Muhammad Khairul Azman - Project Portfolio

PROJECT: DukeCooks 👨‍🍳

What can I expect from this document?

This document presents forth a project portfolio chronicling the my contribution to the Software Engineering project DukeCooks. It will contain also contain extracts of my segments in the User and Developer Guides of DukeCooks.

What exactly is DukeCooks?

DukeCooks is a one-stop healthy lifestyle application that helps you manage your tasks, plan your meals, run workout regimes, monitor your health records and create custom blog pages for sharing. It caters to a high performing, health-conscious individuals who wish to save the time spent on managing their health.

While we concede that there are other applications in the market who can do each of our features better, we take pride in being the only application that covers all the grounds when it comes to health in a concise and seamless manner. This would allow our target users to save the time and effort of having to switch and log in multiple applications.

DukeCooks is also built on a Command-Line Interface (CLI), allowing everything to be done without a mouse and reducing the need to ever move your wrists from the keyboard. So what are you waiting for high performing individual? Take a break from this portfolio and give DukeCooks a go! I'll be waiting.

What created DukeCooks?

Now that you're back, I can finally tell you about the wonderful team behind the application. DukeCooks is developed by a team of Year 2 Undergraduates from National University College as part of a software engineering module. Having 2 members who stayed in campus, it was originally designed to be a recipe book that can schedule your meal plans to ease shopping. However we realised the potential that the project could realise since a huge part of health starts in eating healthy.

We thus added various extensions integral to a healthier lifestyle while retaining the core feature and name, DukeCooks, allowing it to become what it is today.

My Contributions to DukeCooks

Creating the Workout Planner

The Workout Planner in Duke Cooks is used to create your own custom workout regimes. These workouts contains exercises that are built into the application or can be created by the users. Users can then run the workout routine with run command.

The management of Workouts and Exercises is done through Commands similar to that of the AddressBook3, the application DukeCooks was built on. These Commands include the add, clear, find, list, delete and edit Commands/

Enhancement 1: Added the ability to find exercises using MuscleType and Intensity as parameters

The initial find function can be used to filter the exercise list by name. However, users may be interested in creating a workout of only on level of intensity or targeting one muscle group. By entering `find exerciseMuscle <MuscleType>` or `find exerciseIntensity <Intensity>`, users can view muscle/intensity specific exercises.

Enhancement 2: Adding push exercise functionality

When workouts are first added, they are empty and do not contain any exercises. Hence to populate the workout, you can use the `push exercise` command to add an exercise into the workout. The application will then automatically update all the fields in workout to reflect the new addition in exercise.

Enhancement 3: Adding the run workout functionality

What good is a workout if you cannot run it? After filling your workout with all the exercises you plan to do, you can then execute the workout with the `run workout` command. A new window will open, showing all the details you need for each set of exercises as well as a progress bar to keep track of your completion.

Upon completion you will be greeted with a congratulatory message and the workout records will be updated accordingly.

Enhancement 4: Keeping track of workout progress

After each run, the history of each Workout and Exercises will be automatically updated and can seen using the `view exercise/workout` command. This includes statistics such as how long the run went on for as well as list of all the previous time you did the workout/exercise.

Code contributed:

Visit this link to check out my code contributions to DukeCooks.

Other contributions:

	What I did
Project management	Created Labels to be used for issues Instantiated the Milestones in issues
Reviewed PRs	PRs reviewed (with non-trivial review comments): #44, #55
Reported bugs	Reported bugs and suggestions for other teams in the class (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)

User Guide Contributions

The following information are my **contributions** to the **User Guide** for the feature, WorkoutPlanner.
To view the full **User Guide** of DukeCooks, please visit this link.

Workout Tracker

So you've decided to put your gym membership to use and want to plan your next workout regime. Head down to the workout planner tab to create your custom workouts and track your progress!

Workout Planner Screen

Alright, I'm now at the workout screen! What's next?

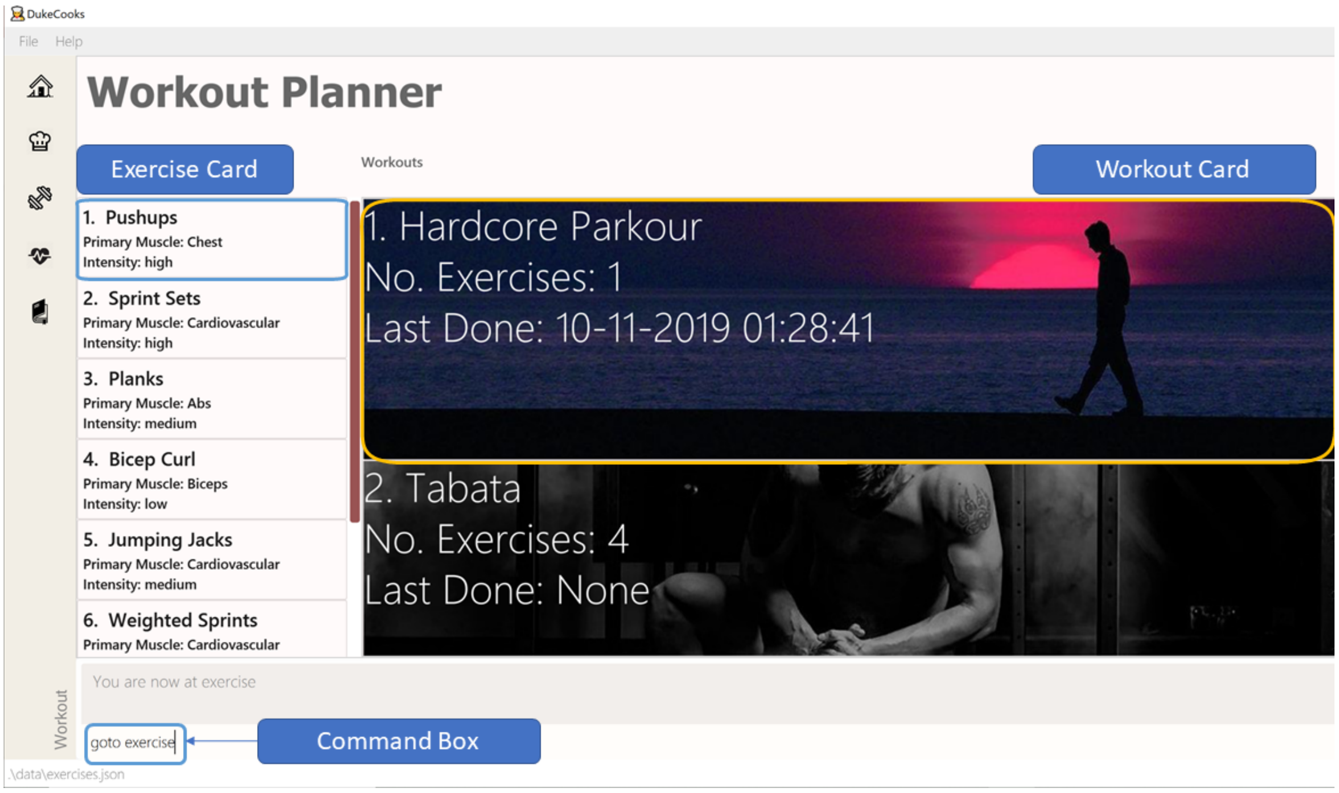


Figure 1. Workout Screen

As seen in the screenshot above, the Workout Planner screen is similar to the other screens, with the main difference being the list of Exercise cards and Workout Cards. Not too sure about what to work on or need suggestions? Workout Planner begins with prebuilt Exercises and Workouts installed to get you started.

