Asnapp - Workout Video Recommender

Amanda Shu, Najeem Kanishka, Peter Peng

Project Description

For those who work out at home, finding a good workout routine is difficult. It is hard to constantly have to find workout videos that meet your fitness needs, as well as your time and equipment constraints. Plus, the pre-set workouts found online are not one-size-fits-all. In other words, while there are many workout videos online to choose from, these choices are non-personalized. Our project, Asnapp, is a solution that can solve all these problems, as well as meet the growing need for an easy way to build a good at-home workout routine during the pandemic.

Asnapp is a web application that provides personalized recommendations of workout videos by Fitness Blender[1]. Our website displays several lists of recommendations (similar to Netflix's user interface), such as "top upper body workouts for you". Users can login into our website, choose between several models to generate their recommendations, browse through personalized recommendations lists, and choose a workout to do, saving them the time and effort needed to build a good workout routine.

Report Overview

From Fitness Blender's website, we scrape workout related data from all of their free workouts, as well as the Fitness Blender users' comments on each of their workout webpages. We use the comments data to avoid the cold start problem, as it serves as a proxy for user-item interactions needed to train the models with. Thus, we would be able to train models even before building our web application and deploying to users. We further describe the data collection and preprocessing steps in the Data section.

We then perform offline testing and evaluate three recommendation models (random, top popular, and a pure collaborative filtering model by LightFM[2]). The models are trained with the user-item interaction information from the comments data and evaluated on the NDCG metric (normalized discounted cumulative gain). We describe our model implementations and their performance results in the Models section.

Finally, we built a web application, where users can login and view their workout video recommendations. The application is connected to a database that collects our users' preferences and workout history information. The database has our previously scraped workout video and comments data uploaded to it as well. The three models we previously evaluated are deployed, and users are able to choose which model to get recommendations from. Since we do not expect to have enough user-item interactions data from our own users, we train the models on both the scraped comments data and the interactions data from Asnapp

[1] Fitness Blender is a company that provides free workout videos.
[2] LightFM is a Python package containing implementations of recommendation algorithms.

users in order to generate our predictions. The recommended workout videos displayed will be only those that match the workout preferences that the user initially inputted, providing further personalization. We describe the details and implementation of our web application in the Web Application section.

In the Conclusion section, we summarize Asnapp's value as well as discuss drawbacks and potential improvements.

Data[3]

Data Source: We choose Fitness Blender as our data source, as they have 580 free workout videos on their website and a large user base (6 million subscribers on Youtube). This means there will be a sufficient amount of workout videos to recommend and enough user-item interactions data to train our models with. Fitness Blender's website contains a separate webpage for each of their workouts, with details on that specific workout, such as the duration, calorie burn, difficulty, required equipment, training type, and body focus. These webpages also contain an embedded youtube video and comments from their users (see Appendix A for screenshots of a workout's webpage on Fitness Blender).

Workout Details: We implement a scraping script that collects details for every free workout on Fitness Blender. We then clean and preprocess the data, including getting rid of special characters in strings, applying type conversions for strings meant to be numerical values, and performing one hot encoding of the *equipment*, *body_focus*, and *training_type* columns. The workout video attribute details are then written to the data files *fbworkouts_clean.csv* and *fbworkouts_meta.csv* (see Table 1 and 2 in Appendix B for all column names and descriptions).

Youtube Data: Furthermore, with the youtube video ids (interpreted from the youtube links we scraped), we then query the Youtube API to gather the workout titles, date posted, and number of likes, shares, and comments for each workout's Youtube video. The data is written to *workouts_yt.csv*.

Comments Data: We also scrape the comments section on each Fitness Blender workout webpage. For each comment, we gather the username and profile picture (if the user has a profile picture, this value is a link, otherwise it is a single letter) of the person commenting as well as the time the comment was posted. During the scraping process, the username and profile picture is combined to create a *hashed_id*. During preprocessing, we create our own *user_id* (integer value) based on the *hashed_id*. We also drop duplicates such that Fitness Blender users who commented on the same workout multiple times will not have that user-item pairing show up more than once in the dataset. We also omit any user who has less than 5[4] interactions (commented on less than 5 videos). The final data file that is outputted, *user_item_interactions.csv*, contains user-item pairings with columns *user_id* and *workout_id*.

[3] See our basic exploratory data analysis of the data here

[4] There is a parameter *d* in our preprocessing function to choose the minimum number of interactions needed to keep the user in the dataset. When d=5, there are 52073 total interactions from 4026 users with 580 workout videos.

Model Input: For performing offline testing on the user-item interactions data inferred by Fitness Blender's commenters, we first perform a random 70%-30% split on the data to get training and testing datasets. We use LightFM's Dataset class to generate the LightFM model's inputs, which are user-item interaction matrices.

Models

We implement three models as described below:

Random: This recommends workouts randomly. For each workout, we randomly assign a score between 0 and 1.

Top Popular[5]: This recommends the most popular workouts based on the number of comments on a workout's Fitness Blender webpage. The score assigned to each prediction is the number of comments.

LightFM: This is LightFM's pure model, which is a traditional collaborative filtering[6] matrix factorization method.

