

# 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 one 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 be 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:

	<b>What I did</b>
Project management	Managed releases <a href="#">v1.2.1</a> (1 release) on GitHub Created Labels to be used for issues Instantiated the Milestones in issues
Reviewed PRs	PRs reviewed (with non-trivial review comments): <a href="#">#44</a> , <a href="#">#55</a>
Reported bugs	Reported bugs and suggestions for other teams in the class ( <a href="#">1</a> , <a href="#">2</a> , <a href="#">3</a> , <a href="#">4</a> , <a href="#">5</a> , <a href="#">6</a> , <a href="#">7</a> , <a href="#">8</a> , <a href="#">9</a> , <a href="#">10</a> , <a href="#">11</a> )

## Getting Started with DukeCooks

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!

But wait! I have no idea how to begin with this, you may be thinking. Fret not, read further to see a breakdown of the workout planner screen and what you can do with it!

### Workout Planner Screen

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

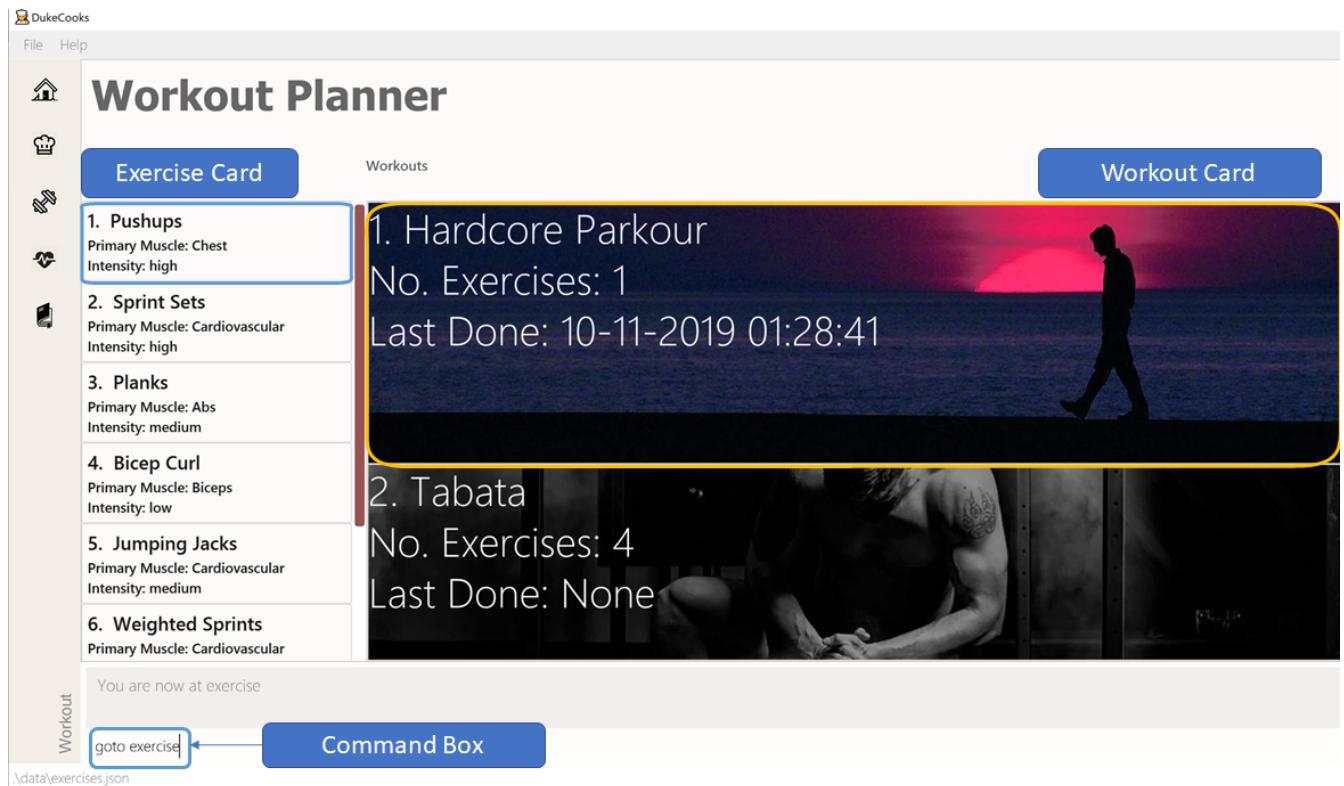


Figure 1. Workout Screen

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.

Similarly on the right we have the list of Workout Cards. The Workout Cards shows the Workout name, the number of Exercises in the workout as well as the last time the workout was ran. You can also tell the intensity of the workout from the image of the card.

3. Legs Day  
No. Exercises: 0  
Last Done: None

Figure 2. Low Intensity Card

## 1. Hardcore Parkour

No. Exercises: 2

Last Done: 10-11-2019 01:28:41



Figure 3. Medium Intensity Card

## 2. Tabata

No. Exercises: 4

Last Done: None



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.

### Adding an exercise: `add exercise`

"I now know the components of the Workout Planner! So what can I do with it?"

Well I'm glad you asked kind person! To start off, we have the add exercise command which creates new exercises for your workouts. To add an exercise, you must first specify its name, primary muscle trained as well as its intensity. You can also add optional fields such as secondary muscles and exercise details like sets and reps. The following shows all the possible fields that can be added:

- \* Name
- \* Primary Muscle
- \* Intensity
- \* Secondary Muscles (optional)
- \* Number of sets (optional)
- \* Number of repetitions (optional)
- \* Timing (optional)
- \* Distance (optional)
- \* Exercise Weight (optional)

To add the exercise, simply input the fields in the Command Box in the following format:

Format: `add exercise n/EXERCISE_NAME p/PRIMARY_MUSCLE sm/SECONDARY_MUSCLE i/INTENSITY s/SETS r/REPETITIONS d/ DISTANCE w/WEIGHT t/TIMING`

Examples:

The following images show an example of the command being inputted and its corresponding outputs.

