Music Recommendation System

Haim Michalashvili
Dorit Lyakhovitsky

82,000,000

songs available according to Spotify.

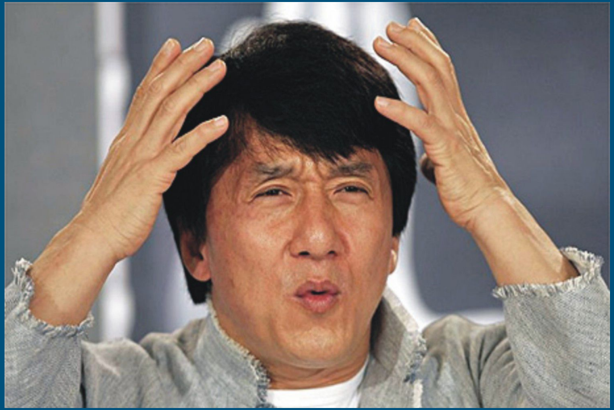
Quora

120,000

new tracks are released every day on music streaming platforms.

Luminate

Due to overwhelming amount of content available, users struggle to discover new music



Our Solution

Build a Music Recommendation system that will suggest new songs to the user based on similarity to his favorite tracks.

Literature Review

The first step in data collection - User Modelling [1] :

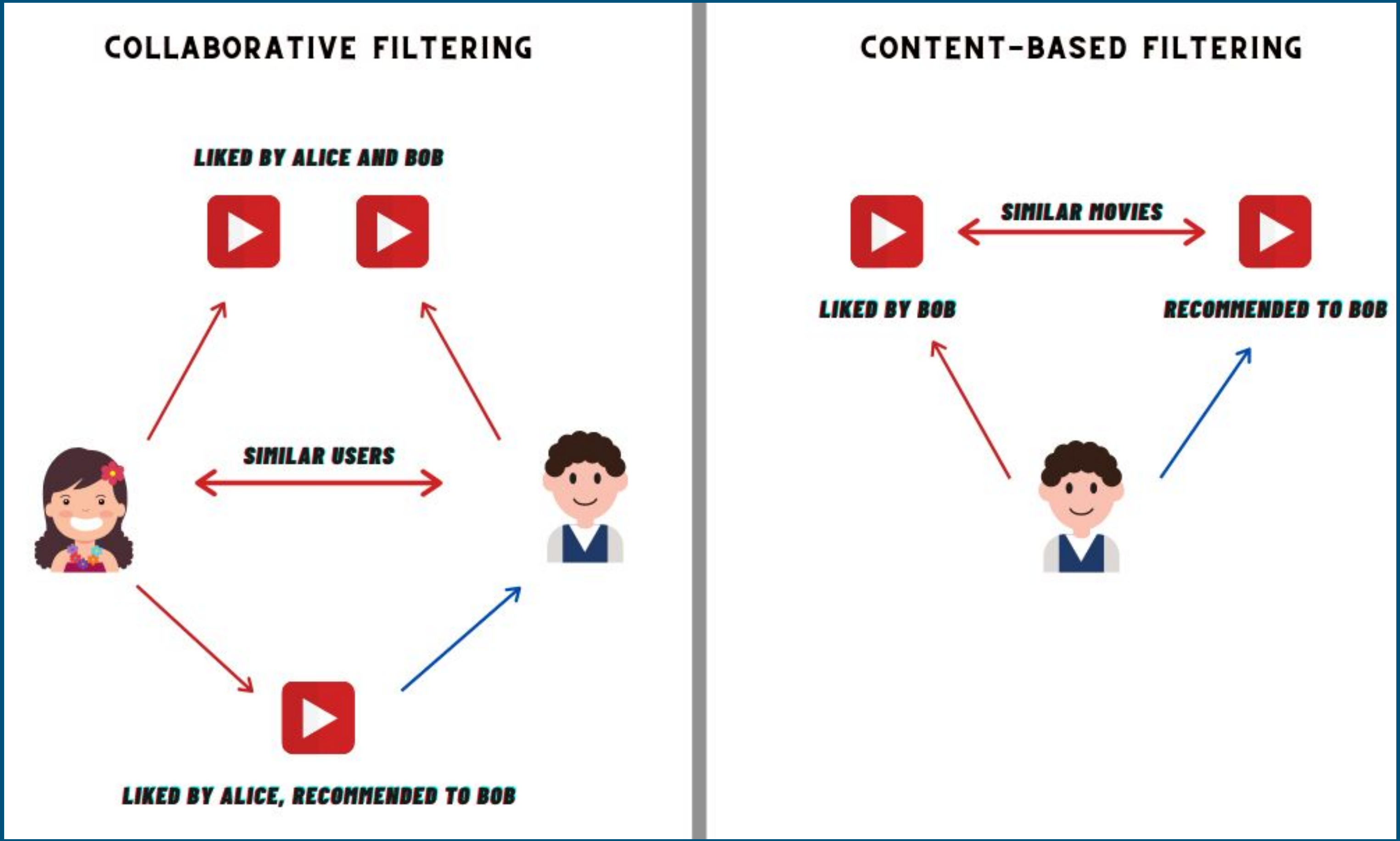
- *Stable properties:* age, gender, location, interests, lifestyle...
- *Fluid components:* mood, attitude, and opinions...