We evaluate these models on the NDCG metric (normalized discounted cumulative gain) using scikit-learn's implementation. NDCG is frequently used to measure the relevance and ordering of ranking tasks. It is an appropriate metric to use because our objective is to recommend the most relevant workouts to a user and thus the ordering of our predictions matter.

We train the models on the training user-item interaction data (users with less than 5 interactions are dropped) and evaluate NDCG (with only the top 20 scores[7] considered) on the testing data[8]. We note that for evaluating these models, we use LightFM's method of creating the testing user-item interactions matrix as the true label input of the evaluation function (*y_true* in scikit-learn's *ndcg_score*). Since LightFM uses their internal indices to identify a workout (these differ from the workout ids outputted by our top popular recommender), for evaluating top popular, we also add a mapping function that maps the workouts ids predicted by the model to LightFM's internal indices. The results are displayed in Table 1 below.

Table 1: Model Results

Model	NDCG@20
Random	0.017
TopPop	0.099
LightFM	0.098

[5] We also implement an extension of top popular that includes the data from the Youtube API, but find that it performs worse (see our notebook here)

[6] A collaborative filtering system recommends to a user items that similar users have interacted with but the user has not previously interacted with

[7] k=20, where k is a parameter to scikit-learn's ndcg implementation

[8] We also compare scores across various choices of d and k here.

Although the top popular model and LightFM's pure collaborative filtering model both perform better than the random recommender (which is as expected), the top popular model and LightFM's model perform approximately the same. This is surprising as we expect collaborative filtering to perform better than a top popular recommender, since the top popular recommender is non-personalized. We believe this is a result of a vastly sparse dataset. We also note that we attempted to use the vanilla collaborative filtering algorithm and KNN[9], which scored roughly the same as random guessing and is consequently not included. This outcome informed us about the possible sparsity of interactions in the data, and the above results further reaffirm it.

Despite these results, we still choose to deploy all three of these models. Although the random recommender has poor performance with NDCG, we believe that it provides users with the option to diversify their workout routine. Also, even though the top popular recommender is non personalized, we believe that people are naturally drawn to trying what is popular. We note that Netflix's recommendation system also has a top popular recommendation, so is it not unreasonable to assume that the top popular model will perform well with our users online. With LightFM's collaborative filtering algorithm, we believe it is possible that with the addition of user-item interaction data coming from our own users, the model will perform better.

Web Application

We built a web application which allows users to create an account, login, choose a recommendation model, view their recommendations, and like/dislike workouts they've completed. The back-end is developed using Flask and the front-end with HTML, CSS, and Bootstrap. The application is connected to a MySQL database using Amazon RDS and is deployed with Heroku. The Fitness Blender workouts and comments data collected, as described in the Data section, are all uploaded to the database (see Tables 2, 4-5 in Appendix C).

Login/Registration: When users come to our website, they can either login as an existing user or register themselves as a new user (see Screenshot 1 in Appendix D). During registration, the user is asked for their name, email, password, available equipment, as well as their preferred training types, minimum and maximum calories burned, and duration of workouts (see Screenshot 2 in Appendix D). The registration form has validators to ensure that no fields are not filled out, the password and confirm password fields match, and the ranges for calories and duration are reasonable. After registering, the user's information is inserted into the users table in the database, with their password hashed to ensure privacy (see Table 1 in Appendix C for user table column information). On the login page, users fill out email and passwords, and the form validates whether the email exists in the database and that the password matches. Aftering logging in, the user's id is recorded as session data, so that users are still logged in even when revisiting the website (assuming they do not log out).

Model Deployment: Once logged in, users are redirected to the recommendation page, where they are able to choose which model to generate their recommendations with (see Screenshot 3 in Appendix D). The three models we previously evaluated are deployed (displayed to the user as "Random", "Most Popular", and "Recommended for you"), using SQL statements for the

[9] Our implementation of the KNN collaborative filtering approach can be found here

random and top popular recommenders and our previously implemented python function for the LightFM model. Similarly to before, since the predictions of LightFM use their internal indices to identify workouts and users, we also add mapping functions that maps LightFM's internal item and user indices to the workout and user ids in our data in order to display the corresponding workouts for the appropriate user. Both the top popular and LightFM models are given the combined user-item interactions data inferred from scraped comments on FitnessBlender and interactions recorded from our users on our website as the input to the models.

Recommendations: After choosing a recommender, users click a button to view their recommendations (see Screenshot 4 in Appendix D). The recommender's predicted workouts are filtered to keep only ones that match the user's preferences specified during the registration process. Specifically, the only workouts kept are the ones that have either no equipment listed or lists equipment that the user has. Similarly, only workouts that match the user's preferred training types and are within their preferred difficulty range and workout duration are kept. For calorie burn, if there is some overlap between the user's preferred range and the workout's estimated range, then the workout is kept. The filtered recommendations are displayed in four lists for the categories upper body, lower body, core, and total body. Each list shows the top nine ranked workout videos in the respective category (36 recommendations total). Users can click onto a specific workout, which displays a popup to start the workout's Youtube video. Inside the popup, users can skip to the next video in the lists and have the option to like or dislike the workout (see Screenshot 5 in Appendix D). We also allow users to change the display of workouts, such as only including certain body focuses and choosing how many workouts to show on the screen (see Screenshot 6 in Appendix D).