Over on the left, is the list of Exercise Cards. Each card informs you of the name, primary muscle as well as the intensity of the exercise. To get more information of the exercise, you can use the view command which will be covered in section 3.7.3 of the User Guide.

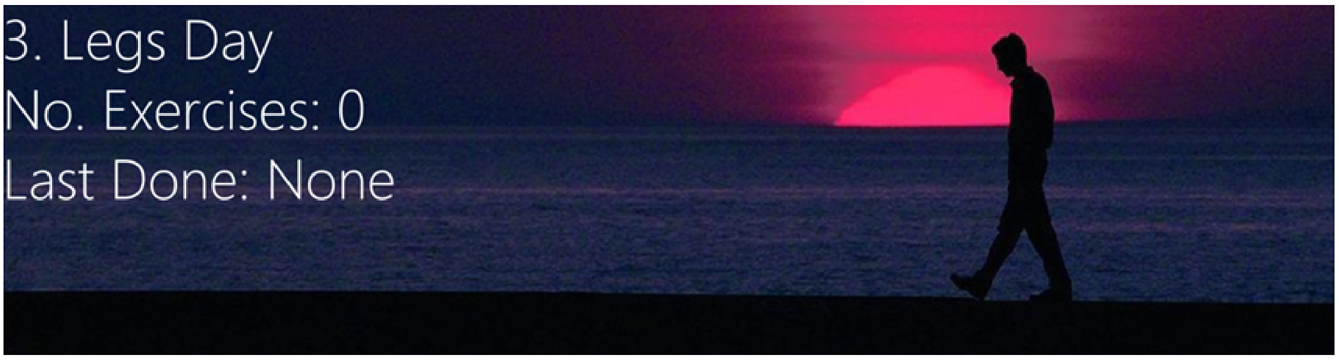


Figure 2. Low Intensity Card

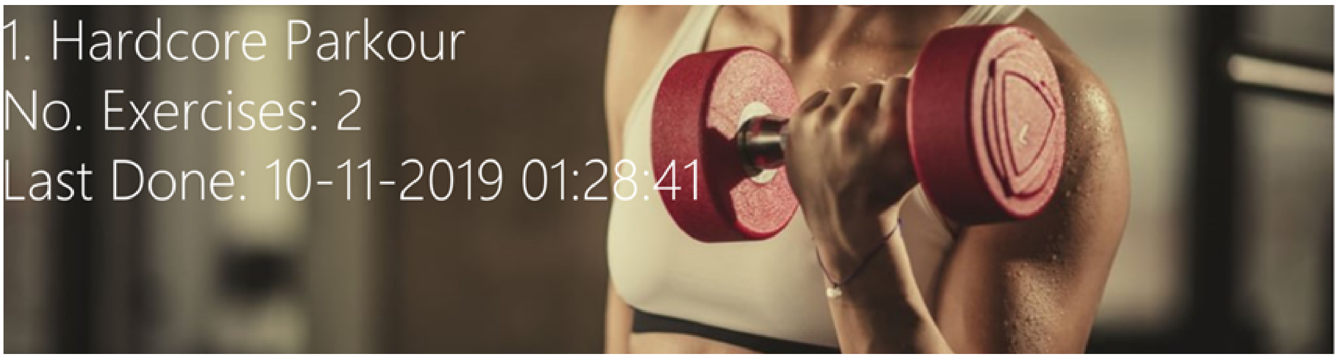


Figure 3. Medium Intensity Card

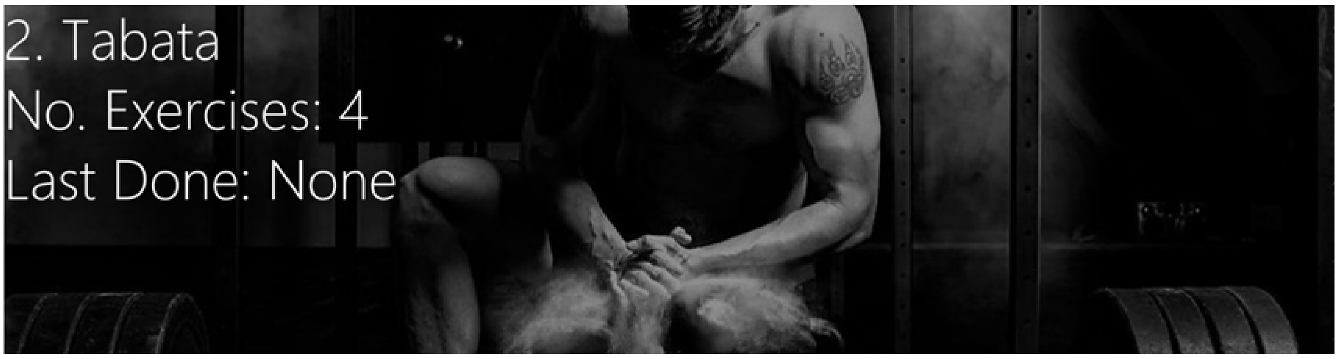


Figure 4. High Intensity Card

Similarly, to find more information on the workout, you can use the view command which is covered in a later section of this User Guide.

