

What is your project about, who is it for, what does it do + Justifications for my design (Research, feedback, best practices):

What is my project? And how will I ensure it works functionally and correctly?

I want to create a website that others can use to search for exercises and find out more information about them. Because I plan on others using this interface, I will need to ensure it will not be responsible for injuries or unnecessary danger, such as users accidentally using the wrong exercise form and injuring themselves. To do this effectively I will need to ensure I collect correct, safe and up to date information for my exercises. To ensure that my information about exercise is correct, I will use sources like: [Nutrition.gov](https://www.nutrition.gov), [Acefitness.org](https://www.acefitness.org), [Harvard.edu](https://www.harvard.edu), which are more reliable than a website with a less official domain name eg: “.com”. By focusing on reliability and safety, I ensure that I minimise injury, and that users are satisfied with my product. Even though majority information will be from a reliable url, other websites that use a less trustworthy url may still have good information, and so if needed I may include this information if I am able to research it further and ensure that it is safe.

Some personal trainers maintain that personal evidence is more effective than scientific evidence, and so I may need to take this into account when choosing what information to include in my database. I could use both my own experience, my friends, or could talk to personal trainers at my gym if needed to ensure that I gather multiple different perspectives on the information, as well as real world application.

I will need to ensure that the programming is robust and functional to avoid any unnecessary errors. And by following the standards of the programs that I am going to use, I ensure that my code is likely to work as intended, as well as making it easier to understand, and to update and change if needed.

To do this, I researched common SQL database conventions (source: [SQL Shack](https://www.sqlshack.com)), and made note of how I should name the tables and columns in my database.

- Tables should be named in lowercase letters eg: “id” not “ID” or “Id”
- If a name is more than one word long, use an underscore between words eg: “language_id”
- Include a primary key named “id” on each table
- When using foreign keys use names like “table_id” eg: “language_id”.

I also researched standard python coding conventions (source: [PEP8](https://www.python.org/dev/peps/pep-0008)), and made note of how I should type my code such as variable naming conventions, function and loop formatting, and various other standards included in the style guide.

- Use 4 spaces or ‘tab’ when indenting lines of code.
- Limit all lines to a maximum of 79 characters.
- Imports are always put at the top of the file

- Pick between “” and ” when declaring strings and stick to it unless necessary.

Because I am making something that other people are going to use, I need to ensure that the programming is robust and functional to avoid any unnecessary errors, by following the standards of the programs that I am going to use, I ensure that my code is likely to work as intended, as well as making it easier to understand, and to update and change if needed. However, using these rigid conventions could make code harder to interact with, and could involve a learning curve that would result in extra time spent on something that may not be entirely beneficial. I will need to balance following conventions with accessible code, and make decisions about if it's worth trying to learn a new method for something I may already understand.

It can be argued that readability and flexibility is more important than following strict design conventions, and so I will try to take these alternative perspectives into consideration when deciding on variable names and other formatting. For example, while it is said to be necessary to limit lines to 79 characters, it is more important that the code is functional, and so if by shortening a line, the code becomes slower, doesn't work at all, or involves me spending more time learning how to fix it then it may not be worth it to follow these conventions.

Who is it for?

My interface will be majority made for gym goers, or any other people who want to learn more about exercise physiology. Based on my research, the average age for gym goers in the UK is 18-34 (according to [JHSA Health club](#)), so my main target audience will be somewhere around this age range, eg: 15 - 40, and while there will be extremes on either end of that spectrum, more commonly in the upper end, I will try to make my interface as easy to use as possible to ensure that elderly, impaired and my general user base will be able to interact correctly. Although a simpler interface could deter more advanced and experienced gym goers, I think after considering the scope of my project, as well as who my target audience is, I think that focusing on accessibility is more important.