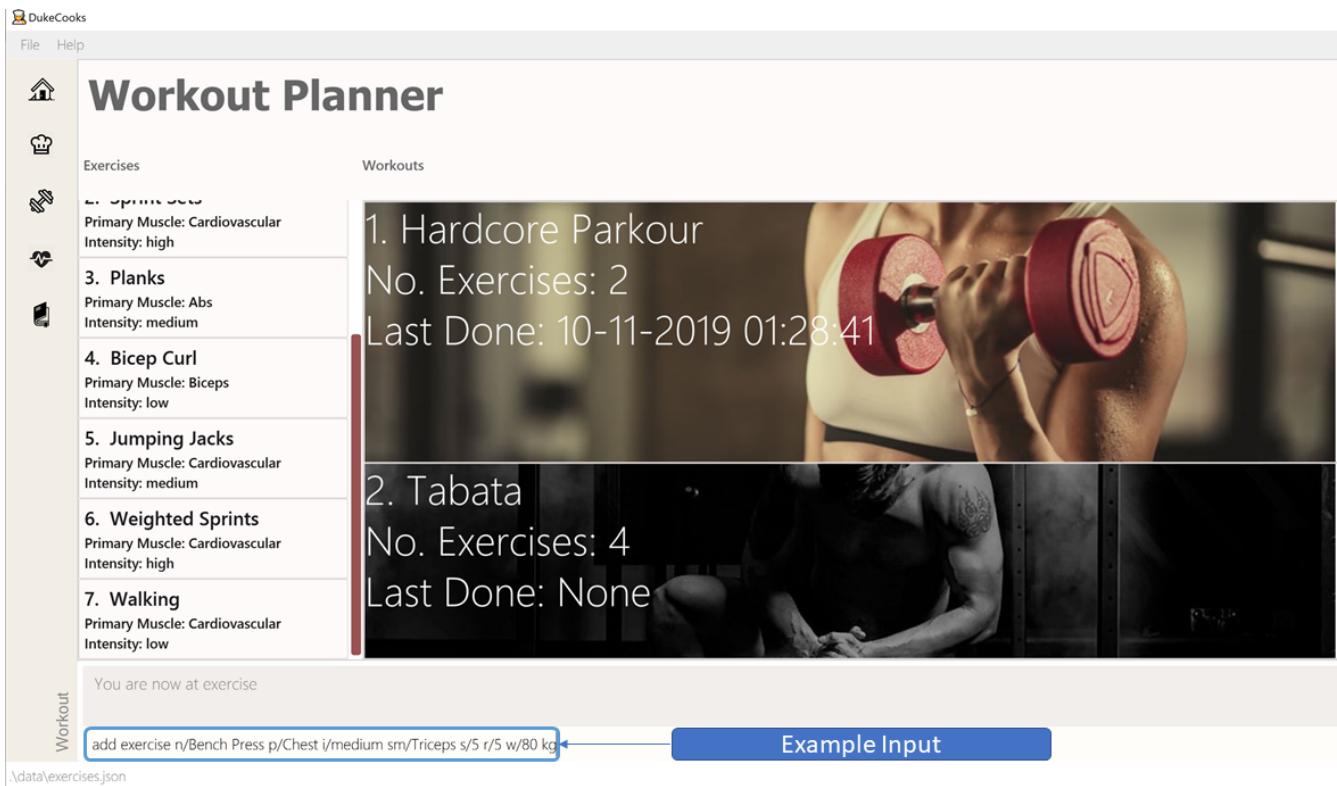


Figure 5. Example input

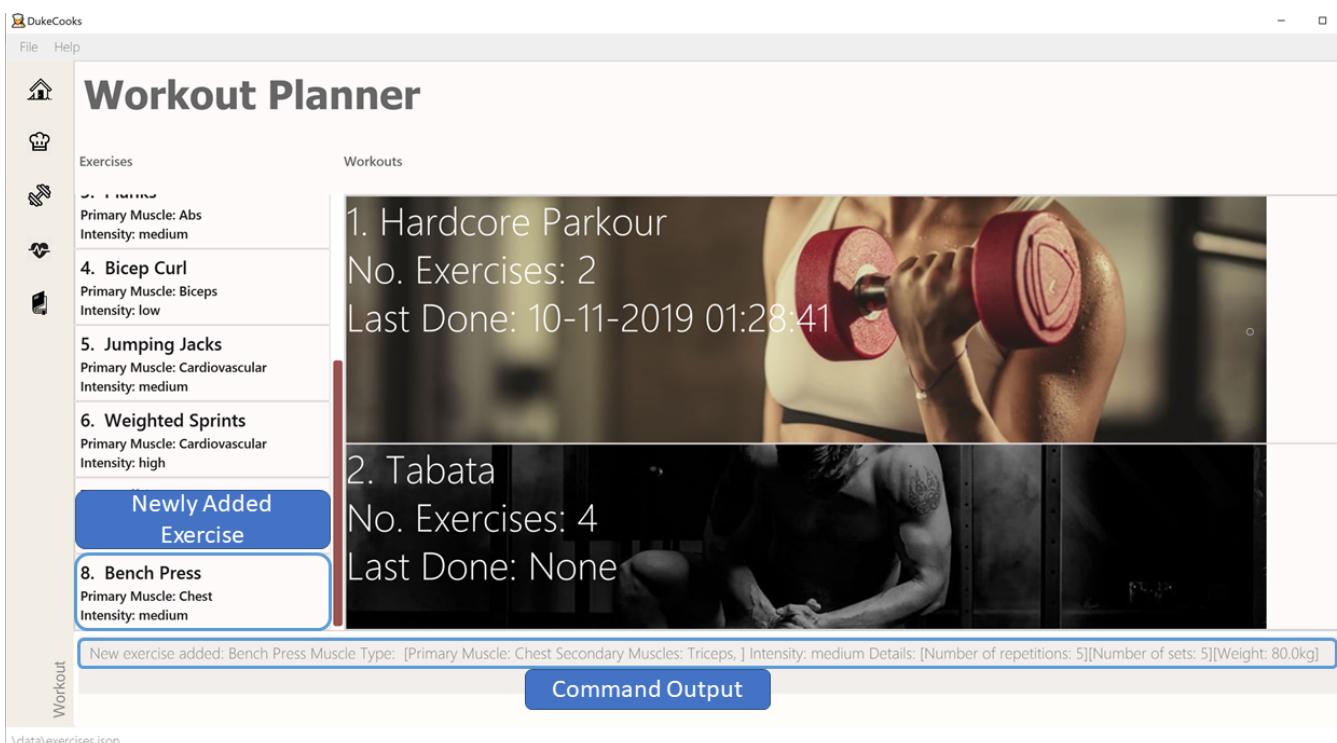


Figure 6. Example output

## View Exercise: `view exercise`

Ok, now that you've added your exercise, you probably want to know all of its details. Simply input the command `view exercise` followed by the index of the exercise you want to view as shown below.

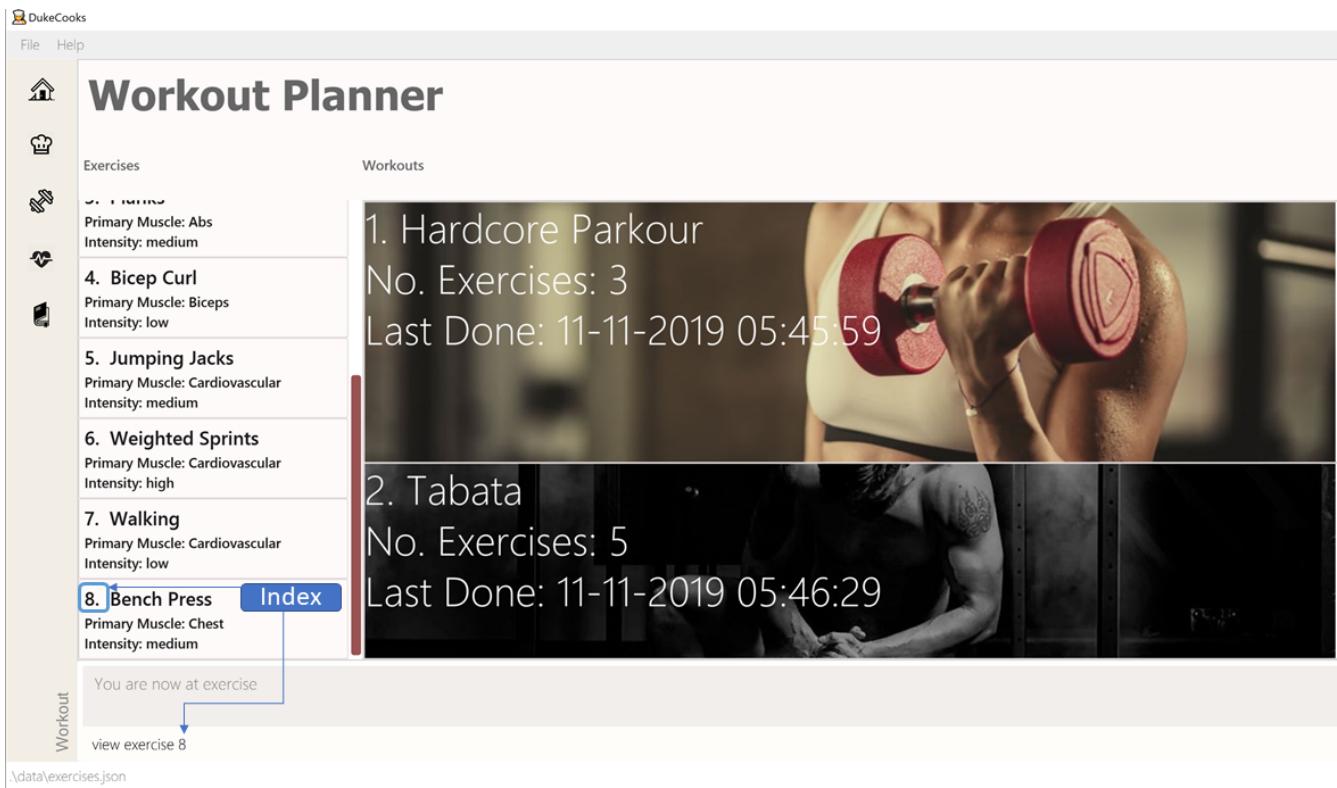


Figure 7. Example Input

The screen should now show the full details of the exercise in question. Let's get you up to speed with what's what.

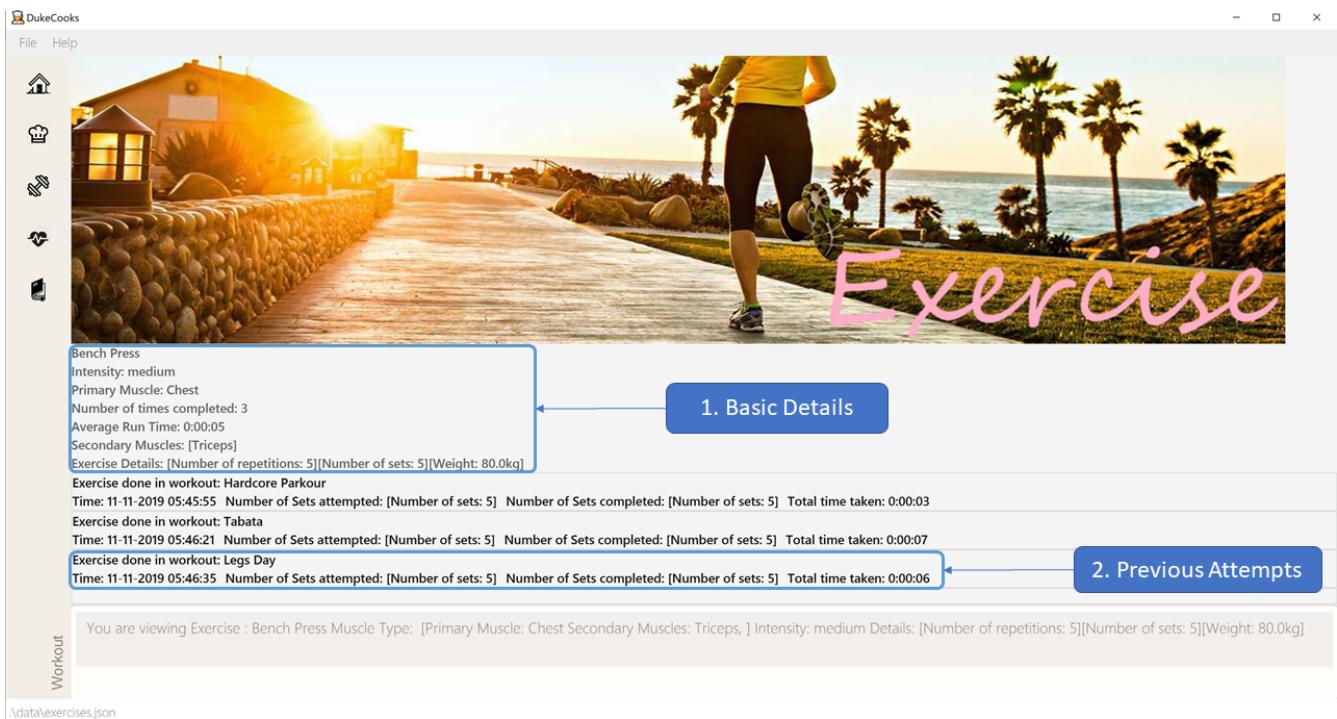


Figure 8. Exercise Details Screen

## 1. Basic Details

The first segment comprises of the basic details of the exercise. This includes the Exercise Name, Intensity, Primary and Secondary Muscles, Number of times the exercise has been executed, Average Run Time of the Exercise as well as some of its details.

