed buddy. 5. The app shall not suggest the removed buddy profile to the RegisteredUser in UC016 View suggested buddy details
Priority:	Low. This function is only 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 RegisteredUser clicks the remove buddy icon. 2. The ChatUi displays an option list. 3. The RegisteredUser confirms to proceed to remove buddy. 4. The ChatUi sends a signal to the ChatController of the RegisteredUser's choice. 5. The ChatController fetches the chat reference ID from the ChatService. 6. The Chat Service then deletes the match from both RegisteredUsers and updates the UserDatabase accordingly.

	7. The Chat Service then deletes the chat between both RegisteredUsers and updates the ChatDabase accordingly. 8. After removal is completed, a return signal is sent and the ChatUi returns to UC018 View All Chats which displays the chat list excluding the chat which has been removed.
Alternative Flows:	AF-S3: The RegisteredUser selects No 1. The use case terminates. 2. The app returns to UC019 View Chat with Buddy.
Exceptions:	-
Includes:	UC003 Login to main app account
Special Requirements:	-
Assumptions:	1. There is a stable internet connection. 2. The included use case UC003 Login to main app account, is functional. 3. The cloud service used to host our UserDatabase is operational.
Notes and Issues:	-

Use Case ID:	UC022		
Use Case Name:	Share Workout		
Created By:	Joshua Khoo	Last Updated By:	Joshua Khoo
Date Created:	31/1/2022	Date Last Updated:	01/04/2022

Actor:	RegisteredUser (Initiating Actor), ChatDatabase, UserDatabase
Description.	The user can choose to share their workouts with a buddy.
Preconditions:	1. Invocation as an inclusion of use case UC003 Login to main app account 2. There must be an existing match between the user and the suggested buddy. 3. This use case extends UC019 View chat with buddy. 4. The user must have already signed up for a gym buddy account.
Postconditions:	The secondary user should be able to see the workout shared by the RegisteredUser in the UC010 Display created workouts
Priority:	Medium. This function integrates the gym buddy functionality of our application with the workout functionality and is a core feature to allow users to share a common goal of being fit together.
Frequency of Use:	Low. This function will be invoked when a user wants to share a workout with a buddy.
Flow of Events:	1. The RegisteredUser clicks on the share workout button in the Chat interface. 2. The ChatUi displays an option list to the RegisteredUser. 3. The RegisteredUser confirms to proceed to share workout. 4. The ChatUi sends a signal to the ChatController which then notifies the ChatService to update the shared workouts. 5. The ChatService extracts the workouts collection from the SecondaryUser and extracts the workouts collection from the RegisteredUser and filters for duplicates. 6. The ChatService then stores the non-duplicate workouts from the RegisteredUser into the collection of the secondary user. 7. The Chat service notifies the ChatController once the workout is shared. 8. The ChatUi is then updated by the ChatController of the status and the pop up is dismissed. 9. The ChatUi then returns to the chat display as in UC019 View chat with buddy
Alternative Flows:	AF-S3: The RegisteredUser selects No 1. The use case terminates and goes to step 9.
Exceptions:	-

Includes:	UC003 Login to main app account
Special Requirements:	-
Assumptions:	The RegisteredUser must have a stable internet connection. The included use case UC003 Login to main app account is functional. The cloud service used to host our UserDatabase is operational.
Notes and Issues:	-