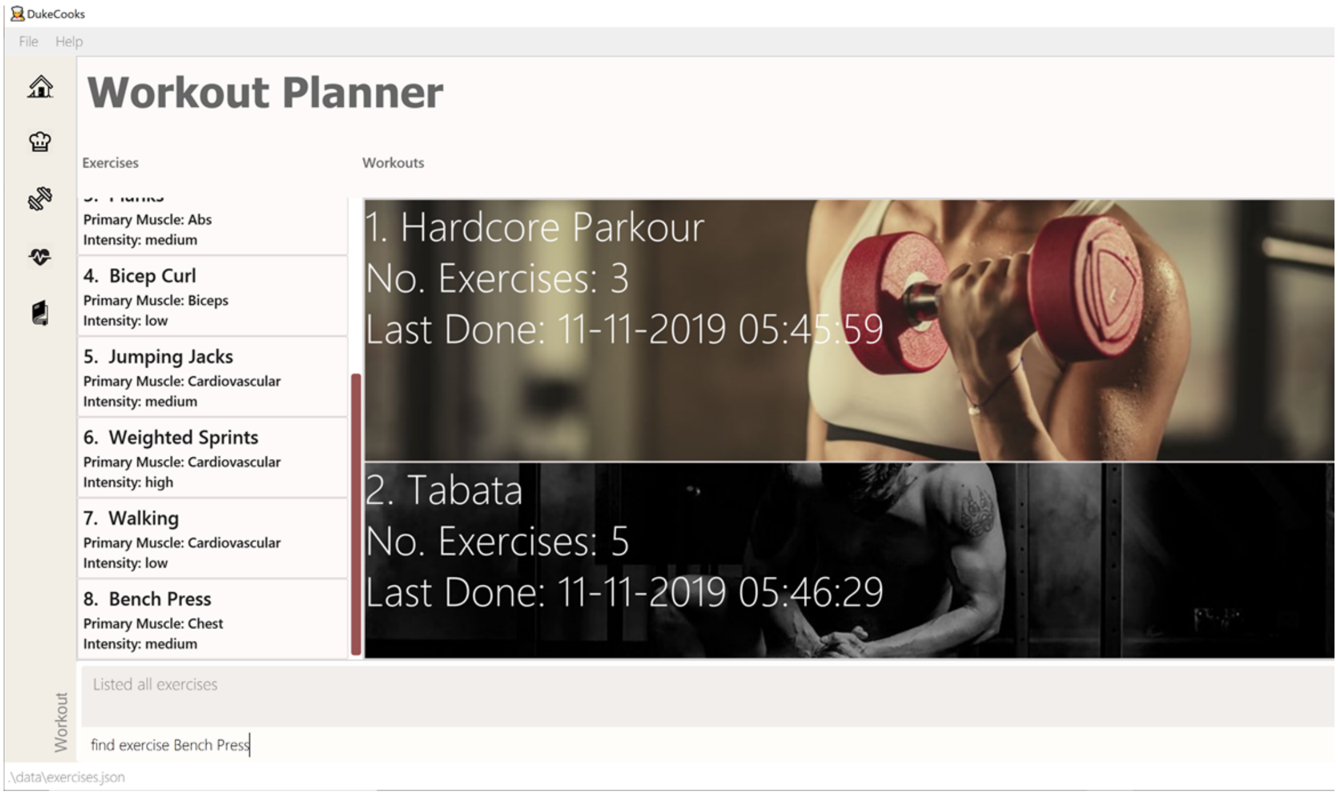
Find Exercise:

Now that you have a sizable amount of Exercises, you realise that you cannot easily find them amongst the sea of cards. To tackle this problem of seamless navigation, we implemented the find command.

Find Exercise works in 3 ways: By name, muscles trained (inclusive of both primary and secondary muscles) and intensity

To use the following command, simply type `find` in the command box followed by the variant you wish to utilise and the predicate.

- To find by name, the variant word is `exercise`
- To find by muscles trained, the variant word is `exerciseMuscle`
- To find by intensity, the variant word is `exerciseIntensity`



3. Planks
Primary Muscle: Abs
Intensity: medium

4. Bicep Curl
Primary Muscle: Biceps
Intensity: low

5. Jumping Jacks
Primary Muscle: Cardiovascular
Intensity: medium

6. Weighted Sprints
Primary Muscle: Cardiovascular
Intensity: high

7. Walking
Primary Muscle: Cardiovascular
Intensity: low

8. Bench Press
Primary Muscle: Chest
Intensity: medium

1. Hardcore Parkour
No. Exercises: 3
Last Done: 11-11-2019 05:45:59

2. Tabata
No. Exercises: 5
Last Done: 11-11-2019 05:46:29

Listed all exercises

find exercise Bench Press

.\data\exercises.json

Figure 5. Find by Exercise Name

The filtered lists will then be shown.

To show every exercise again, simply invoke the `list exercise` command.