What I found during my research:

- Most common gym going age is 18-34 years old
- Ages 35 to 44, prefer free weights (dumbbells/hand weights), running, and elliptical machines over other forms of exercise.
- Ages 55 to 64 favor walking, stationary cycling, and the use of dumbbells and hand weights.
- Ages 28 and 44, bias yoga, HIIT, and running

Based on these above findings, I think that 16 to 40 years old is a good target age range for my product. I will try to include more dumbbell exercises in my database, as it seems that many prefer using dumbbells, and also especially due to the fact that dumbbell form is much easier to mess up than machines are. I don't think that I need to include anything about walking, or stationary running, as this is just cardio so form does not matter as much, and injury is rare, and no muscles are actually being targeted other than stabilising muscles. I likely also won't include yoga either, as I have little to no experience in that area, and I want my project to be less about cardio and stretching and more about the muscles being used in exercises, and form. Although this does lower the inclusivity of my project, yoga does not actually work any muscles, and so doesn't actually fit within my database. By

understanding who my target audience is, I am able to more effectively design my product to appeal to these users, as well as giving me a better understanding of what users will want me to include, which should help me create a more satisfying and positive experience for users.

Why did I choose this project? What problem am I solving?

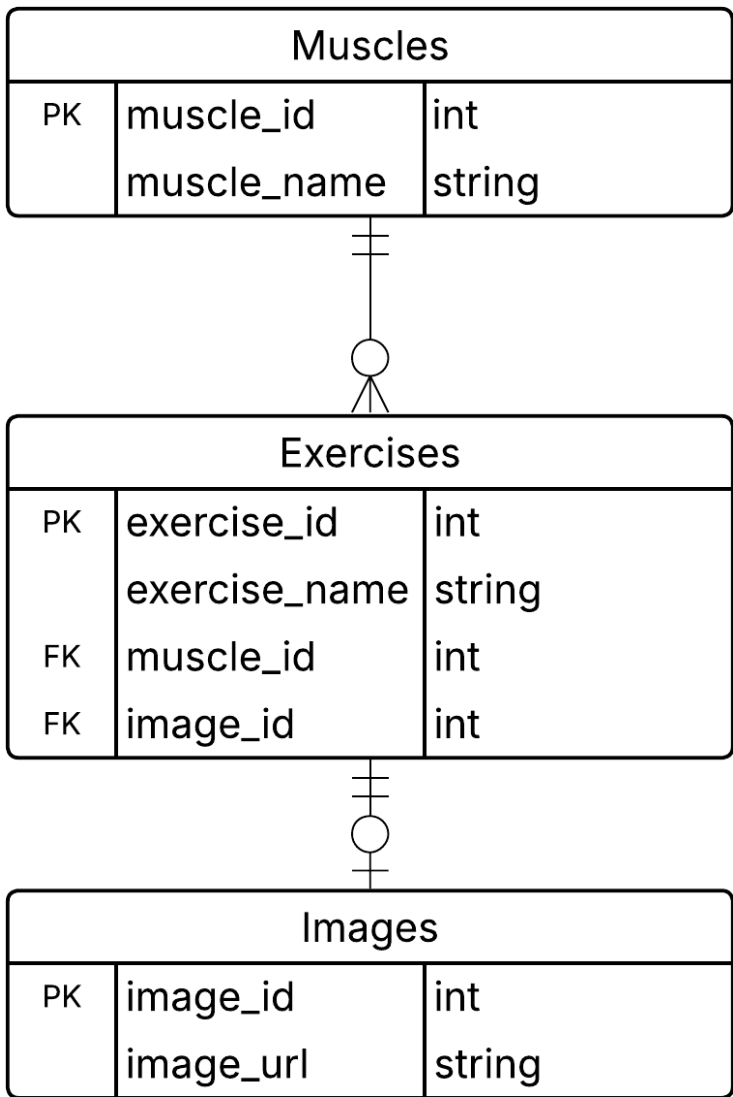
It is time consuming to individually search multiple muscle groups manually into google to find exercises that target them, the results may also be biased or unreliable, as websites could pay to boost the engagement on incorrect or inaccurate information. I want to create an unbiased source of information, as well as to make it easier to search through exercises when designing a split by grouping them into one website, where people can quickly and easily find new exercises, or learn more about exercises they already know about. It can also be costly if gym goers need to pay for a personal trainer to recommend them exercises/form, and although I will never be able to fulfill their in person role, I want to help people know where to start when asking for help, or if I'm lucky, they may be able to use my program in place of a personal trainer if what they need is simplistic, which could save money and time.