Collecting user interactions: If the user clicks on the like button in the popup, this information is recorded into the *user_item_interaction* table (Table 2 in Appendix C). We note that we only consider clicking the like button as an interaction to be added into the table. Thus, users need to click like on at least one workout in order for our LightFM model to be executed[10]. If a user has no previous interactions and choses the "Recommended for you" option, we default to using a random recommender. If the user clicks on the dislike button, this is added into a table in our database as well (Table 3 in Appendix C). We also filter out workout recommendations that a user has disliked.

Updating Preferences and Interaction History: Users can also update their preferences with the "Update Preferences" feature (see Screenshot 7 in Appendix D). This takes users to a form similar to the original registration page, but it is prefilled to include their current workout preferences. When submitted, the user's information in the database is updated. Furthermore, the "Workout History" feature (see Screenshot 8 in Appendix D) takes users into a page that is similar to the recommendation page but has the option for users to view their previously liked and disliked workouts. This is particularly useful for users who may want to redo a workout they previously liked or un-dislike a workout they have previously disliked.

[10] The LightFM model is trained on the *user_item_interaction* table, so the user must first exist in that table in order to get the user's predictions.

Next Steps/Conclusion

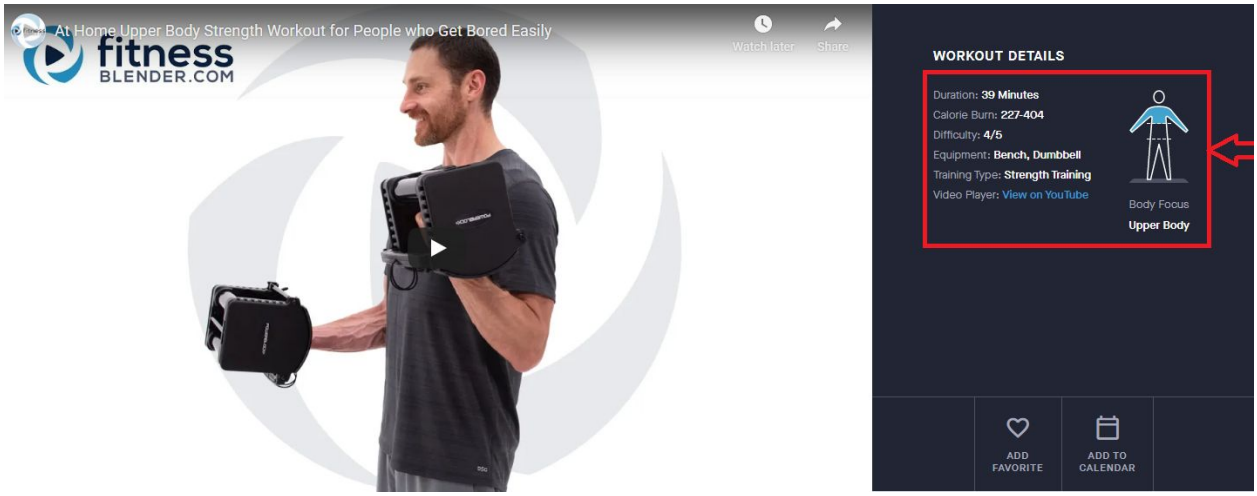
We acknowledge there is uncertainty as to whether the NDCG performance of the models during offline testing (with only data from the Fitness Blender comments), will be similar to the performance of our deployed recommendation system (which uses the addition of Asnapp users' data). If the user behavior of the Fitness blender comments matches the behavior of Asnapp users, we would expect similar results. However, this would be a strong assumption, so it is possible that if Asnapp were to gain more users, we would see certain models either perform better or worse than seen in offline testing. We would then need to further adjust our chosen models accordingly.

Another potential future improvement to Asnapp would be to add more recommender options for users. For example, we could add a content based recommender that recommends workouts that are similar to the user's most recent workout (or perhaps the average of their liked workouts). With this model, the score would be a similarity measure between the workouts based on duration, calorie burn, difficulty level, etc. If we also gathered the workout descriptions, we could perform text mining and take into account the similarity of the workout descriptions as well. A content based recommender would be useful because it would give personalized recommendations that are purely based on the user's history and not other users (unlike our collaborative filtering model). This would be appealing to users who want to understand exactly where their recommendations are coming from.