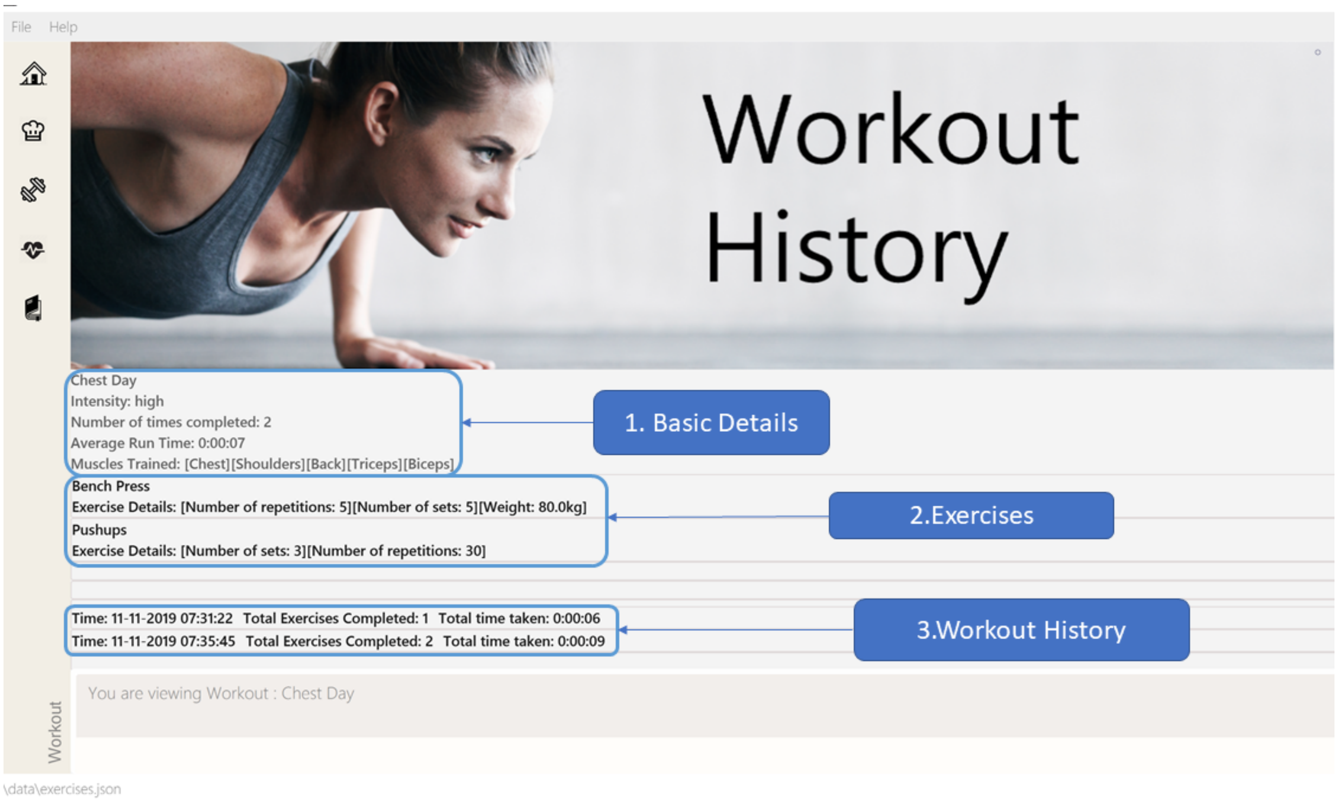
View Workout: `view workout`

You've created your workout regime, but you're unsure of the details and your progress. With the `view workout` feature, you can check all the details of your workout as well as its history!

Similar to `view exercise`, type in `view workout` followed by the index of the workout you're interested in.

The following screen will now appear. To orientate you, here are the components of the screen:

1. Basic Details
   This component shows the Name, Intensity, Number of times completed, Average Run Time and Muscles Trained by the workout.
2. Exercises
   The exercises segment show a list of all exercises contained in the workout as well as its exercise details
3. Workout History
   The workout history compiles a list of all the times you have ran and completed this workout



Chest Day
Intensity: high
Number of times completed: 2
Average Run Time: 0:00:07
Muscles Trained: [Chest][Shoulders][Back][Triceps][Biceps]

Bench Press
Exercise Details: [Number of repetitions: 5][Number of sets: 5][Weight: 80.0kg]
Pushups
Exercise Details: [Number of sets: 3][Number of repetitions: 30]

```
Time: 11-11-2019 07:31:22  Total Exercises Completed: 1  Total time taken: 0:00:06
Time: 11-11-2019 07:35:45  Total Exercises Completed: 2  Total time taken: 0:00:09
```

```
You are viewing Workout : Chest Day
```

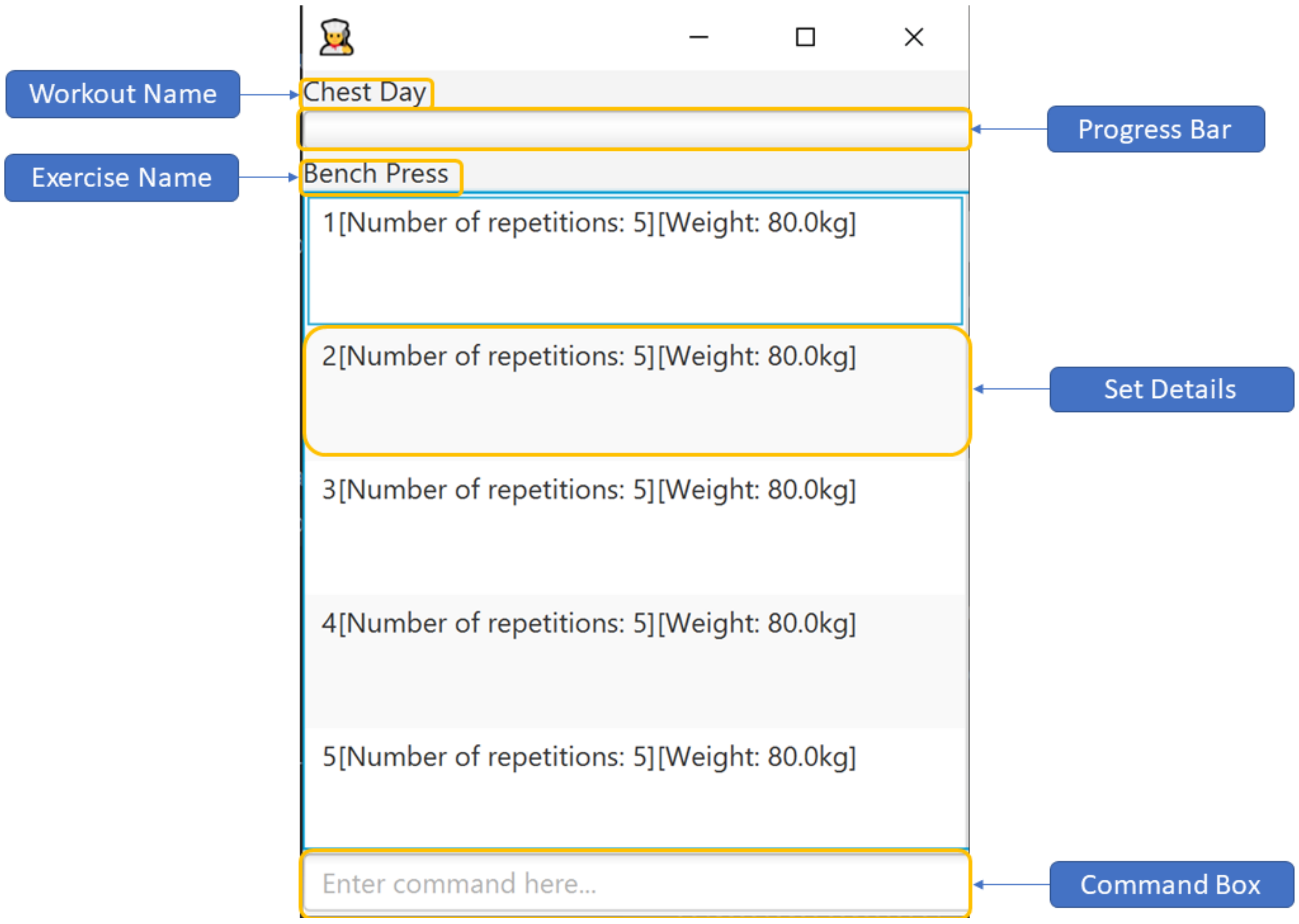
\data\exercises.json

Figure 6. Chest Day Screen

Similarly to `view Exercise` command, this page will be automatically updated after each run of the workout. You can also return to the workout planner page with `goto exercise`.

Run Workout: `run workout`

You've created all your exercises and added it into your ultimate workout. Now you're ready to run it! To run the workout, invoke the `run workout` command with the index of your ultimate workout.