## 2. Previous Attempts

Next up we have a history of all the previous attempts of the exercise. This includes the workout that the exercise is done in, the time it was done as well as the time it took. It also specifies the number of sets attempted and completed.

The details on this page is automatically updated with each workout run. To return back to the exercise page, simply invoke the `goto exercise` command.

More statistics as well as graph view will be implemented in version 2.0.

## 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
- By Muscles Trained (inclusive of both primary and secondary muscles)
- By 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`

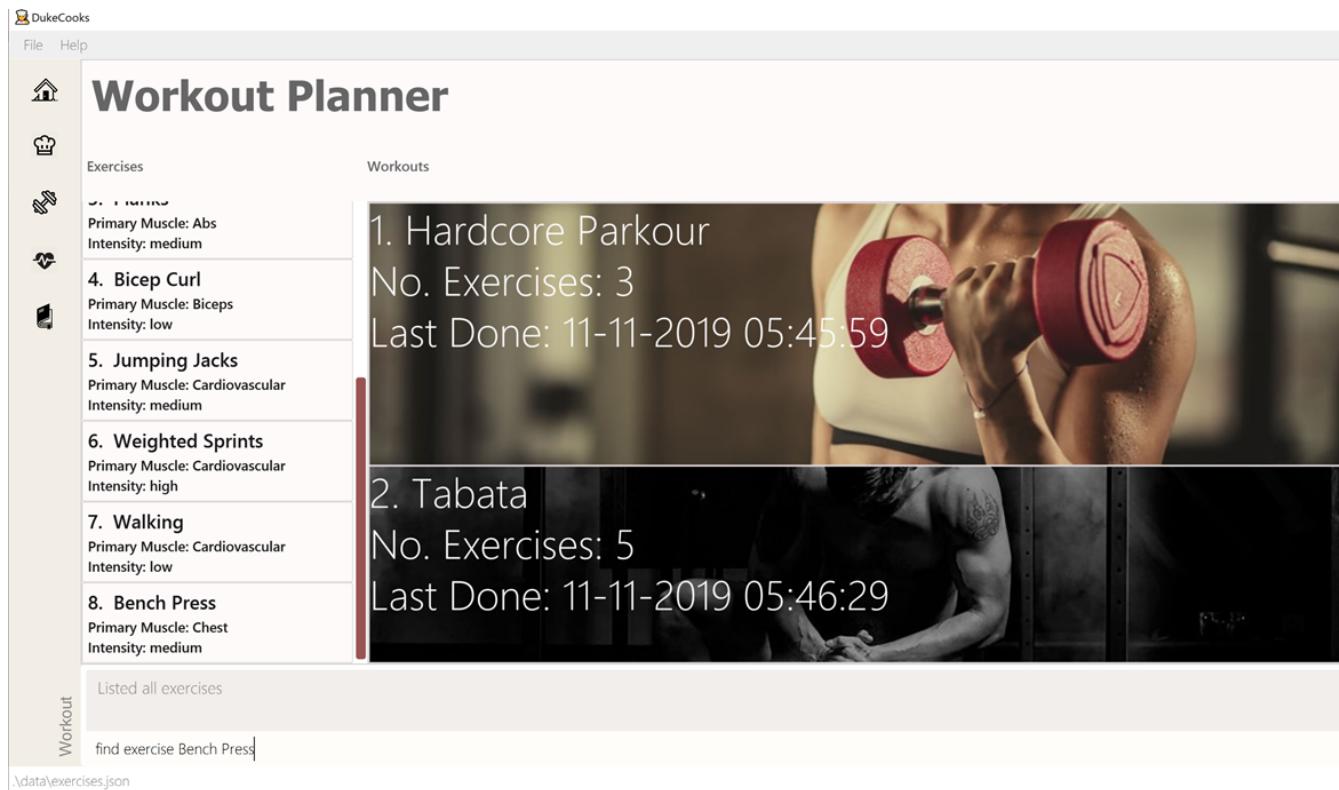


Figure 9. Find by Exercise Name

DukeCooks

File Help

# Workout Planner

**Exercises**

- 1. **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

**Workouts**

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 exerciseMuscle Chest

\data\exercises.json

Figure 10. Find by Muscles Trained

DukeCooks

File Help

# Workout Planner

**Exercises**

- 1. **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

**Workouts**

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 exercisIntensity medium

\data\exercises.json

Figure 11. Find by Intensity

The filtered lists will then be shown as follows:

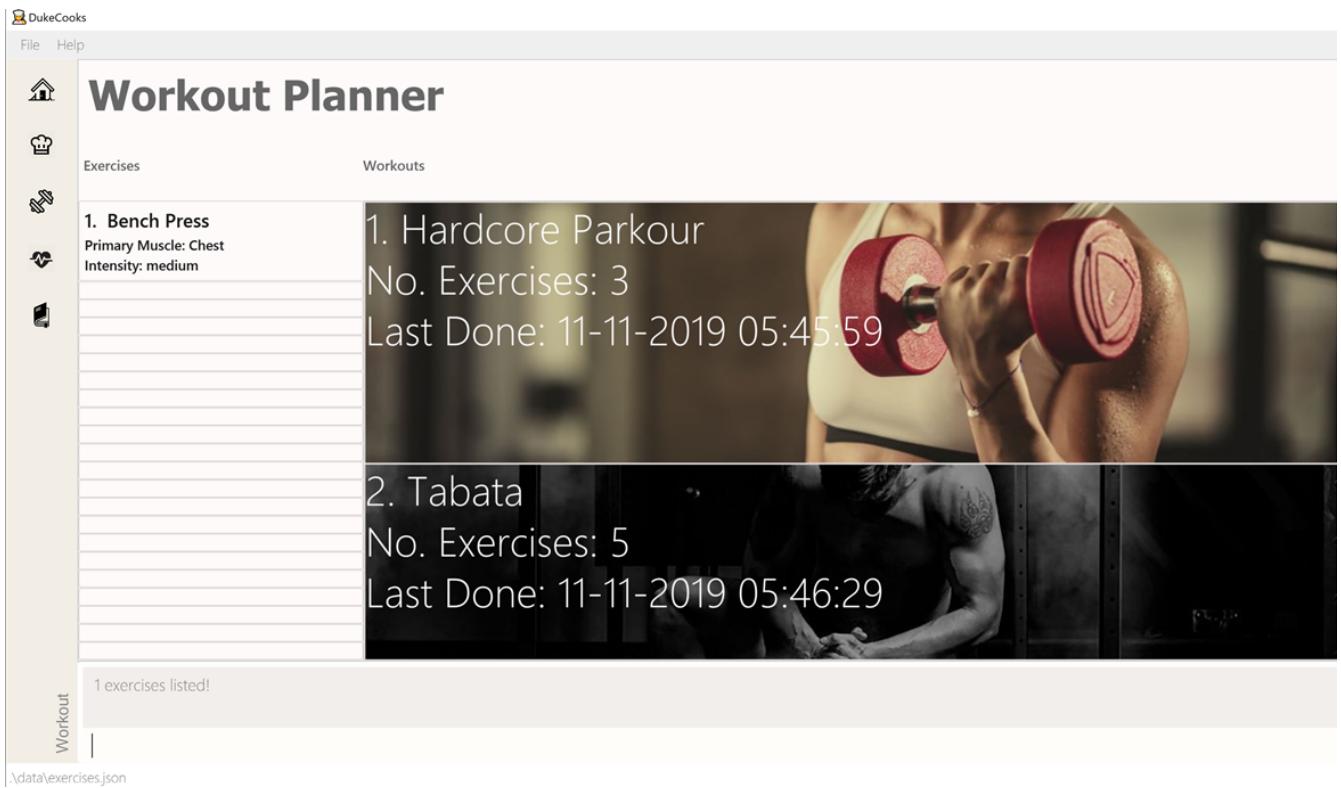


Figure 12. Find Output

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

## Add Workout: `add workout`

Now that you've created all your exercise, you're ready to create a workout plan! To create a workout, input `add workout n/NAME` into the command box, replacing `NAME` with the name of your workout. This will initialise your Workout with no exercises in it.