Although I don't want my database to completely replace a personal trainer, and more so offer assistance, some users could choose to follow the advice over a trainer, which could lead to injury. To help minimise this, I will include a disclaimer on my page to ensure that no users make this mistake, and even if they do, I have tried my best to help them avoid injury. By considering all the possible outcomes, although unlikely, I am able to prepare for the worst, and ensure safety for my users.

Are there any existing alternatives?

There are similar solutions already available, such as [ExRx.net](https://www.exrx.net), or [functionalmovement.com](https://www.functionalmovement.com). ExRx is extremely conclusive and takes a very scientific approach, listing the exercises by specific muscles targeted eg: "[Triceps Brachii](#)", rather than just "arms" or "triceps". There are also quite a lot of different links to click, which could seem overwhelming. It does a very good job of explaining information once you navigate to the exercise though, including gif/video and written instructions. Overall, ExRx has a lot of extremely conclusive information, as well as clear instructions, however it isn't very intuitive and by using scientific names, as well as including a large amount of links, it may seem overwhelming to new users, appealing more to advanced gym goers. Functionalmovement takes a different approach, opting to focus more on bodyweight/kettlebell workouts. It has a very aesthetically pleasing and intuitive UI, as well as accompanying thumbnail photos for each exercise. However the information is quite limited, and it also focuses more on stretching exercises rather than muscle training, so it fulfills a different niche that I don't think I will focus on. Overall, I like the amount of information ExRx stores, as well as its clear instructions and explanations, however I dislike the inaccessibility due to advanced terminology and overwhelming UI, in comparison, I like the UI and layout of Functionalmovements as well its images and accessibility, however it doesn't include muscular information. I think that I will take inspiration from ExRx's information and instructions, and take inspiration from the aesthetic design of Functionalmovements, and try to avoid including their negatives.