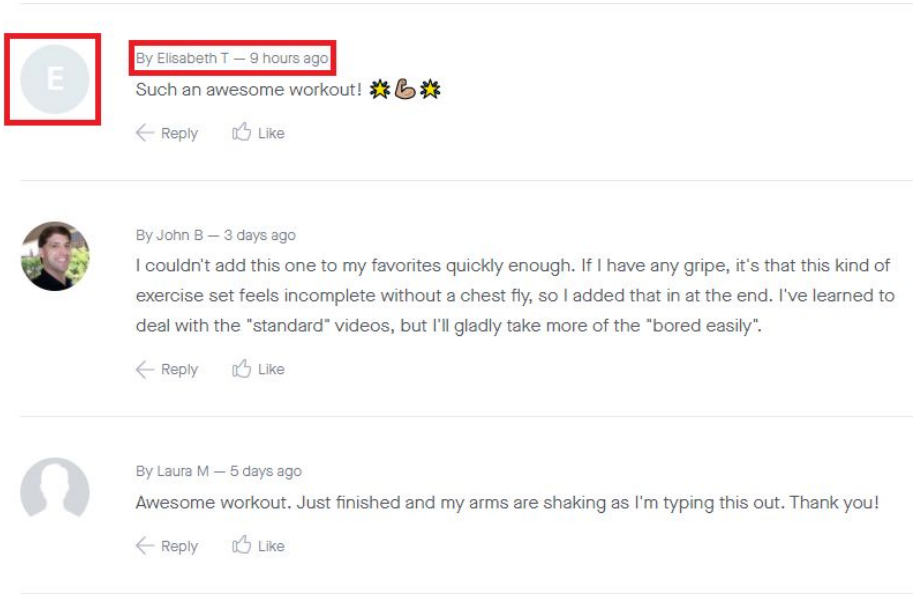
All in all, with several recommendation options and the inclusion of filtering for user preferences and needs, as well as a user friendly and intuitive interface, Asnapp is poised to give value to anyone looking to find workout video recommendations. We hope that Asnapp is able to provide people with an easy and engaging way to build their at-home workout routines.

Appendix

Appendix A: Fitness Blender Webpage



Screenshot 1: *Workout Details on a Fitness Blender Workout Webpage*
(red box is the data collected by scraper)



Screenshot 2: *Comments on a Fitness Blender Workout Webpage*
(red box is the data collected by scraper)

Appendix B: Data Tables From Fitness Blender

Table 1: Fitness Blender Workouts Data (*fbworkouts_clean.csv*)

Column Name(s)	Description
workout_id	Unique identifier assigned to each workout video
duration	Duration of workout video in minutes
min_calorie_burn	Minimum of estimated range of calories burned
max_calorie_burn	Maximum of estimated range of calories burned
difficulty	Integer between 1-5 (5 being most difficult)
equipment	Equipment required for the workout (strings are pythonic, i.e jump_rope)
training_type	Type(s) of training as classified by Fitness Blender (strings are pythonic)
body_focus	Part(s) of body that workout focuses on, as classified by Fitness Blender (strings are pythonic)
core, lower_body, total_body, upper_body	One hot encoding of body_focus
balance_agility, barre, cardiovascular, hiit, low_impact, pilates, plyometric, strength_training, stretching_flexibility, toning, warm_up_cool_down, aerobics_step	One hot encoding of training_type
barbell, bench, dumbbell, exercise_band, jump_rope, kettlebell, mat, medicine_ball, physioball, sandbag, stationary_bike, no_equipment	One hot encoding of equipment

Table 2: Fitness Blender Workouts Metadata (*fbworkouts_meta.csv*)

Column Name	Description
workout_id	Unique identifier assigned to each workout video
workout_title	Workout title on the corresponding Youtube video
fb_link	Link to the Fitness Blender webpage of the workout
youtube_link	Link to the Youtube video for the workout
equipment	Equipment required for the workout (strings are human readable, i.e Jump Rope not jump_rope)
training_type	Type(s) of training as classified by Fitness Blender (strings are human readable)
body_focus	Part(s) of body that workout focuses on, as classified by Fitness Blender (strings are human readable)

Table 3: Youtube Workout Video Data (*workouts_yt.csv*)

Column Name	Description
workout_id	Unique identifier assigned to each workout video
title	Workout title on the corresponding Youtube video
published_at	Time published on Youtube
view_count	Number of views on Youtube
like_count	Number of likes on Youtube
dislike_count	Number of dislikes on Youtube
comment_count	Number of comments on Youtube

see Youtube Data API documentation

Table 4: Fitness Blender Interactions (*user_item_interactions.csv*)

Column Name	Description
user_id	Unique identifier assigned to each user
workout_id	Unique identifier assigned to each workout video

Appendix C: Tables in Our Database[11]

Table 1: *users*

Column Name(s)	Description
user_id	Unique identifier assigned to each user. Starts at 5000 (so it does not overlap with user ids from Fitness Blender comments)
name	Name as inputted in registration form
email	Email as inputted in registration form
password	Hashed password
equipment	User's available equipment (strings are pythonic, i.e jump_rope). Empty string if user has no preferred equipment.
training_type	User's preferred training types (strings are pythonic)
min_duration	User's preferred minimum duration of workout
max_duration	User's preferred maximum duration of workout
min_calories	User's preferred minimum calorie burn of workout
max_calories	User's preferred maximum calorie burn of workout
min_difficulty	User's preferred minimum difficulty of workout
max_difficulty	User's preferred maximum difficulty of workout
balance_agility, barre, cardiovascular, hiit, low_impact, pilates, plyometric, strength_training, stretching_flexibility, toning, warm_up_cool_down, aerobics_step	One hot encoding of training_type
barbell, bench, dumbbell, exercise_band, jump_rope, kettlebell, mat, medicine_ball, physioball, sandbag, stationary_bike	One hot encoding of equipment, without no_equipment

[11] Our SQL statements for the creation of these tables can be found here

Table 2: *user_item_interaction*

Column Name(s)	Description
user_id	Unique identifier assigned to each user. Contains ids from both the inferred users by Fitness blender comments and Asnapp users.
workout_id	Unique identifier assigned to each workout video

Note: The schema matches Table 4 in Appendix B as *user_item_interactions.csv* file was directly uploaded into the database, but additional data is inserted into this table as our users' interactions are recorded.