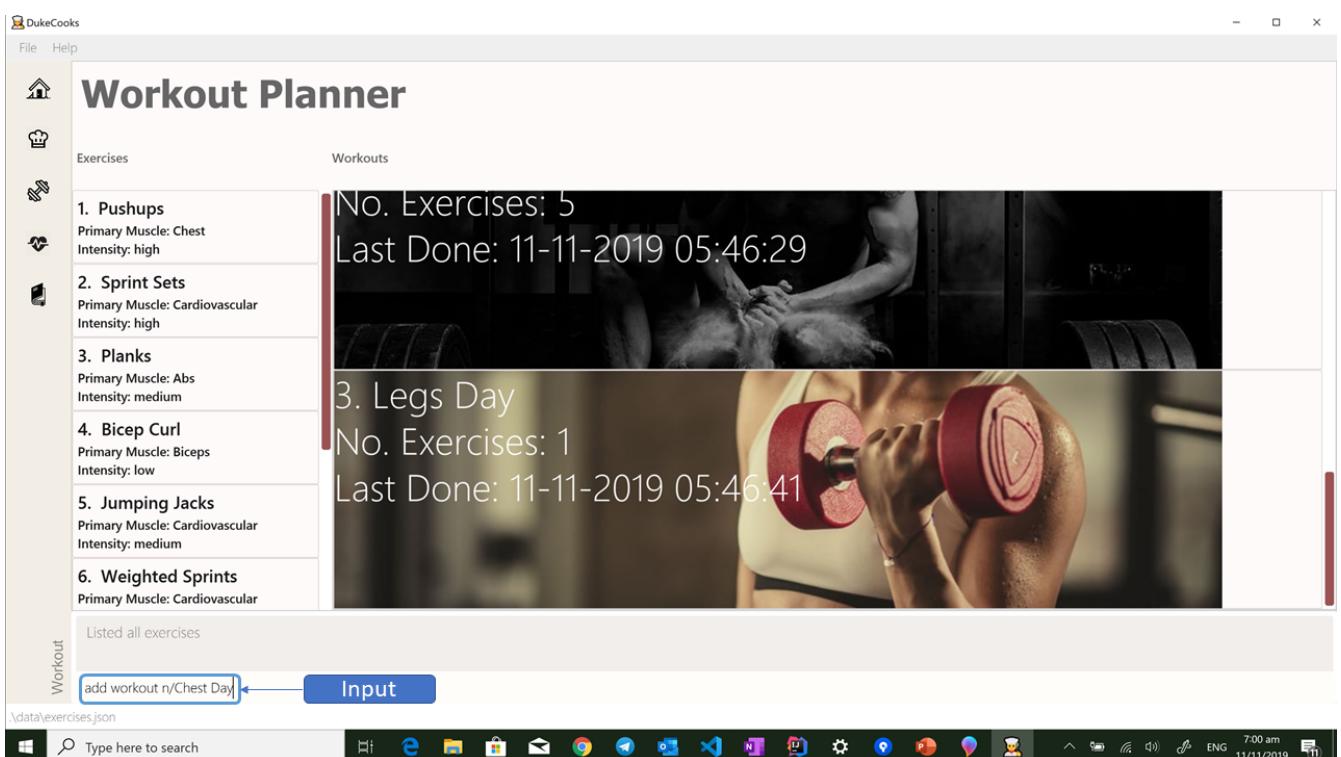


Figure 13. Add Workout Input

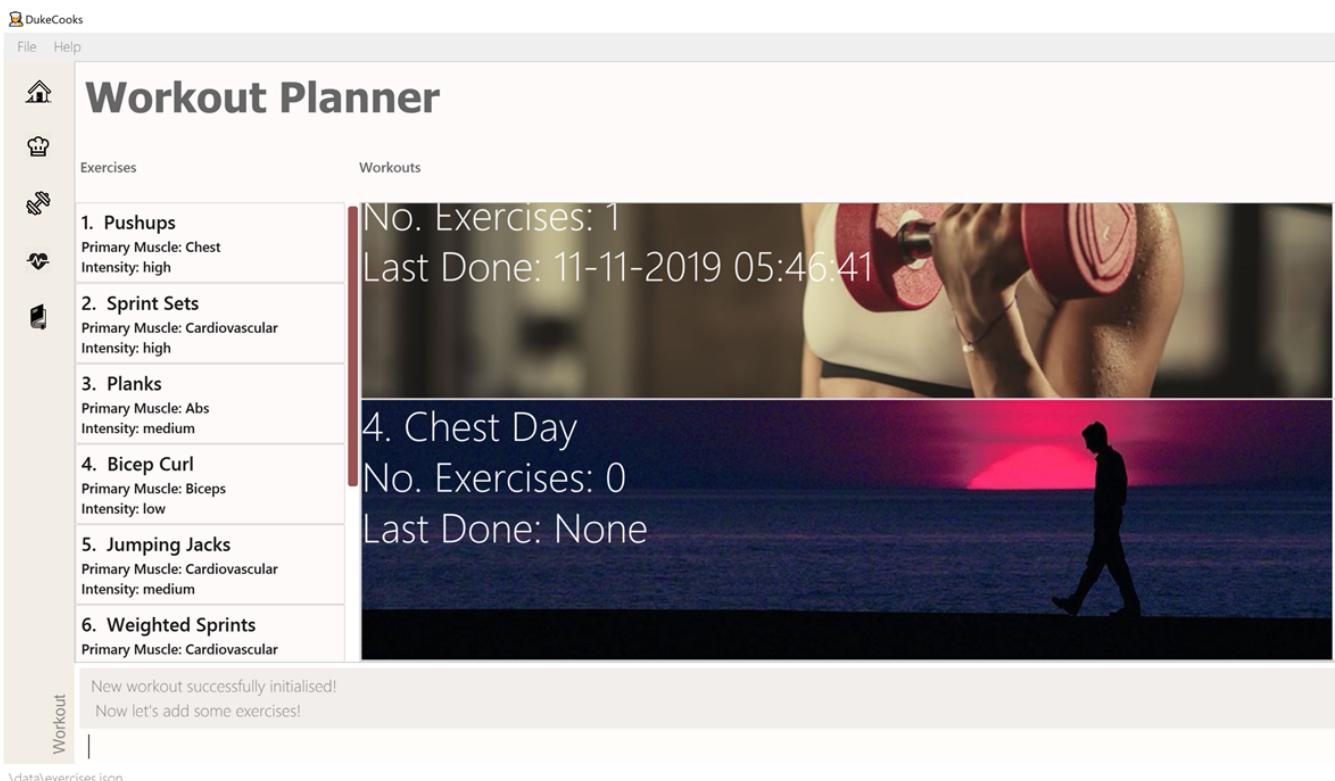


Figure 14. Add Workout Output

## Push Exercise push exercise

Once you have initialised your workout, you are now ready to add your exercise into your workouts! To do so simply locate the index of the exercise you want to push and the workout you want to push your exercise into. Then input the command in the following format `push exercise wi/WORKOUT_INDEX ei/EXERCISE_INDEX`.

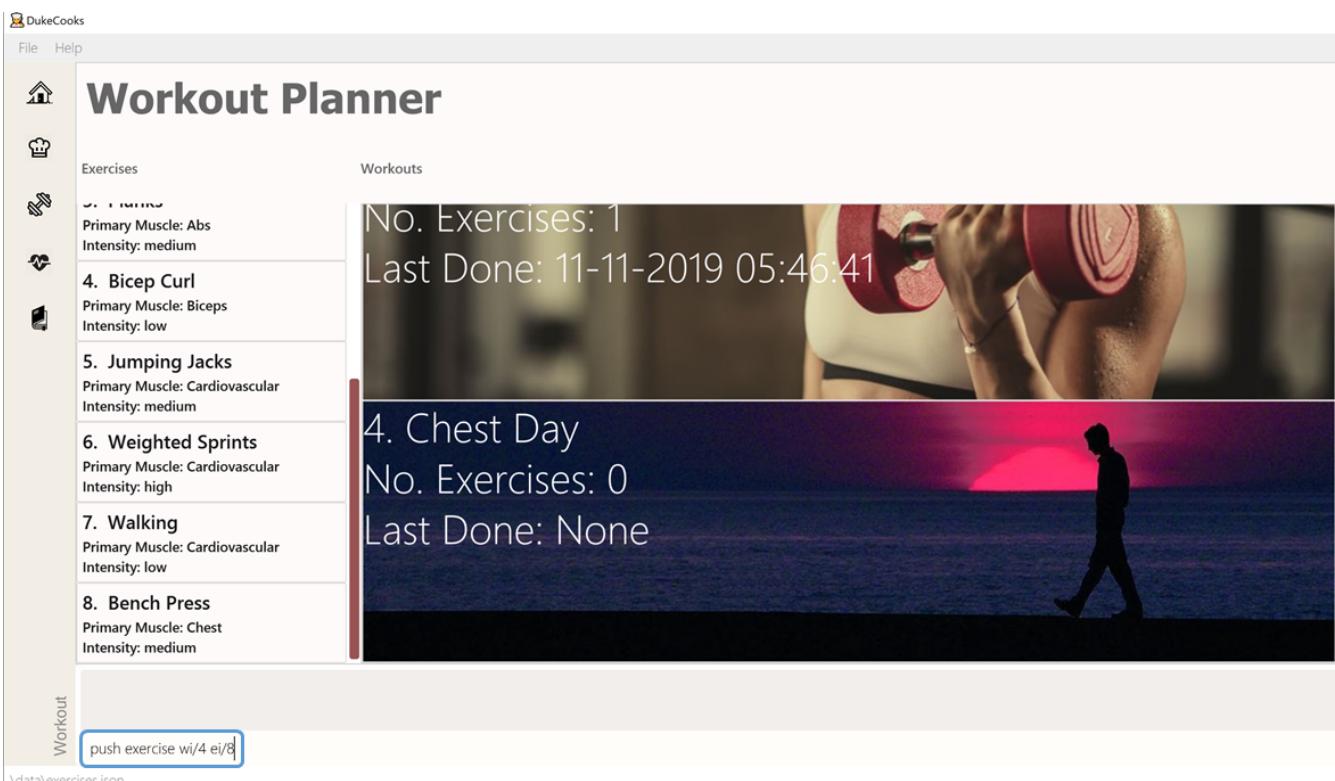


Figure 15. Pushing Bench Press into Chest Day

For example, if you want to add Bench Press exercise into Chest Day workout, type in **push exercise wi/4 ei/8** in the command box as seen above. You will get the following input.