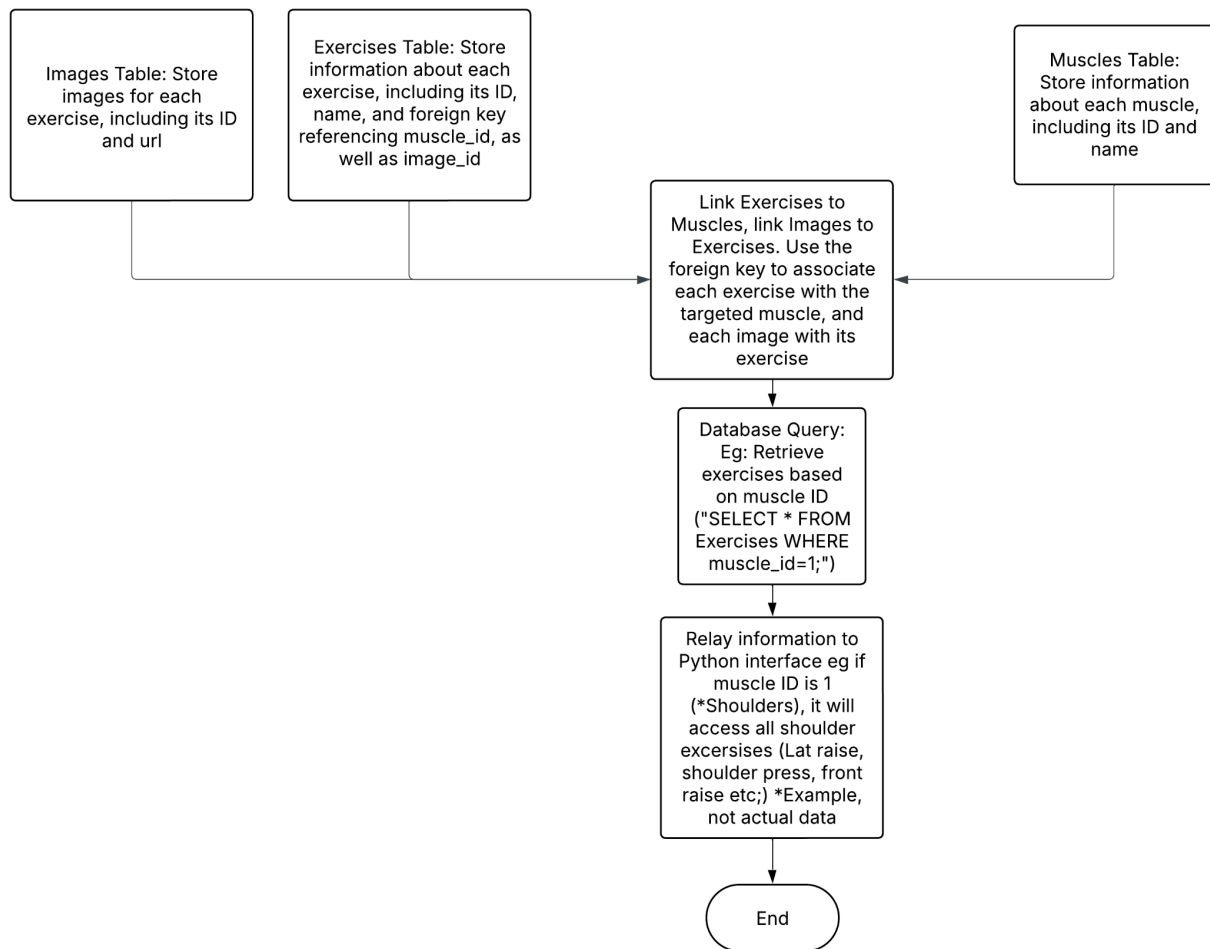
Entity Relationship Diagram:



Muscle and Exercise Relationship:

The database structure consists of three tables: Muscles, Exercises, and Images. Each exercise is linked to a specific muscle through a foreign key relationship, indicating which muscle is targeted by the exercise, each exercise is linked to an image through an ID foreign key.

Muscle and Exercise Database Flowchart:




Website Layout and design prototype:

This is a prototype of what the search page could look like, with an interactable image that allows the user to click on the muscle group they want to search, however if this is outside of my abilities then this could be swapped to a standard search bar. There is a title of the muscle you have searched on the right, and a list of exercises that work that muscle underneath, along with accompanying images. The names of the exercises have been underlined to show that they will be links which will lead to a

more detailed description of the selected exercise.

[Home](#) [Search](#) [Exercises](#)






Click muscle to search: [Chest](#)

CHEST

[Bench Press:](#)
Muscles worked:
Chest, triceps, front deltoids (shoulders)

[Chest Press:](#)
Muscles worked:
Chest, triceps, front deltoids (shoulders)

[Chest Flies](#)
Muscles worked:
Chest, front deltoids (shoulders)



In contrast this image shows the more detailed linked page of the “bench press” exercise, and lists equipment, muscles worked in detail, and technique advice. It also includes further relevant images to demonstrate equipment and form technique.

[Home](#) [Search](#) [Exercises](#)




Image credit: Jun - Getty Images

BENCH PRESS

Bench Press Overview
The bench press is a compound exercise that strengthens the chest, shoulders, and triceps. It is widely used in strength training, bodybuilding, and powerlifting.

