A GMRA Approach to Workout Optimization

Project Report

Evan Wells

Senior Practicum - COSC3997

Introduction

The goal of this senior practicum was to design and implement a workout application that leverages Role-Based Collaboration (RBC) [8] and Group Multi-Role Assignment (GMRA) [7] to dynamically assign exercises based on user input. Users can specify the number of exercises and target muscle groups, enabling them to create personalized and optimal workout plans tailored to their goals for each gym session. This innovative approach allows gym-goers to achieve efficiency and specificity in their training routines.

The application successfully integrates multiple components that demonstrate the E-CARGO model, a framework for effective role collaboration in complex systems. Other components include using a 3D model to gather data and user interface design. As a result, the application provides a robust solution for dynamically selecting exercises across various workout scenarios, showcasing both technical and practical utility.

Implementation

There were numerous large functions and components involved in the implementation of this application. Namely – the 3D model implementation, the GMRA algorithm implementation, the shuffle implementation and the sorting algorithm implementation.

First, the 3D model implementation. This required a lot of research in the documentation of Blender and Three.js. Blender is an open-source modelling application used by many worldwide [1] while Three.js is a framework for implementing it within an application [6]. With the newly acquired knowledge, I was able to create a human body within Blender, isolating muscle groups to prepare for it's use within the program. The Blender file can be found in the assets folder (WorkoutApp/assets/Body2.glb). Implementing the model proved to be a much harder task. With the aid of Three.js, I was able to correctly orient, resize, and rotate the model. However, the rotation of the model was quite skewed, as it relied on a click-drag movement, which is inconsistent across platforms like android and webpage. As such, a rotation button was opted instead of a click-drag rotation. With the implementation of the movement complete, the next step was quantifying the resulting values from the model selection. This was done by quantifying the values relative to one another and normalizing the vector created by the model. This created vector is used as the *role range vector* (L) for the GMRA algorithm.

Next, using the L vector and a Q matrix (can be found in WorkoutApp/components/gra/qMatrix.csv), I set to implementing the Group Multi Role Algorithm. This approach utilized Python's PuLP, which is well known and used to aid in RBC systems. As such, its implementation was relatively simple as I have acquired a good experience within the realm of RBC and linear problem solving. The majority of the issues when it came to implementing the GMRA algorithm was the creating of the *role range vector* and

$$\sigma = max\left\{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} Q[i,j] \times T[i,j]\right\}$$

Subject to:

$$T[i, j] \in \{0, 1\} (0 \leq i \leq m, 0 \leq j \leq n) \quad (1)$$

$$\sum_{i=0}^{m-1} T[i,j] = L[j] (0 \leq j < n) \quad (2)$$

$$\sum_{j=0}^{n-1} T[i, j] \leq L^a[i] (0 \leq i < m) \quad (3)$$

$$\sum_{j=0}^{n-1} L[j] \geq a \quad (4)$$

Figure I. GMRA Formalization

the additional constraints relating to the amount of exercises desired. The algorithm can be formalized as follows:

Where expression (1) states an integer constraint, (2) shows what constitutes a workable group, where (3) limits the amount of roles per agent and (4) ensures there are enough roles for amount of exercises. Through this, the workable assignment matrix returns the optimal exercises for the given input. It's quick runtime and succinct assignment process allows the application to return exact results that are directly applicable to the selected inputs of muscle usage provided by the user. In sum, it showcases the utility of the GMRA algorithm within a real-world problem.

In addition, as the GMRA algorithm doesn't account for human behaviour, such as injuries, lack of necessary equipment and disinterest. To account for this, a shuffle function was required within the application. I first opted for *role transfer* but quickly realized that this approach sets priority to returning a maximal value of σ rather than a exercise similar to the original one. As such, often the exercise swapped had little to nothing in relation to the original one. As a result, I instead implemented cosine similarity, an algebraic method to quantify the similarity between two vectors. Cosine similarity returns a value between 0 and 1, with the higher the value, the more similar [4]. Through this, I can iterate through exercise and find the best fitting exercise as such:

$$Q[new] = \max\left\{\frac{Q[o] \cdot Q[i]}{\|Q[o]\| \times \|Q[i]\|}\right\}$$

Subject to:

$$\sum_{j=0}^{m-1} T[i,j] = 0 \quad (0 \leq j < n) \quad (1)$$

$$i \nexists B \quad (2)$$

Figure II. Shuffle function formalization

Where expression (1) states that the new exercise cannot already have been assigned and (2) indicates that the new exercise cannot exist in the banned set of exercises (*B*). The banned set of exercises simply exists to make sure there is not repeats of exercises when shuffling. Overall, the shuffle function allows the user to shuffle any exercise to replace it with the next closest exercise in terms of muscle use, which is defined within the Q matrix.