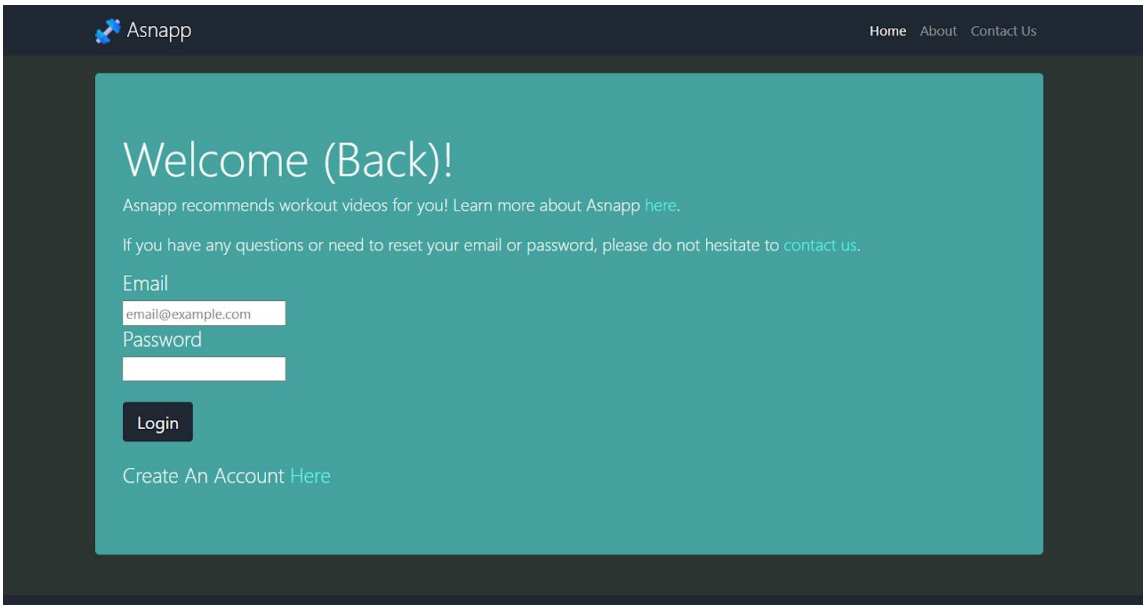
Table 3: *user_disliked_items*

Column Name(s)	Description
user_id	Unique identifier assigned to each user. Starts at 5000 (same user_id from *user* table)
workout_id	Unique identifier assigned to each workout video