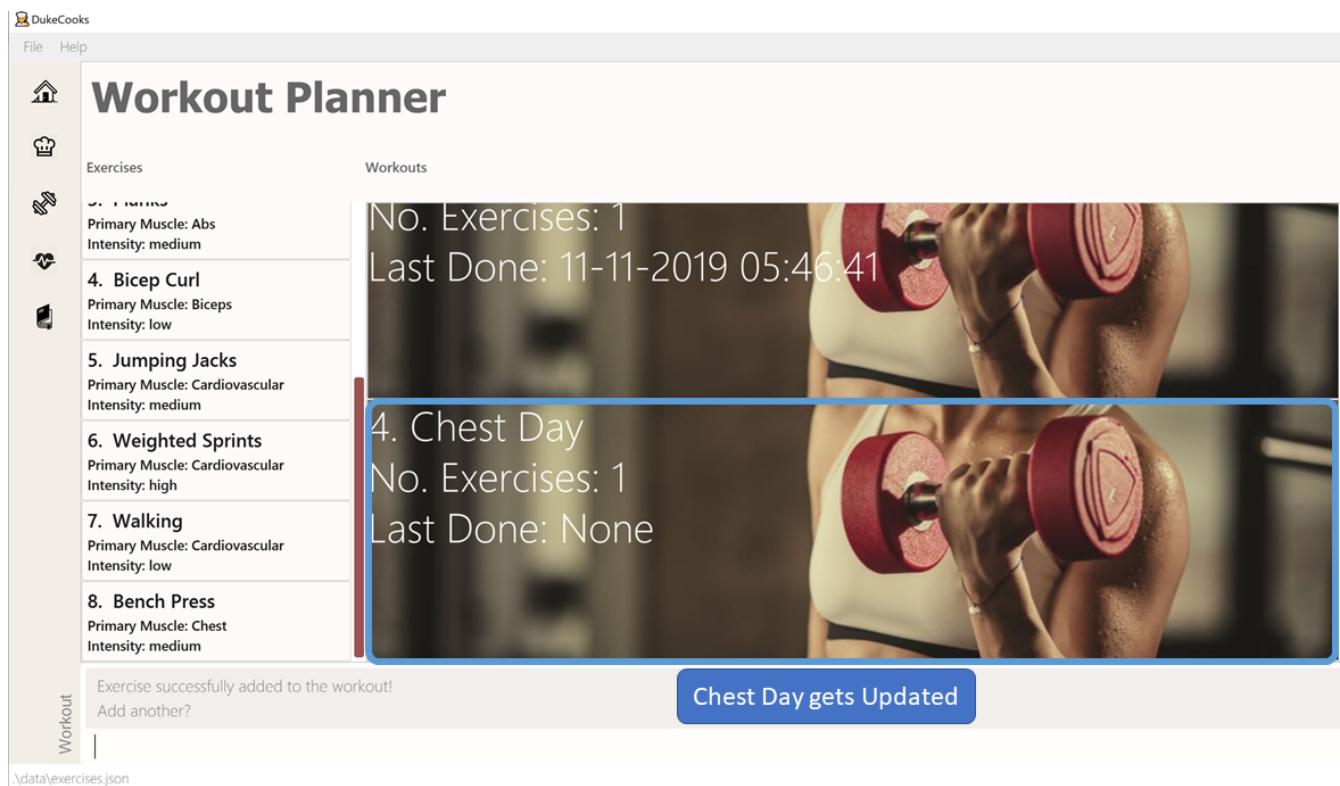


Figure 16. Push Exercise Results

Now that you know you know how to push exercises, you can go ahead and add in all the exercises needed for your ULITIMATE workout regime!

## 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. So if you want to view 'Chest Day' you can follow the image below.

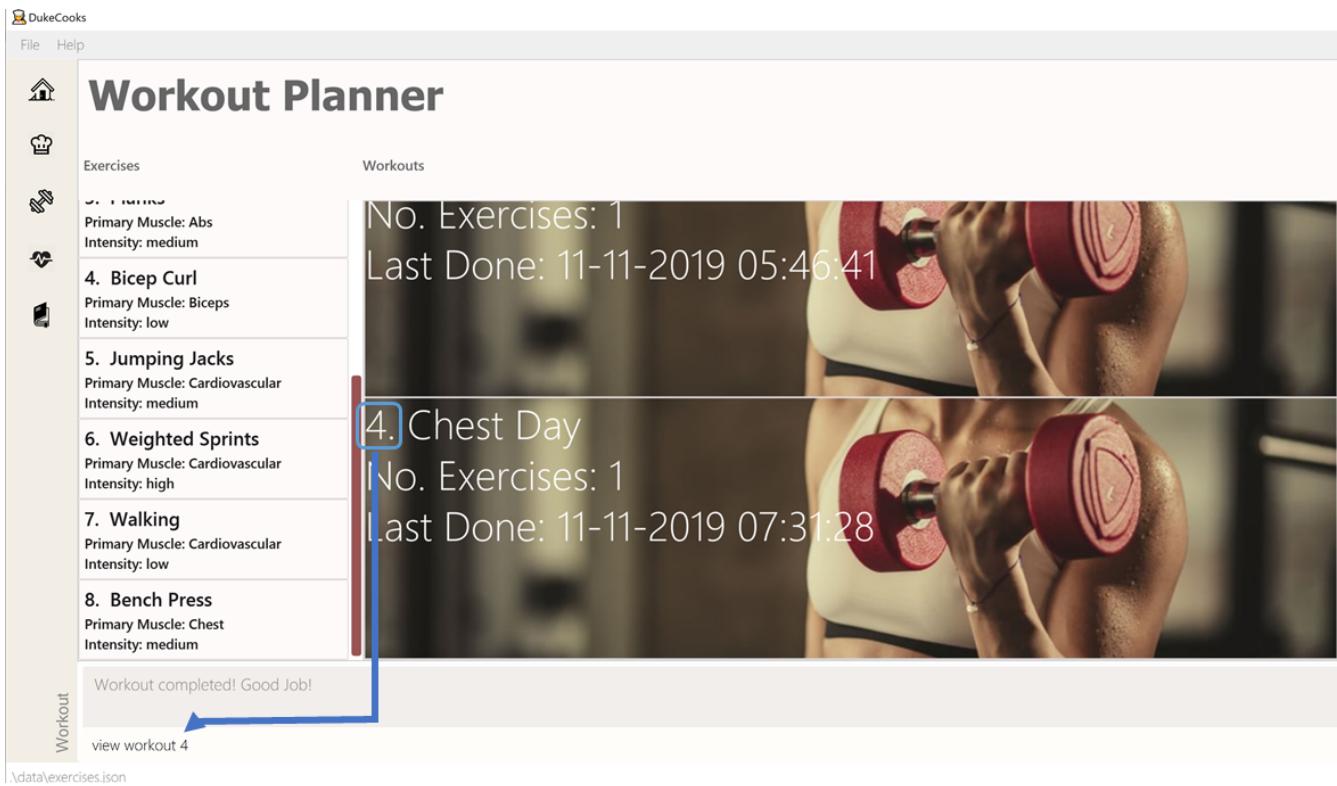


Figure 17. Viewing Chest Day Workout

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

The screenshot shows a workout history interface. At the top, there's a navigation bar with icons for File, Help, Home, Profile, Settings, and Log Out. Below the navigation is a large image of a woman in a grey tank top performing a plank exercise. To the right of the image, the title "Workout History" is displayed in a large, bold, black font.

Below the title, there are three sections:

- 1. Basic Details**: A blue box containing information about a "Chest Day" workout:
  - Intensity: high
  - Number of times completed: 2
  - Average Run Time: 0:00:07
  - Muscles Trained: [Chest][Shoulders][Back][Triceps][Biceps]
- 2. Exercises**: A blue box containing details for two exercises:
  - 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]
- 3. Workout History**: A blue box containing historical data:
  - 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

At the bottom left, there's a vertical sidebar with the word "Workout" and a file path "\data\exercises.json".