Chest Day

Bench Press

1[Number of repetitions: 5][Weight: 80.0kg]

2[Number of repetitions: 5][Weight: 80.0kg]

3[Number of repetitions: 5][Weight: 80.0kg]

4[Number of repetitions: 5][Weight: 80.0kg]

5[Number of repetitions: 5][Weight: 80.0kg]

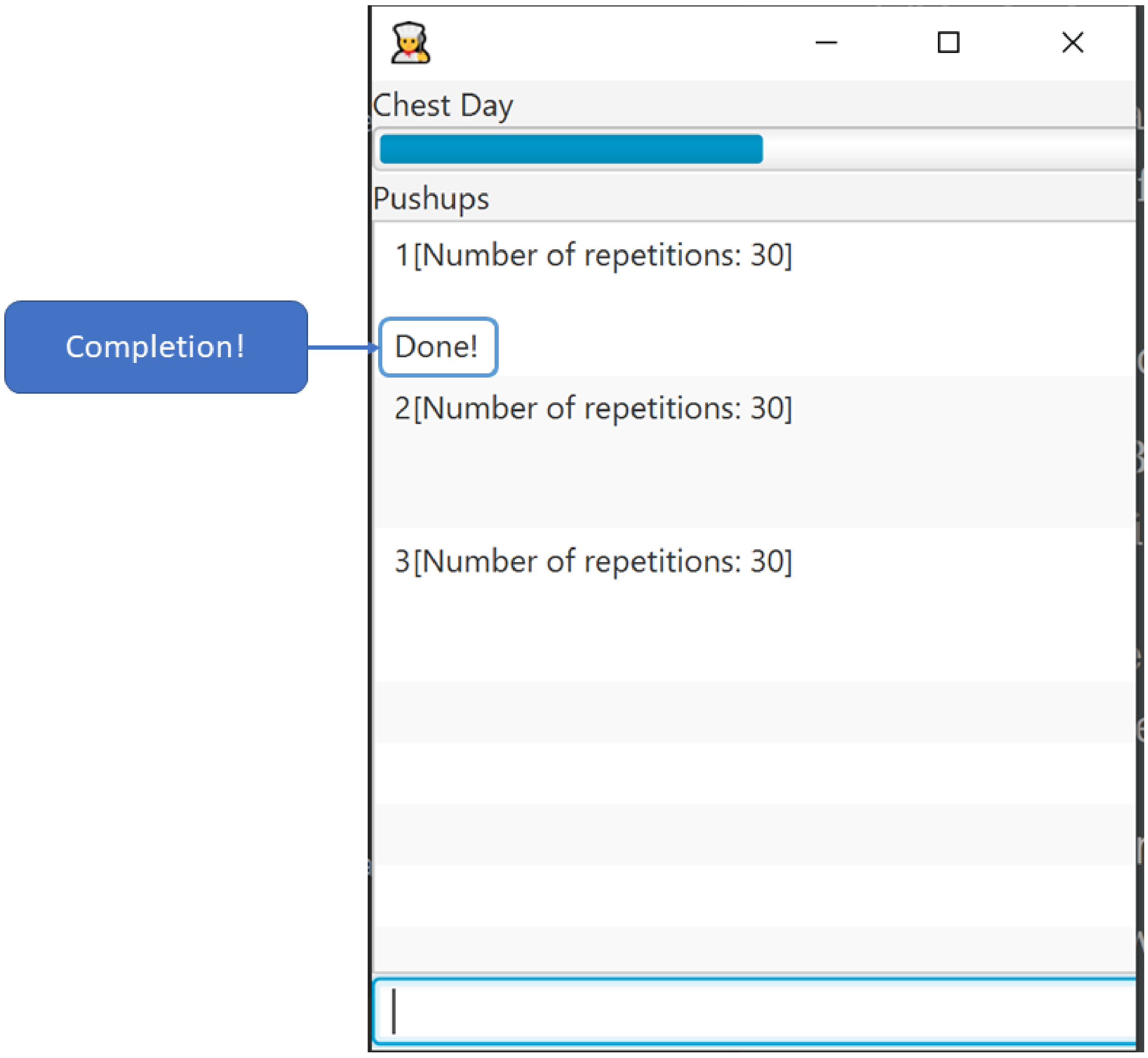
Enter command here...

Figure 7. Run Workout Window

The window above will pop up upon inputting the command. The window can be broken down into 5 major segments which are :

1. Workout Name
2. Progress Bar
3. Exercise Name
4. Set Details
5. Command Box

When you have completed a set, you can input `done` into the command box and it will indicate the set as completed as shown below.



Chest Day

Pushups

1[Number of repetitions: 30]

2[Number of repetitions: 30]

3[Number of repetitions: 30]

Figure 8. Marking a set as Done

After completing all the sets in all exercises, you will be brought back to the main page where a congratulatory message will greet you!

Now you know all there is to the workout planner! What are you waiting for? Go out there and put your gym membership to use!

DukeCooks Developer Guide

The following information provides my **contributions** to the **Developer Guide**. An explanation of how the feature (Dashboard), that I am in-charged of, is made.
To view the full **Developer Guide** of DukeCooks, please visit this link.

Workout Planner feature

The workout feature allows users to create their own custom workouts with `add Workout` command and adding their own custom exercises to it with `push exercise`. With these custom workouts, they can then choose to run them through `run workout` and monitor their progress and workout history with `view workout`.