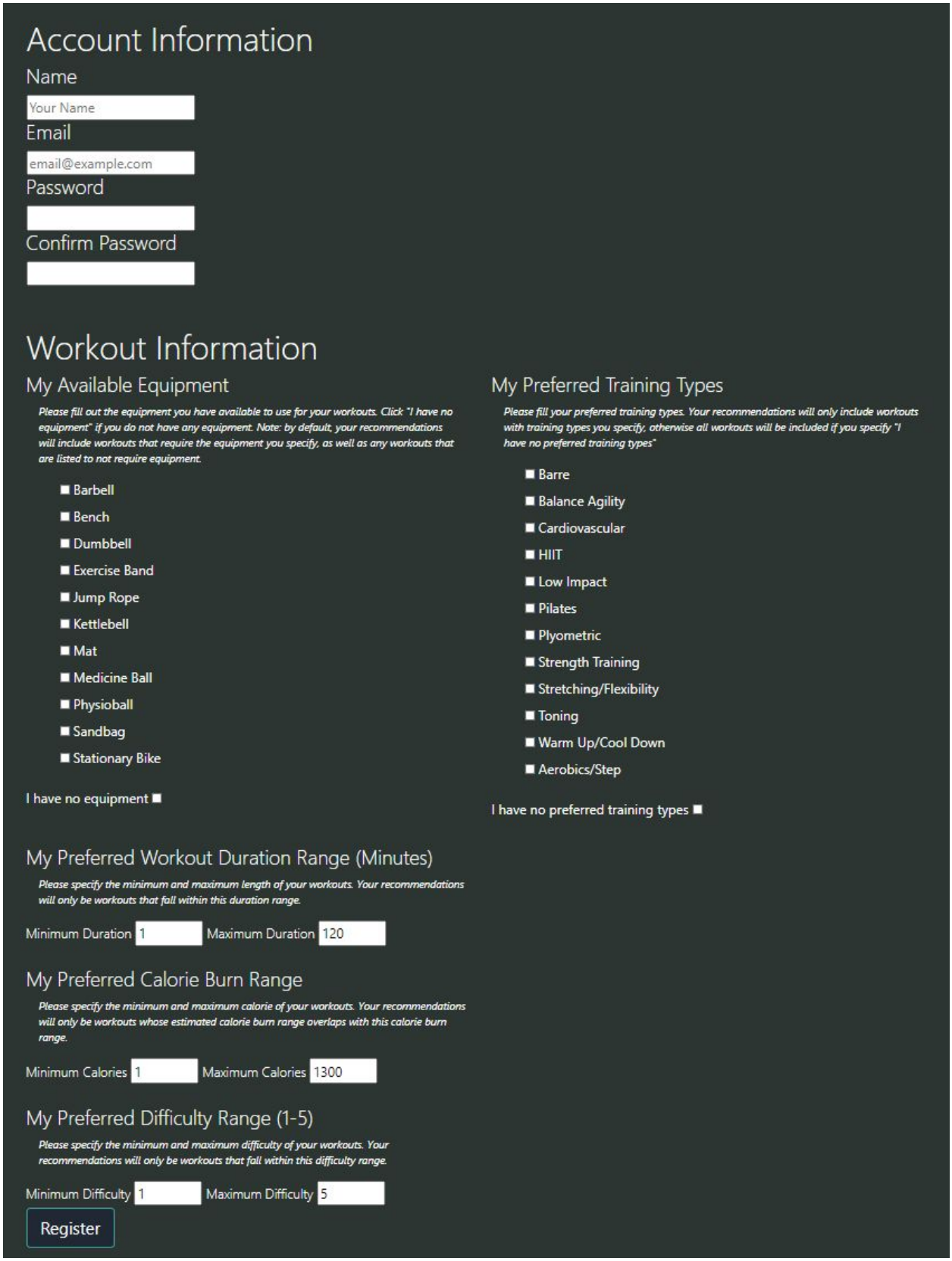
Table 4: *fbworkouts*
The schema matches Table 1 in Appendix B as *fbworkouts_clean.csv* file was directly uploaded into the database. There are no further changes to this table.

Table 5: *fbworkouts_meta*
The schema matches Table 2 in Appendix B as *fbworkouts_meta.csv* file was directly uploaded into the database. There are no further changes to this table.