Lastly, a need for a sorting function was identified. Although the GMRA algorithm optimally assigns the exercises, it does not consider the order in which its displayed. As such, I implemented the greedy algorithm - it executes as follows. Every non-zero row of the T Matrix is assigned a value of muscle usage. The exercise with the most muscle usage is the first one. The remaining exercises are evaluated and assigned a spillover value. A spillover value is determined by the number of muscle groups used that are also used in the previous exercise. Whichever exercise returns the smallest spillover value is used next. This process continues until no exercises remain. Essentially, the algorithm uses the best possible option at each iteration. Since most workouts range between 3 to 9 exercises, this method of sorting is proficient in sorting the exercises properly.

Running Results

In this section, screenshots of the overall program with an explanation of the screenshots will be provided.

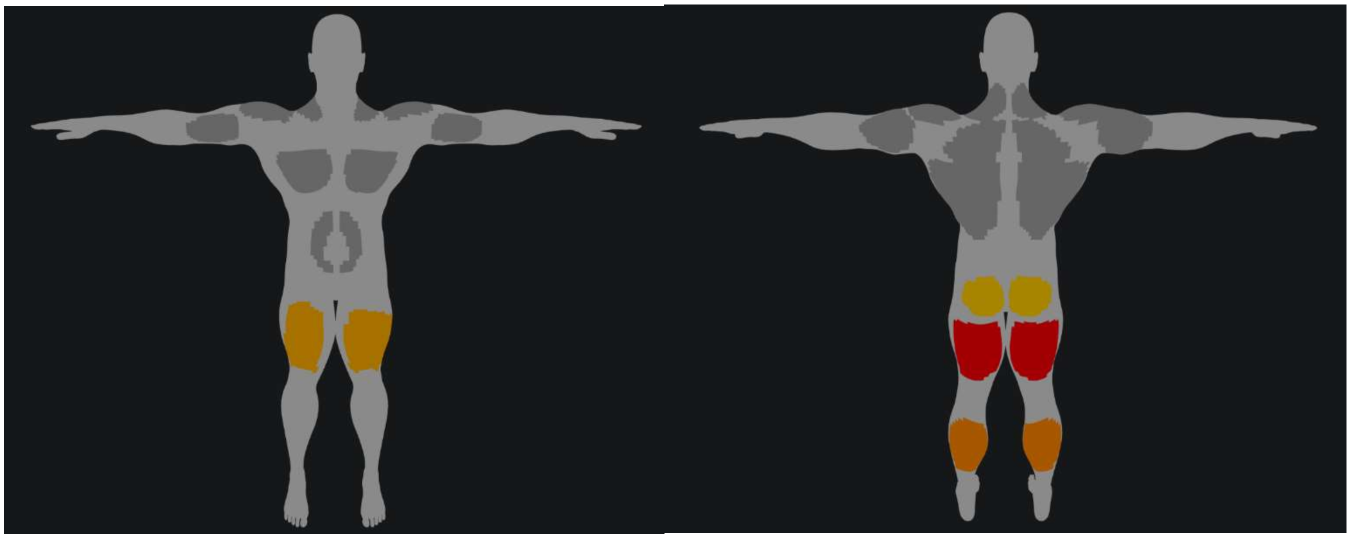
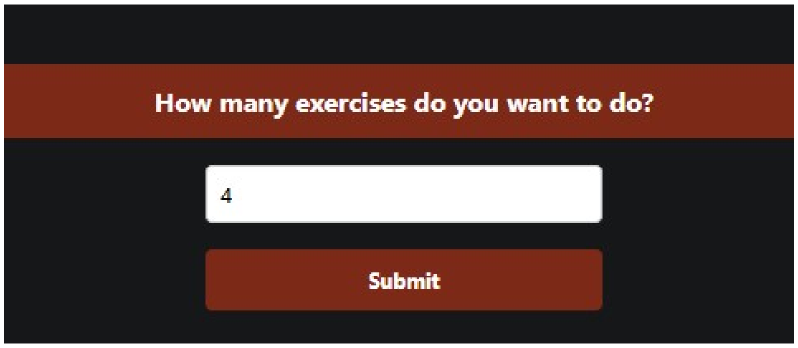