Implementation

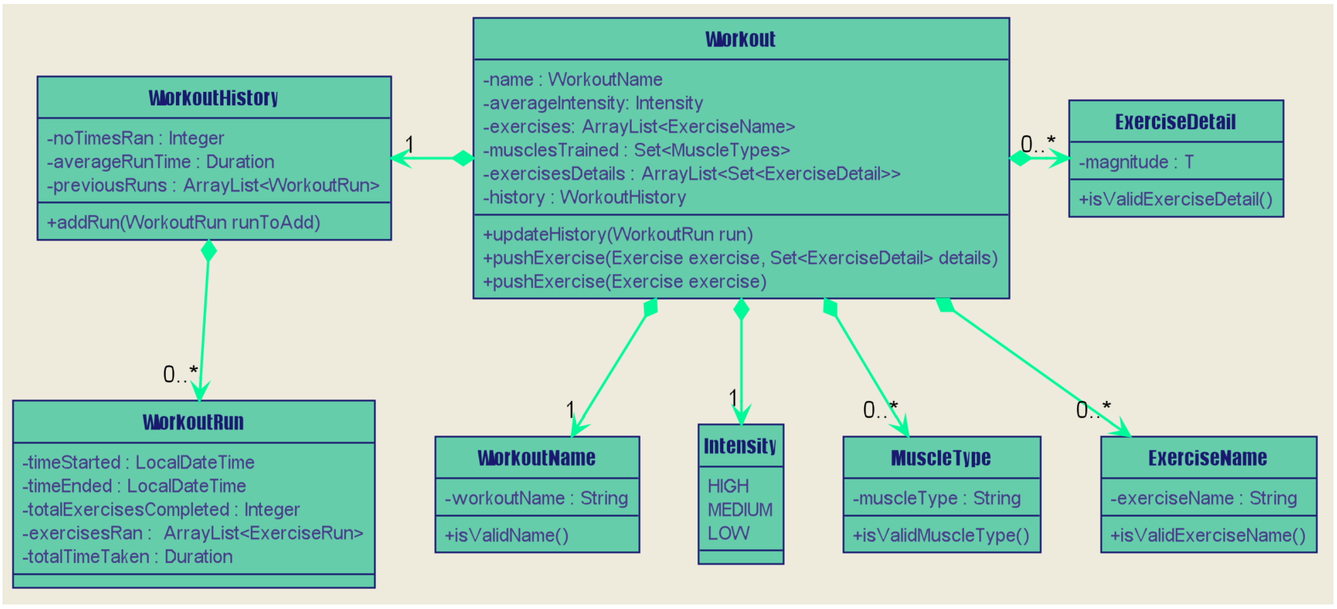
Workout Management

Every workout comprises of the following information:

- `WorkoutName` representing the name of the workout
- `Average Intensity` representing the average demands of the exercises in the workout
- A set of `MuscleType` which represents all the muscles trained by the workout
- An ArrayList of `ExerciseName` of exercises that would be carried out in the workout
- `WorkoutHistory` containing information on all the previous runs of the workout as well as some statistics

The Workout Class also contains the function `updateHistory(WorkoutRun run)` which adds the `WorkoutRun` into the WorkoutHistory and updates all the relevant fields accordingly, returning a new Workout instance with updated `WorkoutHistory`. The class also utilises `pushExercise(Exercise exercise, Set<ExerciseDetail> details)` function to add new `Exercise` and return a new Workout with update fields. There is also an option to push an exercise without the details with the overloaded method which instead opts to use the pre-built Set of `ExerciseDetails` in the `Exercise` itself.

The Workout Class is represented by the class diagram below.


Workout
-name : WorkoutName
-averageIntensity: Intensity
-exercises: ArrayList<ExerciseName>
-musclesTrained : Set<MuscleTypes>
-exercisesDetails : ArrayList<Set<ExerciseDetail>>
-history : WorkoutHistory
+updateHistory(WorkoutRun run)
+pushExercise(Exercise exercise, Set<ExerciseDetail> details)
+pushExercise(Exercise exercise)
WorkoutHistory
-noTimesRan : Integer
-averageRunTime : Duration
-previousRuns : ArrayList<WorkoutRun>
+addRun(WorkoutRun runToAdd)
ExerciseDetail
-magnitude : T
+isValidExerciseDetail()
WorkoutRun
-timeStarted : LocalDateTime
-timeEnded : LocalDateTime
-totalExercisesCompleted : Integer
-exercisesRan : ArrayList<ExerciseRun>
-totalTimeTaken : Duration
WorkoutName
-workoutName : String
+isValidName()
Intensity
HIGH
MEDIUM
LOW
MuscleType
-muscleType : String
+isValidMuscleType()
ExerciseName
-exerciseName : String
+isValidExerciseName()
1
0..*
0..*
1
1
0..*
0..*
Figure 9. Workout Class Diagram

The Workout Class is managed by the following commands:

- `AddWorkoutCommand` - Adds a new empty `Workout` into `WorkoutPlanner`
- `DeleteWorkoutCommand` - Deletes a `Workout` specified by an `Index` from `WorkoutPlanner`
- `PushExerciseCommand` - Pushes an `Exercise` specified by an `Index` into an existing `Workout`

Exercise Management

In order to run a `Workout`, users will have to add `Exercises` into the `Workout` as an empty workout cannot be ran. Users can use existing exercises or create their own custom exercises. Every exercise contains the following information:

- `ExerciseName` representing the name of the exercise
- `MusclesTrained` comprising of the primary `MuscleType` as well as an ArrayList of secondary `MuscleType` trained
- `Intensity` or how demanding the exercise is
- A set of `ExerciseDetails` which are optional additional information of the exercise such as `ExerciseWeight`, `Distance`, `Sets` and `Repetitions`
- `ExerciseHistory` containing information on all the previous `ExerciseRun` of the exercise

Like `Workout`, `Exercise` also has the method `updateHistory` which returns an updated Exercise with a new `ExerciseRun` accounted for.

The Exercise class is represented by the following class diagram below.

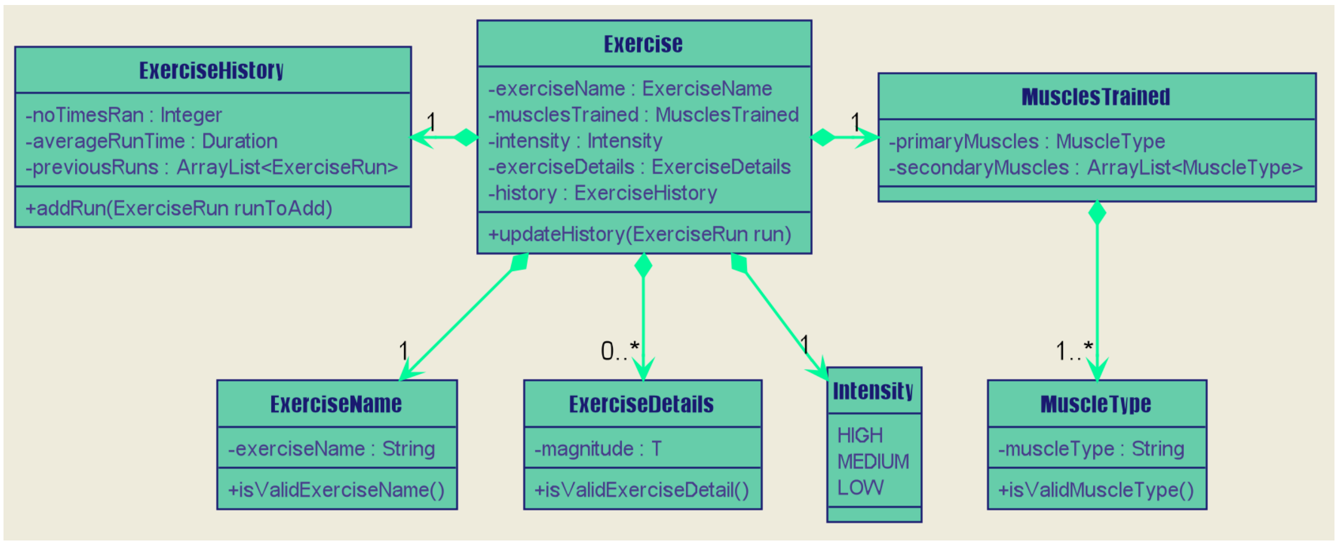

ExerciseHistory
-noTimesRan : Integer
-averageRunTime : Duration
-previousRuns : ArrayList<ExerciseRun>
+addRun(ExerciseRun runToAdd)
Exercise
-exerciseName : ExerciseName
-musclesTrained : MusclesTrained
-intensity : Intensity
-exerciseDetails : ExerciseDetails
-history : ExerciseHistory
+updateHistory(ExerciseRun run)
MusclesTrained
-primaryMuscles : MuscleType
-secondaryMuscles : ArrayList<MuscleType>
1
1
1
0..*
1
1..*
ExerciseName
-exerciseName : String
+isValidExerciseName()
ExerciseDetails
-magnitude : T
+isValidExerciseDetail()
Intensity
HIGH
MEDIUM
LOW
MuscleType
-muscleType : String
+isValidMuscleType()