Figure 18. 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.

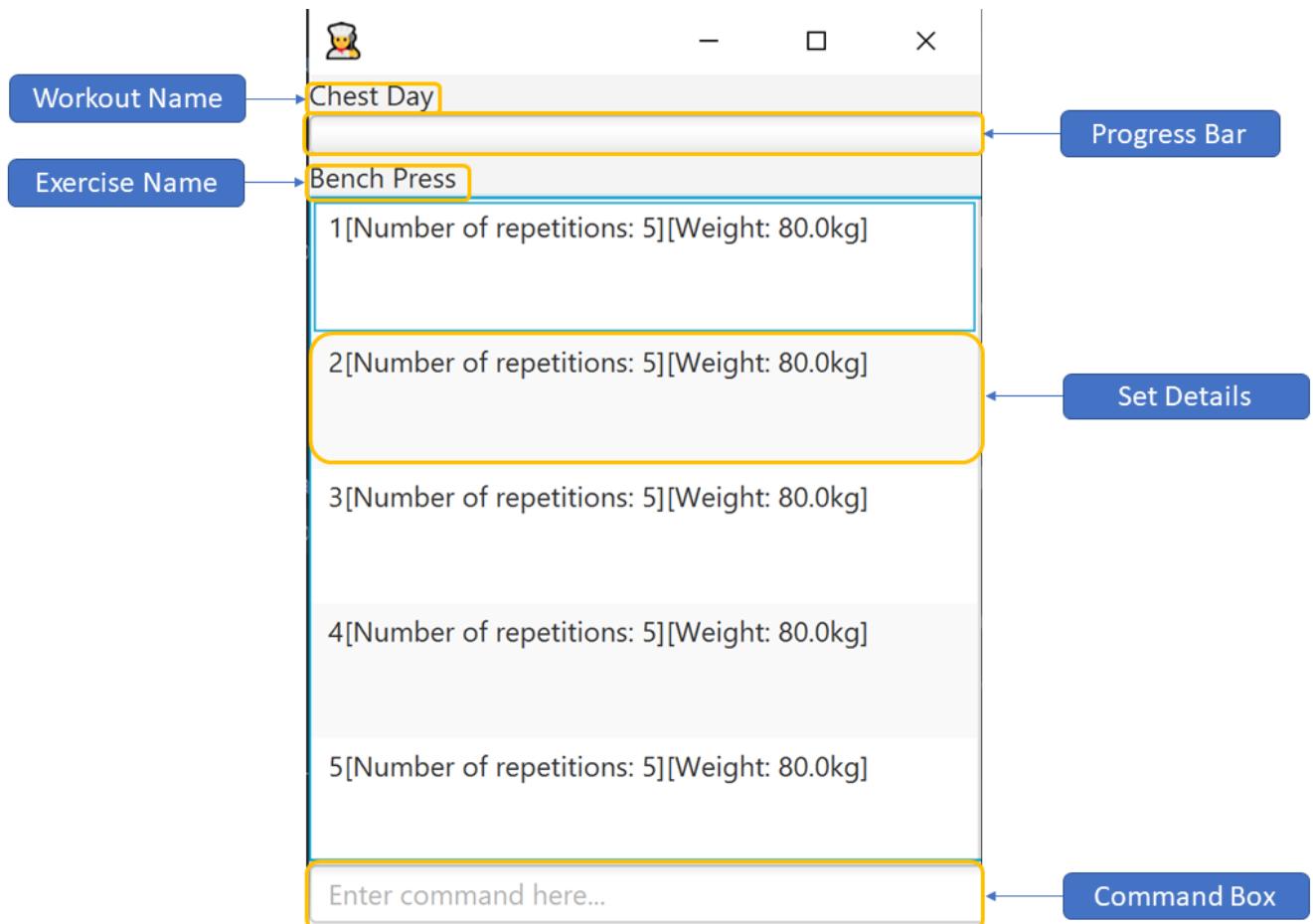


Figure 19. Run Workout Window

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

Name of the workout you are running.

2. Progress Bar

Shows your workout progress/ how far you are from completing the workout.

3. Exercise Name

The name of the current exercise you're doing

4. Set Details

The details of a set of the exercise you're currently doing.

5. Command Box

Where you input your commands

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.

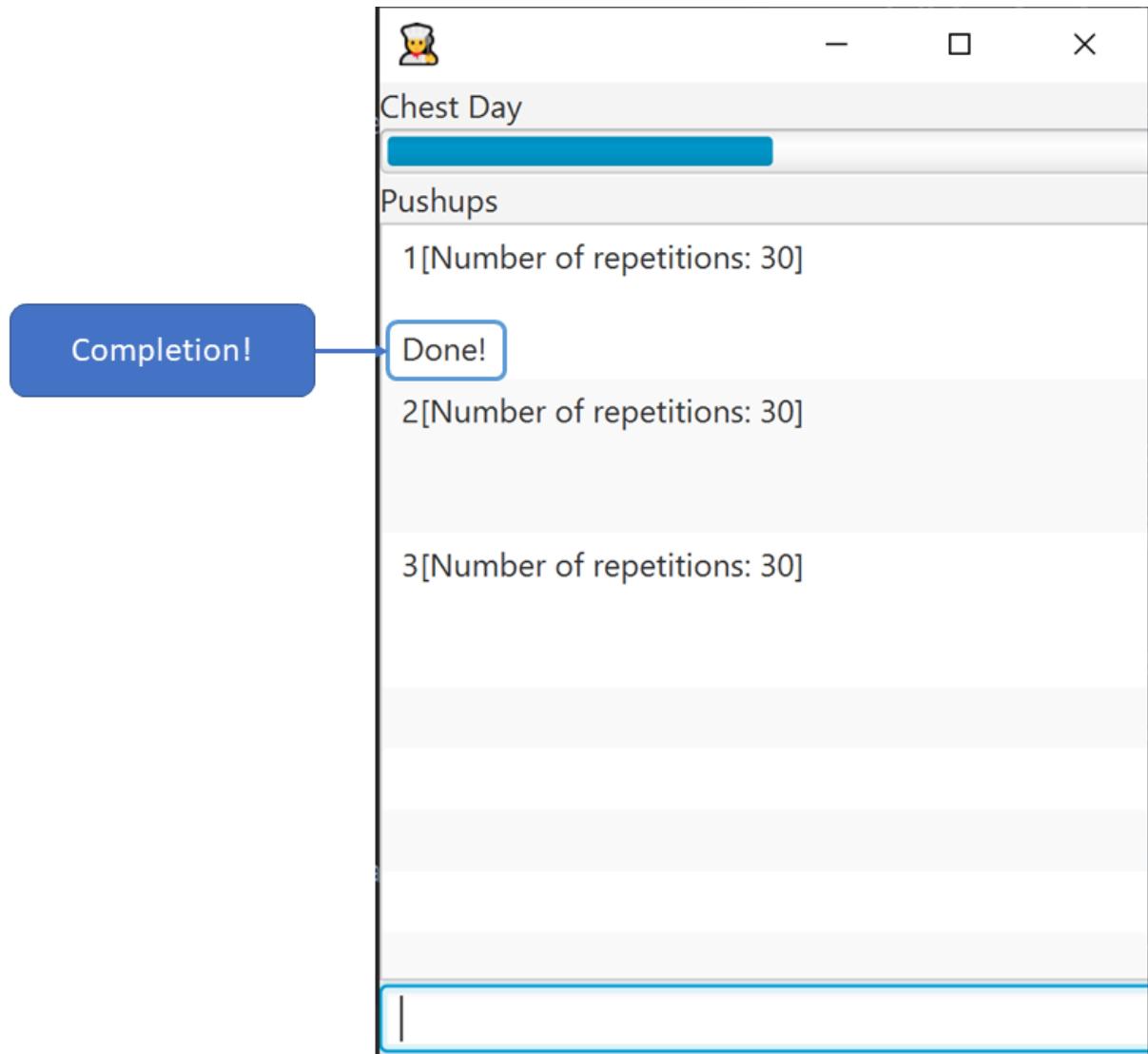
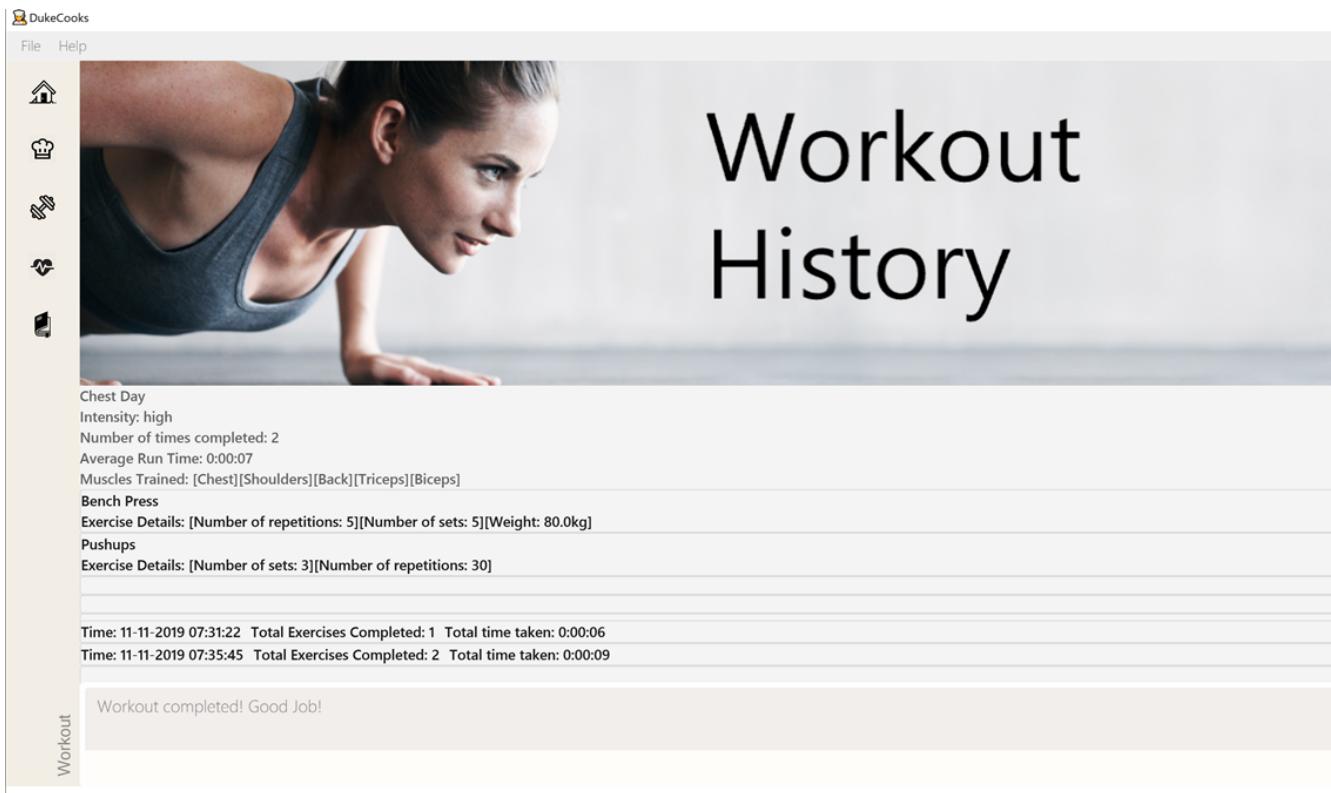