Equipment Needed




- Flat Bench – A stable surface to lie on, barbell or dumbbells, weight plates – added resistance for barbells, rack or power cage – holds the barbell securely, collars/clips – keep weight plates in place, spotter – helps ensure safety when lifting heavy.

Muscles Worked

- Primary Muscles:
 - Pectoralis Major (Chest)
 - Triceps Brachii (Back of the Arm)
 - Anterior Deltoids (Front Shoulder Muscles)
- Secondary Muscles:
 - Serratus Anterior (Near the Ribs)
 - Latissimus Dorsi (Upper Back)
 - Core (For Stability)

How to Perform the Bench Press

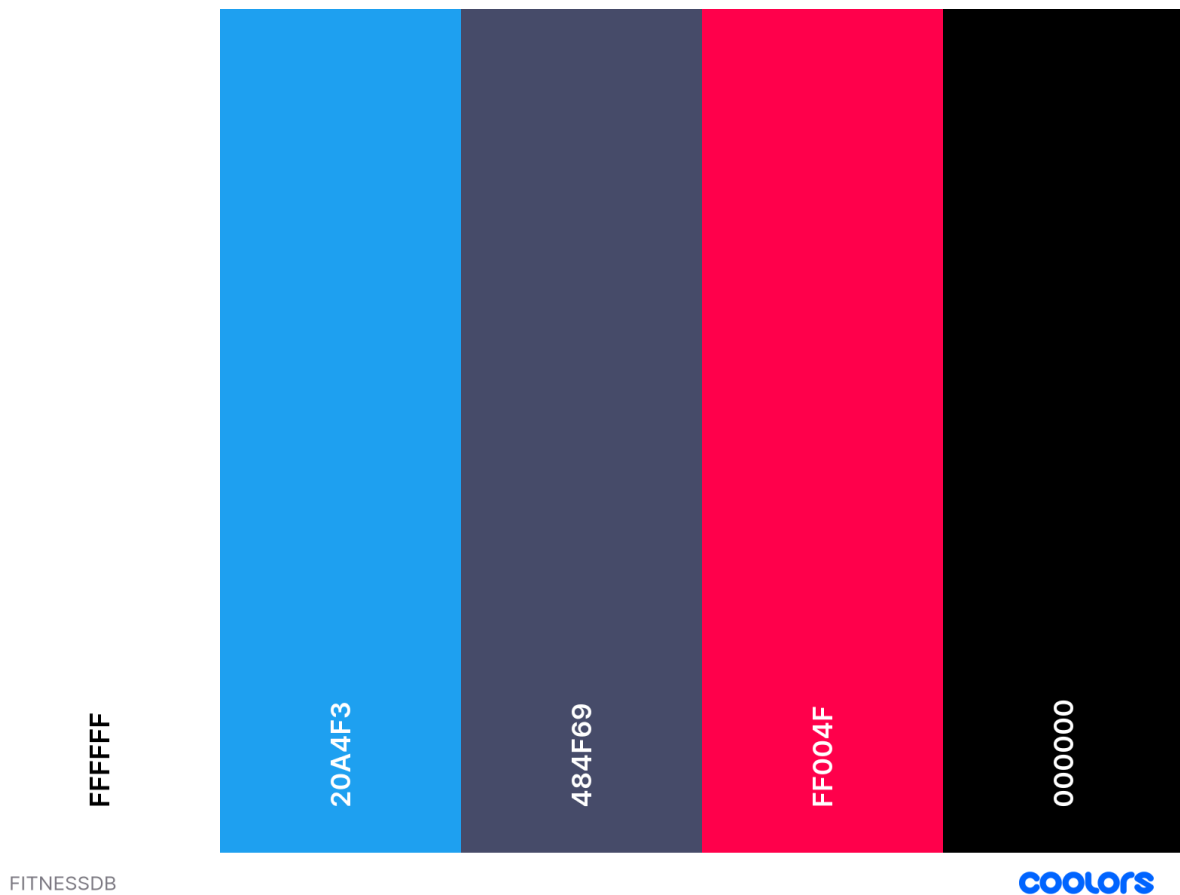
1. Setup: Lie on a flat bench with feet on the ground and back slightly arched.
2. Grip the Bar: Hands slightly wider than shoulder-width.
3. Unrack the Bar: Lift the bar off the rack with arms extended.
4. Lower the Bar: Bring it down to your chest in a controlled motion.
5. Press Up: Push the bar up explosively until arms are fully extended.
6. Lockout & Repeat: Complete the desired reps and re-rack safely.