Figure 10. Exercise Class Diagram

The `Exercise` class is managed by the following commands :

- `AddExerciseCommand` - Adds a new `Exercise` into `WorkoutPlanner`
- `DeleteExerciseCommand` - Deletes an `Exercise` specified by an `Index` from `WorkoutPlanner`
- `EditExerciseCommand` - Edits the specified `Exercise` with newly specified information
- `FindExerciseByIntensityCommand` - Lists all `Exercise` objects with the `Intensity` specified
- `FindExerciseByMuscleCommand` - Lists all `Exercise` objects which trains the `MuscleType` specified
- 'FindExerciseCommand' - Lists all `Exercise` objects with `ExerciseName` that contains the string specified
- 'ListExercise' - Lists all 'Exercise' objects in `WorkoutPlanner`

All the exercise and workout commands above are parsed in `DukeCooksParser`, invoking the respective Command Parsers (Add, Delete, Edit etc.). The `Exercise`/`Workout` variant of the parser will then be instantiated (i.e `AddExerciseCommandParser`,`DeleteWorkoutCommandParser` etc) to create the actual command objects (i.e AddExerciseCommand, DeleteWorkoutCommand etc). These Command Objects will then execute the necessary steps to fulfill their functionality.

Running of Workouts

The core functionality of the WorkoutPlanner is to run a `Workout` and have it automatically tracking your progress by making records in its history. This is done through the `Run Workout Command`. The following sequence diagrams show what happens when the command is invoked.

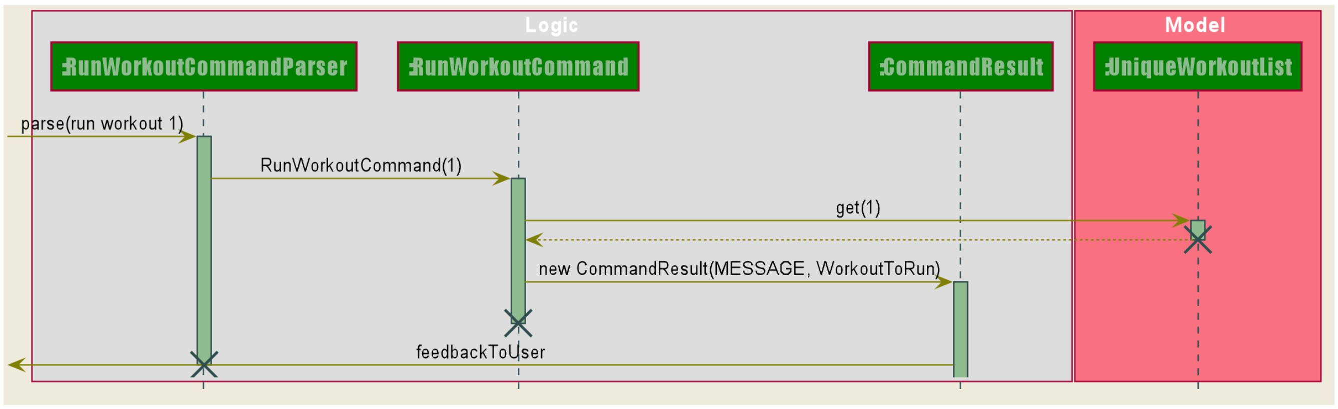


Figure 11. Sequence Diagram of RunWorkoutCommand

As seen in the diagram above, when the command is invoked, the `RunWorkoutParser` is initialised to parse the argument String to initialise `RunWorkoutCommand`. The Command object will then run its execute method, which calls upon get method of UniqueWorkoutList to obtain the target `Workout`. The target workout and message will then be passed back to the Ui through the CommandResult object. The Ui will then boot a new `RunWorkoutWindow` with the targeted workout.