Figure III. Screenshot of selecting muscle groups from model

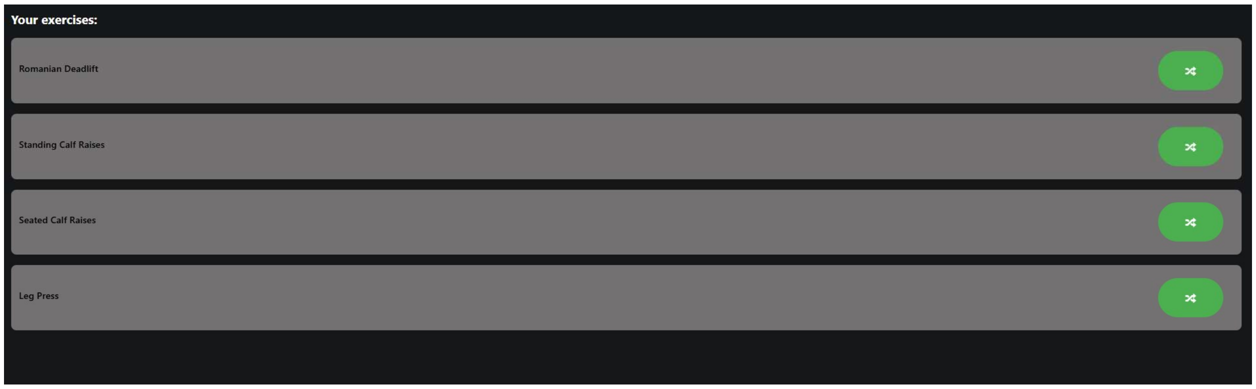
Figure III shows an example of muscle selection by a user. In this case, the user wants to focus their hamstrings more than anything else. As such, they have selected hamstrings till it is dark red. The values quantified from this model and the amount of exercises will determine the *role range vector*.


How many exercises do you want to do?

4

Figure IV. Screenshot of selecting amount of exercises

Figure IV shows the user picking the number of exercises they would like to do. In this case, they would like to do 4.


Your exercises:

Romanian Deadlift

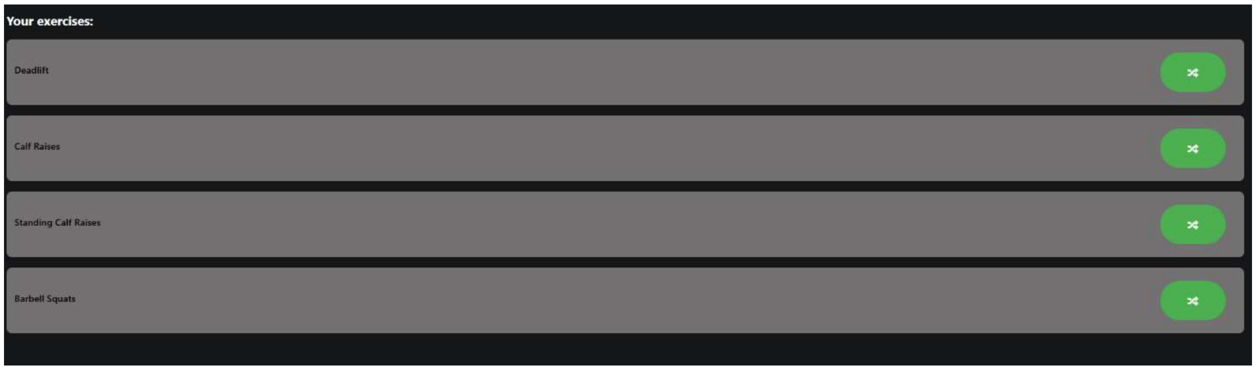
Standing Calf Raises

Seated Calf Raises

Leg Press

Figure V. Screenshot of exercises picked by GMRA algorithm

Figure V demonstrates the results of the GMRA algorithm with the given inputs from Figures III and IV. The exercises picked are relevant and evidently, optimal for the given inputs made on the 3D model.


Your exercises:

Deadlift

Calf Raises

Standing Calf Raises

Barbell Squats

Figure VI. Screenshot of exercises after shuffling all the previous exercises

Figure VI exemplifies the usage of the shuffle function by shuffling every exercise from Figure V. When comparing the names of the old and new exercises, you can infer that they are similar. For example, *Romanian Deadlift* switches to *Deadlift*.