Figure 20. Marking a set as Done

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



.\data\exercises.json

*Figure 21. Congratulations!*

## Other Commands

With that, you have sufficient knowledge to create and run your workout... if you're perfect that is. To make your life easier, we've also included some commands to manage your exercises and workout in case you made errors!

### 1. List Exercise: `list exercise`

List exercises which matches optional parameters specified eg. muscle type/intensity.

Format: `list exercise m/MUSCLEGROUP

### 2. Clear Exercise `clear exercise`

Removes all exercises in the list.

Format: 'clear exercise'

### 3. Delete exercise: `delete exercise`

Deletes exercise of specified index.

Format: `delete exercise <index>`

### 4. Edit exercise: `edit exercise`

Edits exercise with new details

Format: `edit exercise n/EXERCISE_NAME p/PRIMARY_MUSCLE sm/SECONDARY_MUSCLE i/INTENSITY s/SETS r/REPETITIONS d/ DISTANCE w/WEIGHT t/TIMING`

### 5. Add calorie: `[coming in v2.0]` + Tracks calorie burned per rep/set of the exercise in kcal.

Format: `calorie <index> <calories>`

### 6. Delete workout: `delete workout` [coming in v1.4]

Deletes workout of specified index.

Format: `delete workout <index>`

### 7. Show graph: `[coming in v2.0]` Creates a graph showing all your past workouts and exercises.

8. Schedule Workout: [coming in v2.0] Create a workout to be added in to your schedule that will show up in your dashboard.

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!

# Making DukeCooks

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

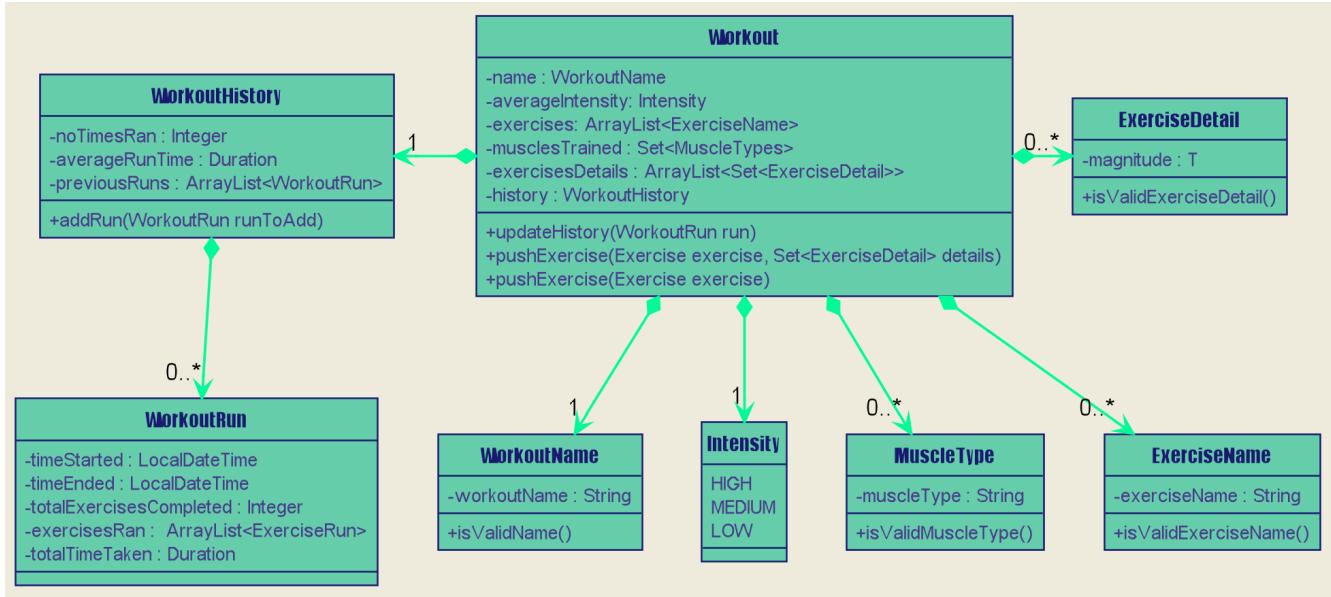
### Workout Management

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.



*Figure 22. 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.

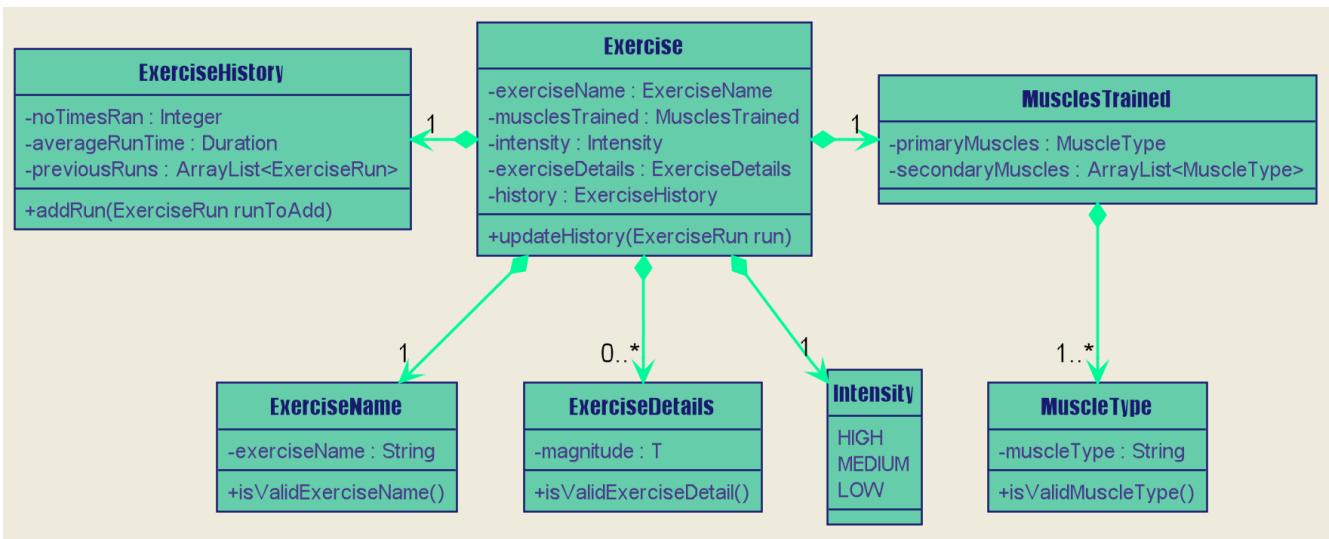


Figure 23. Exercise Class Diagram

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

- **AddExerciseCommand** - Adds a new **Exercise** into **WorkoutPlanner**
- **ClearExerciseCommand** - Clears all the **Exercise** objects in **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.