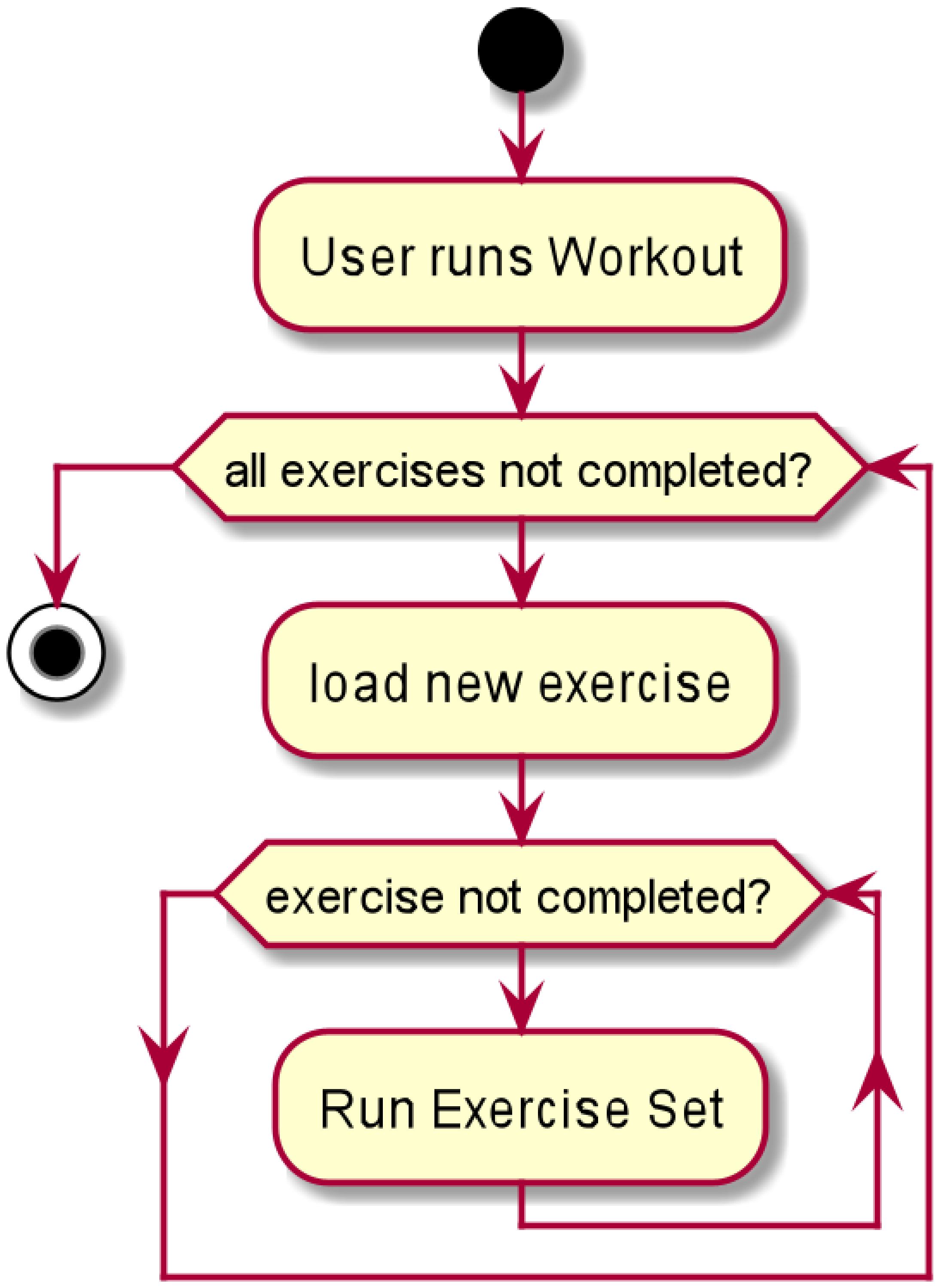


Figure 12. Activity Diagram of RunWorkoutWindow

The user will then run each set of each exercise until the workout is complete. The full loop is demonstrated in the activity diagram in Figure 17.

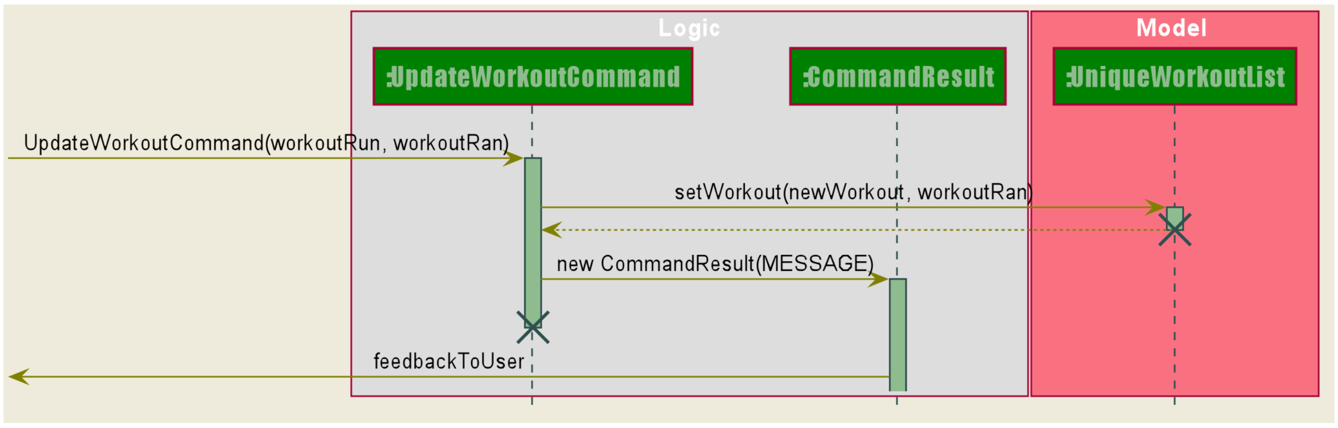

Logic
Model
:UpdateWorkoutCommand
:CommandResult
:UniqueWorkoutList
UpdateWorkoutCommand(workoutRun, workoutRan)
setWorkout(newWorkout, workoutRan)
new CommandResult(MESSAGE)
feedbackToUser

Figure 13. Sequence Diagram of UpdateWorkoutCommand

Upon completion of the workout, the Ui will immediately generate a new `UpdateWorkoutCommand` containing the `Workout` that has been ran and a newly instantiated `WorkoutRun` with the details of the run. `UpdateWorkoutCommand` will then be executed and the following will occur:

1. New Workout will be generated.
   Using Workout’s `updateHistory` method, a new updated `Workout` will be created.
2. The outdated Workout will be replaced by the new Workout.
   Using `UniqueWorkoutList` 's `setWorkout` method, the old workout will be removed and the updated one will be placed in its stead
3. CommandResult is generated and passed back to Ui.
   A new CommandResult will be returned containing a congratulatory message to the Ui signalling the successful completion of the workout.

Design Considerations

Aspect	Option 1 (Chosen)	Option 2
Storing an Exercise /Workout’s intensity	Intensity was stored as an Enumeration instead of a class - Pros: Intensity can be limited only a specific amount of values - Cons: Intensity will only be an estimate instead of a specific value given the value limits This option was chosen in the end to simplify the classification of exercise so that users can more easily filter by intensity. Furthermore, this allows for more Ui diversification by having different images for each intensity.	Setting Intensity as a Class - Pros: Easy to implement. - Cons: Makes filtering by intensity a more tedious affair for both developers and users.

Aspect	Option 1 (Chosen)	Option 2
Storing MuscleTypes	Have MuscleType be a class on its own - Pros: Muscles are referred to by various names and allowing the user to set their own muscle names allow for more familiarity - Cons: MuscleType class will require stricter validation to ensure that users do not mess up the programme with unintended inputs. This option was chosen to allow for greater flexibility of naming for the muscle types but at the same time still limited to prevent the users from going wild.	Store MuscleType as an enumeration - Pros: There are limited muscles in the body, allowing for a proper limit - Cons: Muscles may have multiple names that are not accounted for by the enum.
Storage of Exercises in Workout	Workouts only store a list of ExerciseName and not the full exercise - Pros: Exercises only have to be edited once upon execution of edit command - more cost effective. It also avoids unnecessarily large storage files. - Cons: Each time an exercise of workout has to be referenced, the entire storage of exercise has to be scoured In the end we decided to choose this option as we foresee that the edit command will be utilised more often than calling an exercise from a workout. Furthermore, to improve timing, we kept a sorted storage for exercise to allow for the quicker binary search.	Workouts store whole Exercises - Pros : Exercises can be extracted quickly - Cons : Huge storage space is required. Also complicates editing of exercises.