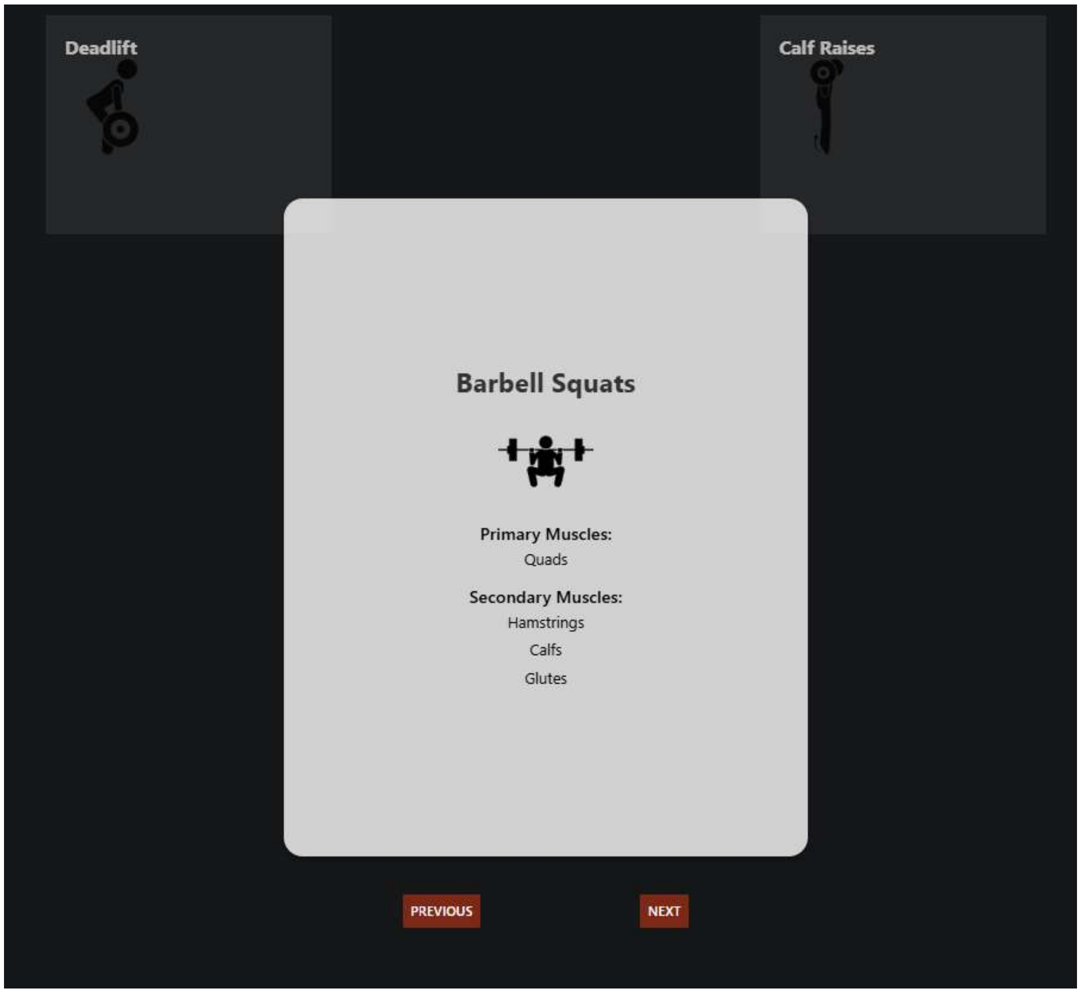


Figure VIII. Screenshot of UI while doing the exercise and showcase of sorting

Figure VII showcases the user interface of iterating through the exercises during the workout. As can be seen, you can move through the exercises with the “previous” and “next” buttons. On the current exercise, the name of the exercise is displayed, an icon relevant to the exercise is shown and the muscles used are on display. As can be seen, Quads are the primary muscles, with hamstrings, calves and glutes as secondary muscles. This data is all calculated by viewing the row relating to this exercise in the Q matrix.

Conclusion

This project was an enriching and rewarding experience, allowing me to explore new concepts, learn great libraries, and challenge my problem-solving abilities. The workout application not only runs efficiently but also integrates the GMRA algorithm with supporting structures that enhance its functionality and reliability.