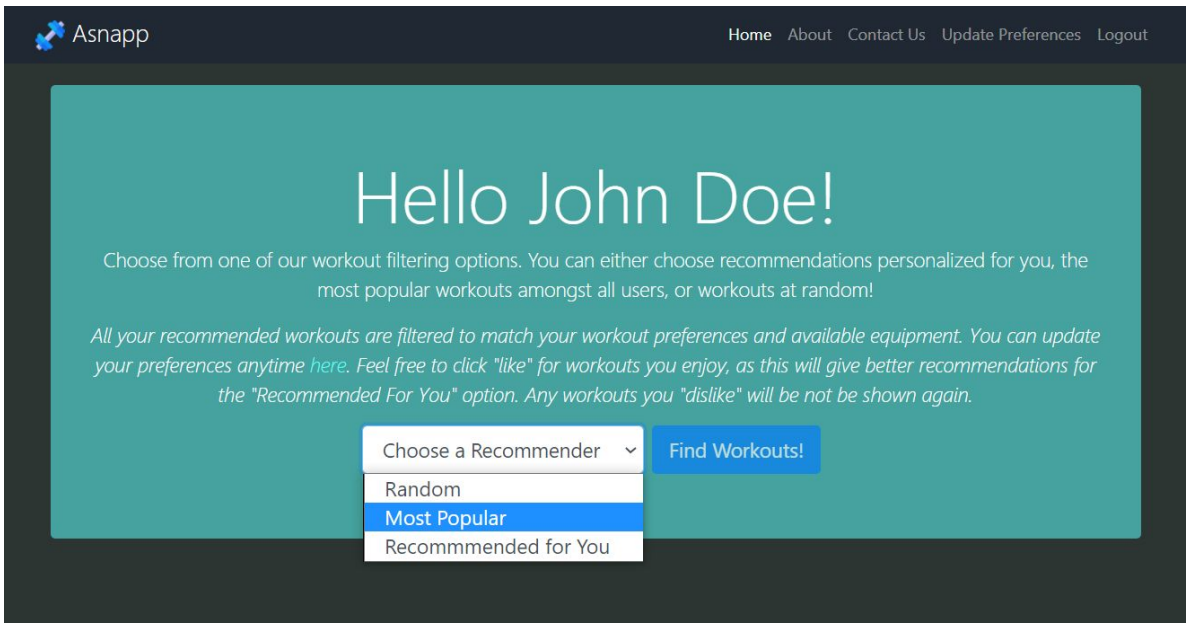
Appendix D: Web Application



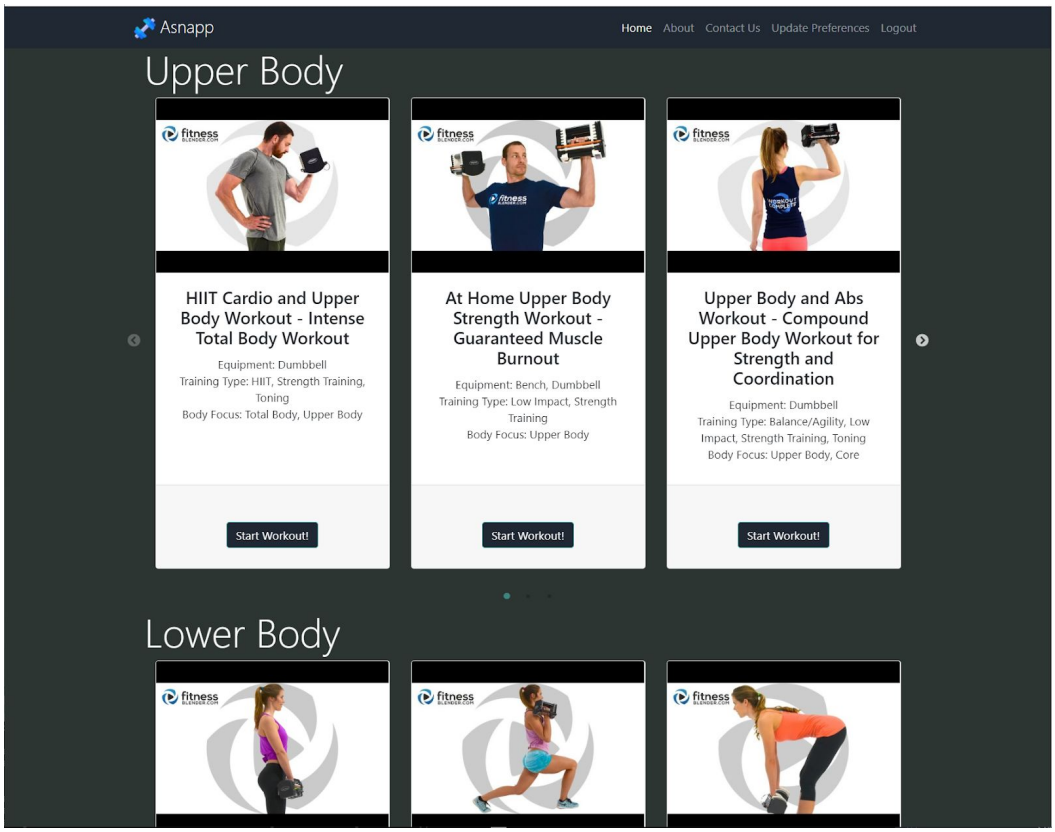
Screenshot 1: *Welcome/Login Screen*



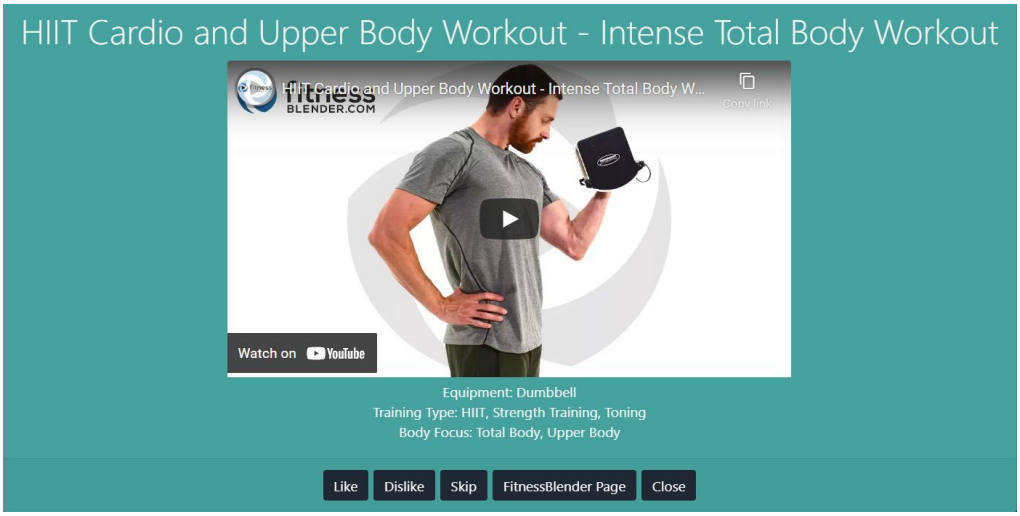
Screenshot 2: *User Registration Page*



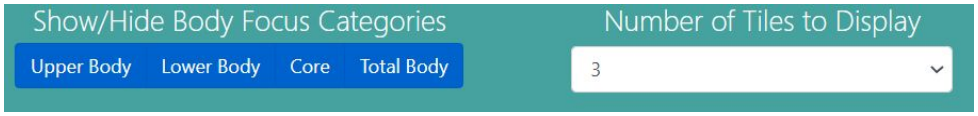
Screenshot 3: *Choosing a Recommender*