Fonts:

As shown in my prototype I think using simple and readable fonts like *Lovelo* for headings and *Calibri* for paragraph text. However this is extremely subject to change, and while it may not be these exact fonts, it will be something similar.

Colour Palette:



This is a possible colour palette designed using canva and coolors. I want the main colours to be the pink/red, grey, black and white, however I may use the blue as an extra if needed. I could possibly change the white and black to be less absolute (FFFFFF and 000000) in the future as well, although this should be fairly easy to do.

Routes, Function Signatures and SQL Queries:

Route:

“@app.route(/)” will be the homepage, which will allow you to access the search page, a list of all included exercises, and a list of all included muscles.

SQL Query:

N/A. No SQL at this stage, will just serve as a hub to access the other pages. (May change in future, could show a summary of all information or glossary)

Function Signature:

```
def home():
```

Route:

@app.route("/all_exercises") will be the page linking all exercises, which will lead to my next page I'm mentioning.

SQL Query:

```
"SELECT * FROM Exercises"
```

Function Signature:

```
def all_exercises():
```

Expected results:

A list of all the names of each exercise, formatted as links to pages showing more detailed information.

Eg:

"Exercises:

Face Pulls

Shrugs

Leg Raise

Russian Twist

Crunch

Plank

Calf Raise

Hip Thrust

Glute Bridge

Deadlift

Romanian Deadlift

Leg Curl

Lunges

Leg Press

Squat

Bent Over Row

Pull Up

Lat Pulldown

Chin Up

Bicep Curl

Tricep Pushdown

Tricep Dips

Incline Bench Press

Push Up

Bench Press

Front Raise
Shoulder Press
Lateral Raise”

Route:

@app.route("/exercise/<int:id>") will be a detailed page for each exercise that the user clicks on from the previous page.

SQL Query:

“SELECT exercise_id, exercise_name, muscle_name FROM Exercises
INNER JOIN Muscles ON muscle_id = muscle_id WHERE exercise_id = ?”

Function Signature:

```
def exercise(id):
```

Expected results:

Exercise name, and the name of the muscle it targets based on an ID number.

Eg:

“Exercise:

Leg Raise

Muscle:

Abs”

Route:

@app.route("/all_muscles") will be the page linking all muscle groups, which will lead to my next page I'm mentioning.

SQL Query:

“SELECT * FROM Muscles”

Function Signature:

```
def all_muscles():
```

Expected results:

A list of all the names of each muscle, formatted as links to pages showing more detailed information.

Eg:

“Muscles:

Abs

Forearms

Calves

Trapezius

Lats

Back

Glutes

Hamstrings

Quadriceps

Biceps

Triceps

Chest

Shoulders”

Route

@app.route("/muscle/<int:id>") will be a detailed page for each muscle that the user clicks on from the previous page.

Function Signature

```
def muscle(id):
```

SQL Query

```
"SELECT muscle_id, muscle_name, exercise_name FROM Muscles  
INNER JOIN Exercises ON exercise_id = exercise_id WHERE muscle_id = ?"
```

Expected results:

Muscle name, and a list of exercises that target it based on an ID number.

Eg:

"Muscle:

Abs

Exercises:

Leg raise, russian twist, crunch, plank"