The user interface, a significant improvement over my previous IT ticket assignment project, demonstrates my growth in designing intuitive and visually appealing UIs. The project's success highlights my ability to overcome challenges, effectively manage resources, and achieve ambitious goals. Overall, this senior practicum has equipped me with invaluable skills and insights that I am excited to carry forward into future projects and research.

In sum, I achieved the goals I set for myself when initially starting this project. I further familiarized myself with Role Based Collaboration systems, implemented a full stack application and created an app I can use for myself.

References

[1] Blender Foundation. *Modeling meshes*. Blender Manual. https://docs.blender.org/manual/en/latest/modeling/meshes/index.html

[2] FreeTrainers. *Muscle exercises*. FreeTrainers. https://www.freetrainers.com/exercise/muscle/

[3] GeeksforGeeks. *Greedy algorithms*. GeeksforGeeks. https://www.geeksforgeeks.org/greedy-algorithms/

[4] LearnDataSci. *Cosine similarity*. LearnDataSci. https://www.learndatasci.com/glossary/cosine-similarity

[5] OpenAI. (2024). *ChatGPT.* https://chat.openai.com/

[6] Three.js. *Three.js documentation*. Three.js. https://threejs.org/docs/

[7] Zhu H, Liu D, Zhang S, Teng S, Zhu Y. *Solving the Group Multirole Assignment Problem by Improving the ILOG Approach*[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017, 47(12): 3418-3424

[8] Zhu H, *E-CARGO and Role-Based Collaboration: Modeling and Solving Problems in the Complex World*, Wiley-IEEE Press, NJ, USA, Dec. 2021.

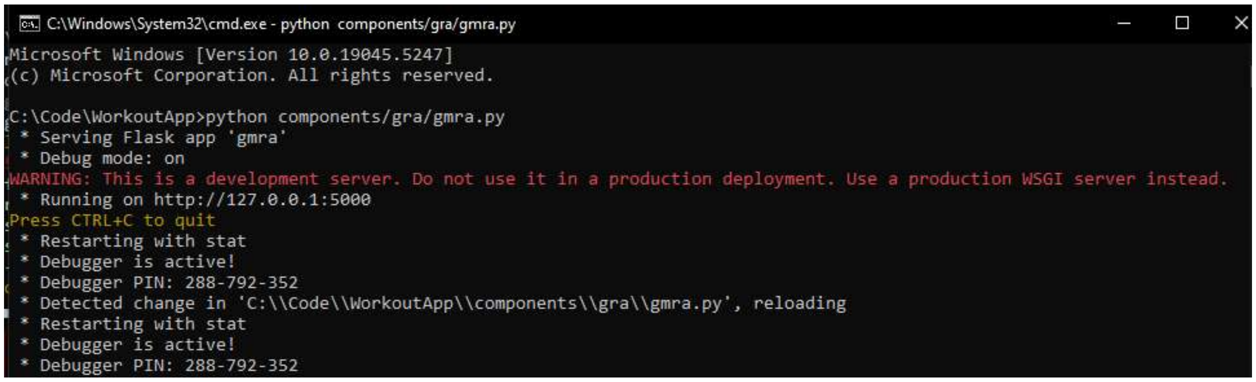
<u>Appendix: How to run the application</u>

To start the application, you must open two separate command prompts in the project directory and run the following commands:

yarn expo start

python components/gra/gmra.py

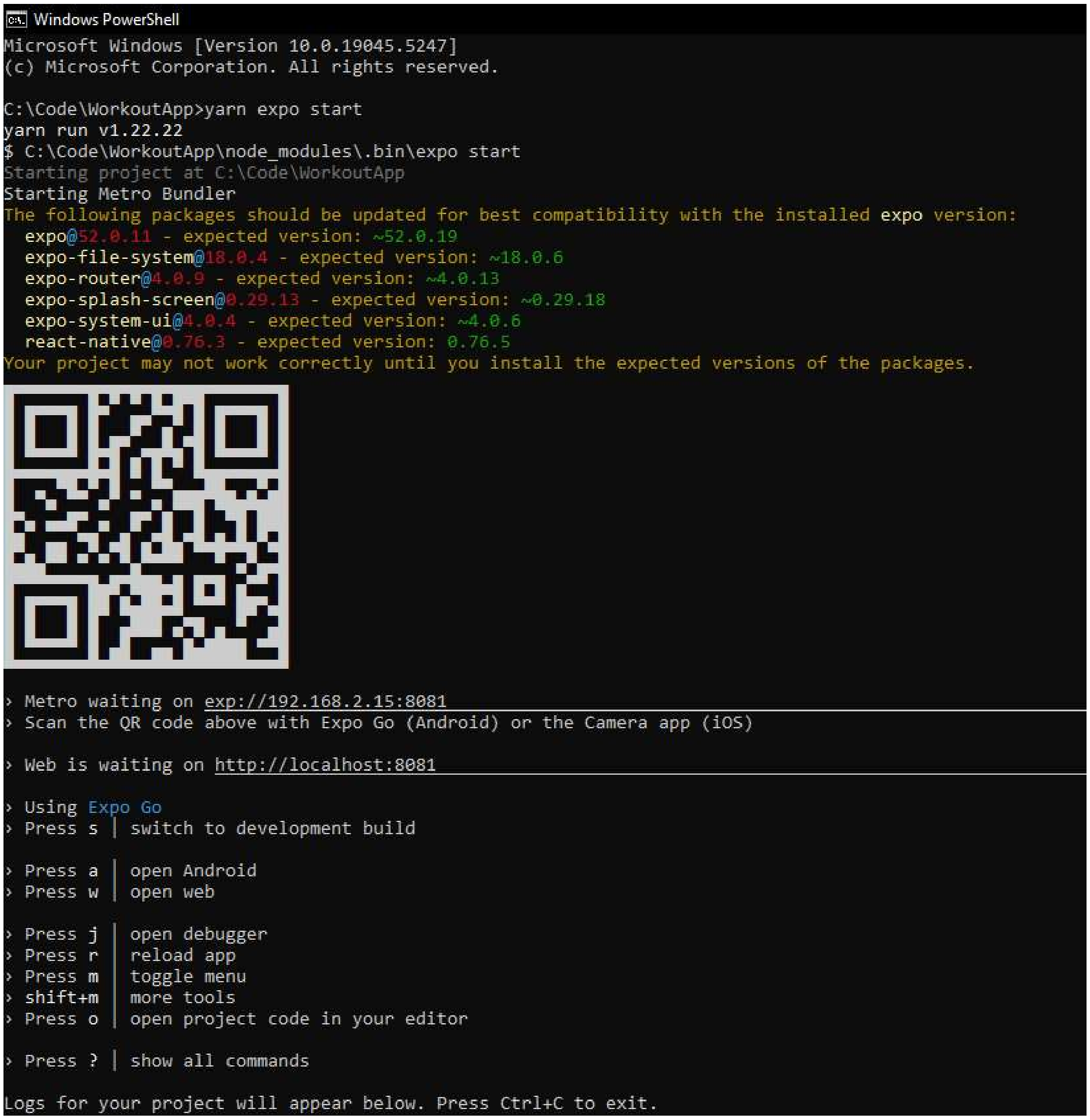
On the command prompt where you ran “yarn expo start”, you can press ‘W’ to open the app or ‘O’ to open the source code.


C:\Windows\System32\cmd.exe - python components/gra/gmra.py

```
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

C:\Code\WorkoutApp>python components/gra/gmra.py
 * Serving Flask app 'gmra'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 288-792-352
 * Detected change in 'C:\\Code\\WorkoutApp\\components\\gra\\gmra.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 288-792-352
```

Figure VIII: Screenshot of the console for the backend functions



```
Windows PowerShell
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

C:\Code\WorkoutApp>yarn expo start
yarn run v1.22.22
$ C:\Code\WorkoutApp\node_modules\.bin\expo start
Starting project at C:\Code\WorkoutApp
Starting Metro Bundler
The following packages should be updated for best compatibility with the installed expo version:
  expo@52.0.11 - expected version: ~52.0.19
  expo-file-system@18.0.4 - expected version: ~18.0.6
  expo-router@4.0.9 - expected version: ~4.0.13
  expo-splash-screen@0.29.13 - expected version: ~0.29.18
  expo-system-ui@4.0.4 - expected version: ~4.0.6
  react-native@0.76.3 - expected version: 0.76.5
Your project may not work correctly until you install the expected versions of the packages.

› Metro waiting on exp://192.168.2.15:8081
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

› Web is waiting on http://localhost:8081

› Using Expo Go
› Press s │ switch to development build

› Press a │ open Android
› Press w │ open web

› Press j │ open debugger
› Press r │ reload app
› Press m │ toggle menu
› shift+m │ more tools
› Press o │ open project code in your editor

› Press ? │ show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
```

Figure IX: Screenshot of the console for the application