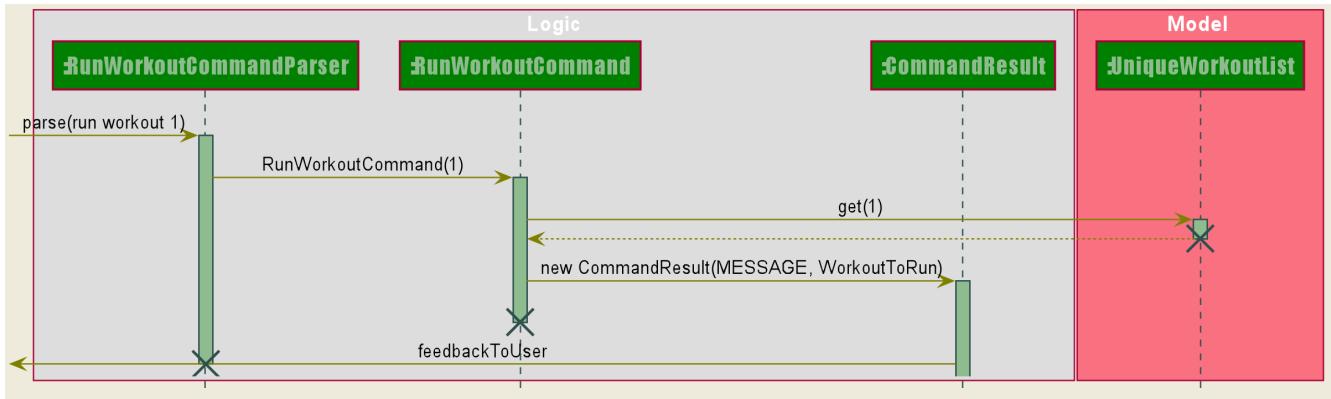


Figure 24. Sequence Diagram of `RunWorkoutCommand`

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.

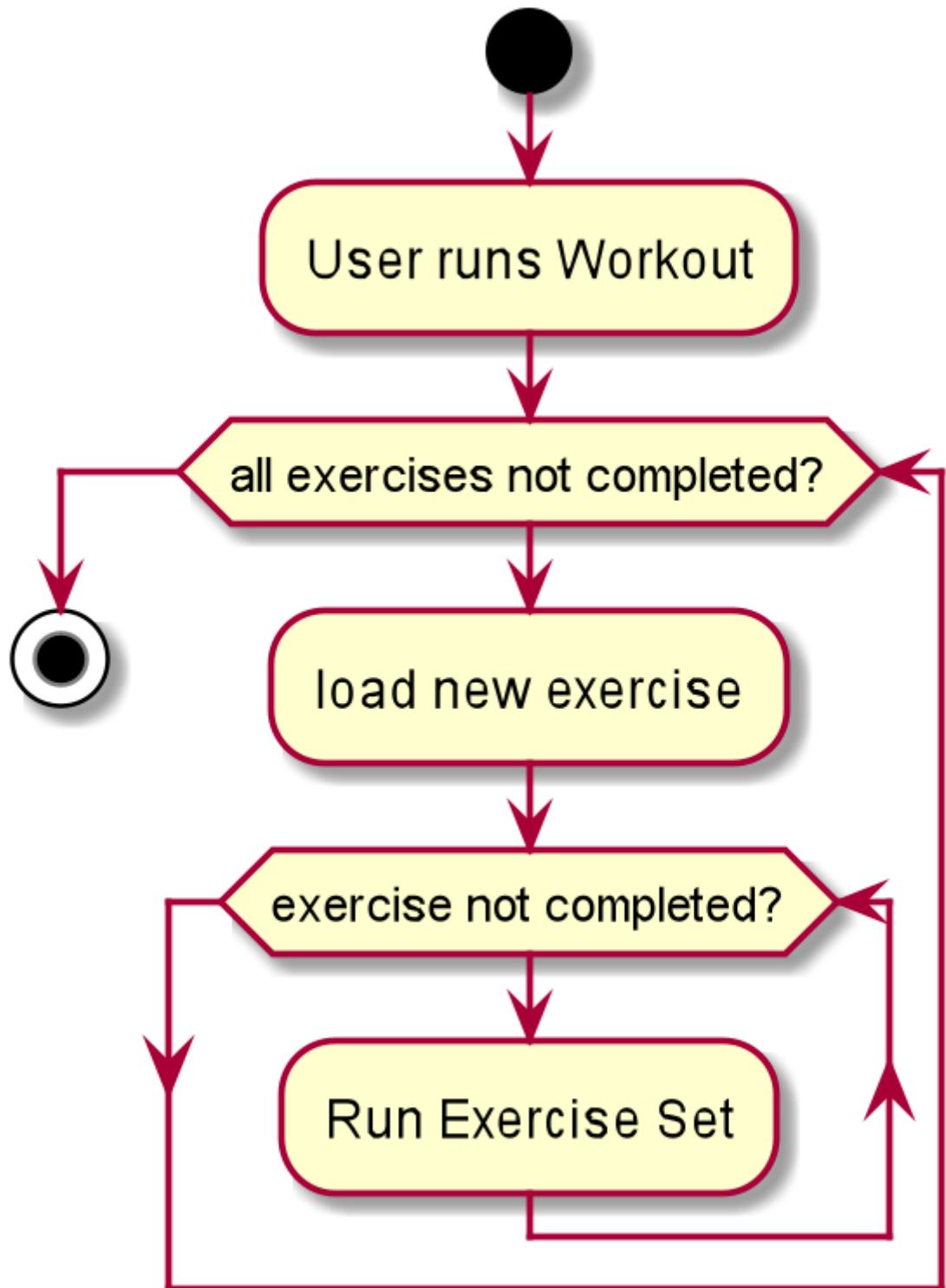


Figure 25. Activity Diagram of RunWorkoutWindow

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.

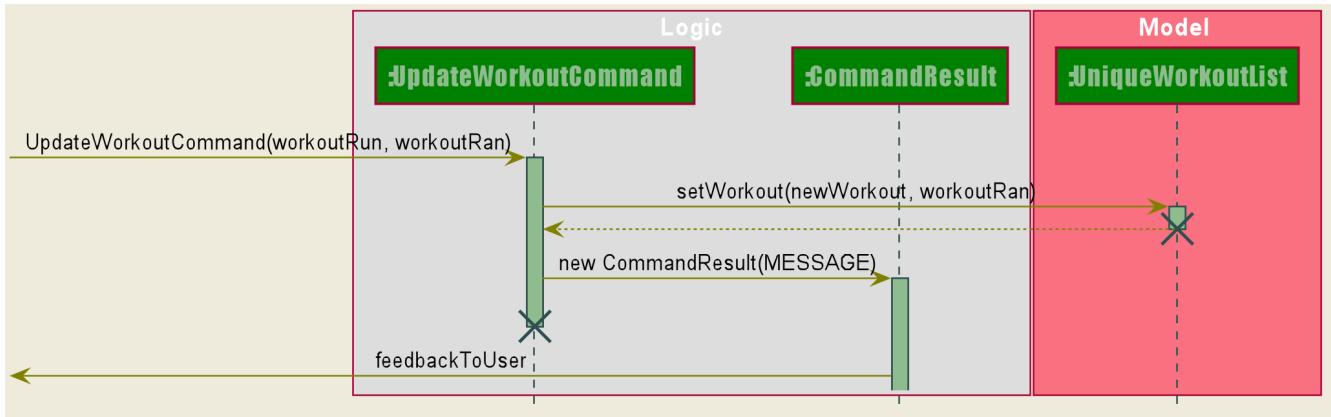


Figure 26. Sequence Diagram of `UpdateWorkoutCommand`

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	<p>Intensity was stored as an Enumeration instead of a class</p> <ul style="list-style-type: none"> <li>- Pros: Intensity can be limited only a specific amount of values</li> <li>- Cons: Intensity will only be an estimate instead of a specific value given the value limits</li> </ul> <p>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.</p>	<p>Setting Intensity as a Class</p> <ul style="list-style-type: none"> <li>- Pros: Easy to implement.</li> <li>- Cons: Makes filtering by intensity a more tedious affair for both developers and users.</li> </ul>

Aspect	Option 1 (Chosen)	Option 2
Storing MuscleTypes	<p>Have MuscleType be a class on its own</p> <ul style="list-style-type: none"> <li>- Pros: Muscles are referred to by various names and allowing the user to set their own muscle names allow for more familiarity</li> <li>- Cons: MuscleType class will require stricter validation to ensure that users do not mess up the programme with unintended inputs.</li> </ul> <p>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.</p>	<p>Store MuscleType as an enumeration</p> <ul style="list-style-type: none"> <li>- Pros: There are limited muscles in the body, allowing for a proper limit</li> <li>- Cons: Muscles may have multiple names that are not accounted for by the enum.</li> </ul>
Storage of Exercises in Workout	<p>Workouts only store a list of ExerciseName and not the full exercise</p> <ul style="list-style-type: none"> <li>- Pros: Exercises only have to be edited once upon execution of edit command - more cost effective. It also avoids unnecessarily large storage files.</li> <li>- Cons: Each time an exercise of workout has to be referenced, the entire storage of exercise has to be scoured</li> </ul> <p>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.</p>	<p>Workouts store whole Exercises</p> <ul style="list-style-type: none"> <li>- Pros : Exercises can be extracted quickly - Cons : Huge storage space is required. Also complicates editing of exercises.</li> </ul>