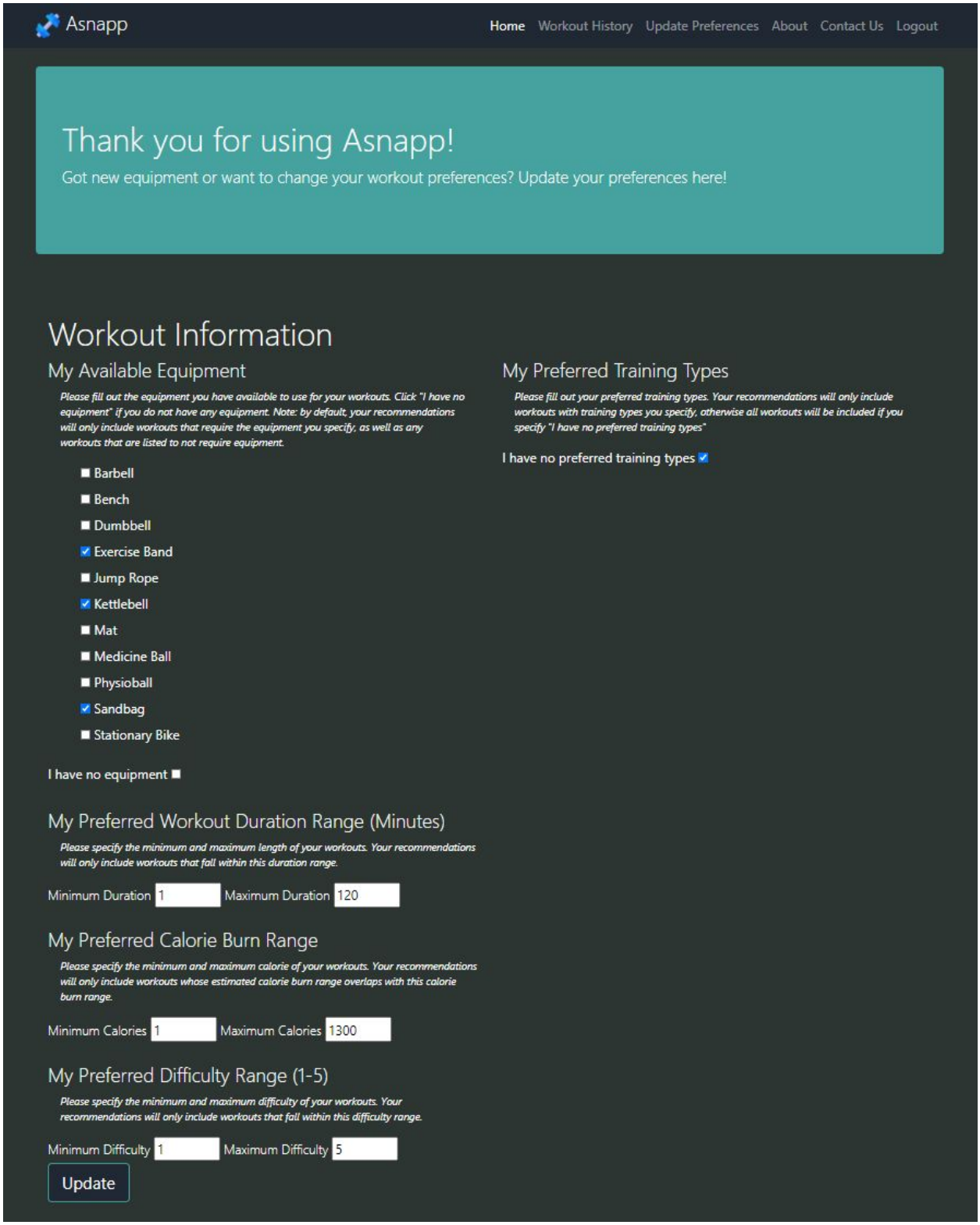
Screenshot 4: *Recommendation Page*



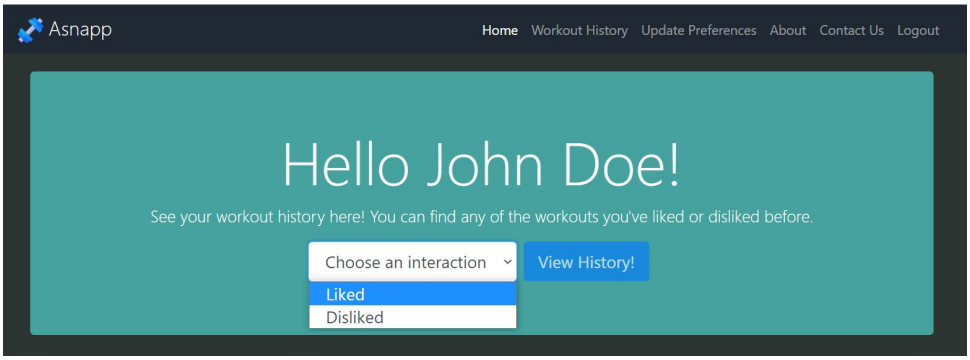
Screenshot 5: *Workout Popup*



Screenshot 6: *Options to Change Display of Workouts*



Screenshot 7: *Updating User Info*



Screenshot 7: *Liked/Disliked Workout History*