Second step - Item Profiling [1][4] :

- *Editorial metadata*: cover name, composer, title, genre...
- *Acoustic metadata*: beat, tempo, pitch...
  - obtained by analyzing the audio signal
- *Cultural metadata* [2]: comments, reviews, tags, friendships...
  - obtained by mining social networks like Facebook, Twitter...

Two Analysis Algorithms have been found to perform well: *Collaborative Filtering* (CF) and *Content-Based Model* (CBM) [3].

- CBM - the system recommends items that are similar to the ones the user has enjoyed in the past.
- CF - the system recommends items that people with similar tastes and preferences have enjoyed in the past.



https://towardsdatascience.com/user-user-collaborative-filtering-for-jokes-recommendation-b6b1e4ec8642

By representing tracks as multi-dimensional vectors:

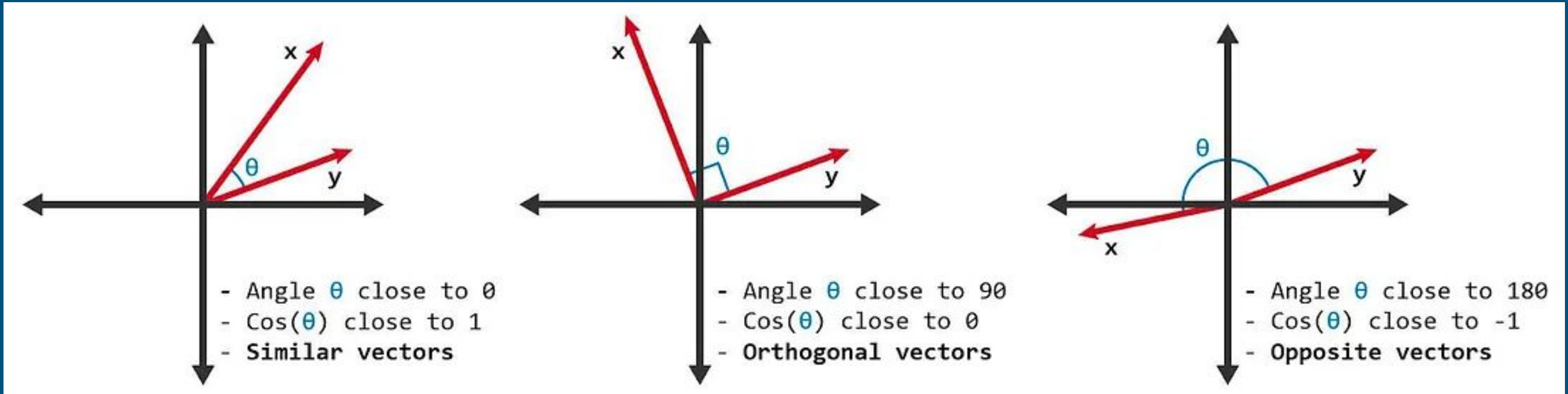
	Artist	Title	acousticness	danceability	energy	tempo	instrumentalness	loudness	valence	speechiness	liveness
0	Richy Mitch & The Coal Miners	Evergreen	0.5570	0.555	0.216	0.058081	0.004160	0.218267	0.504	0.0721	0.1090
1	Angie McMahon	Letting Go	0.4430	0.485	0.753	0.777831	0.009120	0.727726	0.601	0.0548	0.1210
2	Brenn!	Valapriso	0.2080	0.616	0.526	0.318915	0.000000	0.584113	0.415	0.0370	0.3010
3	Yoke Lore	Beige	0.3930	0.470	0.670	0.095654	0.041000	0.638832	0.219	0.0783	0.1170
4	Noah Kahan	You're Gonna Go Far	0.5990	0.590	0.360	0.776771	0.000000	0.423515	0.379	0.0301	0.1120
...	...	...	...	...	...	...	...	...	...	...	...
95	JOSEPH	Green Eyes	0.8450	0.564	0.467	0.290436	0.000000	0.896867	0.411	0.0389	0.7070
96	The Franklin Electric	Borderline	0.6350	0.650	0.684	0.084913	0.000122	0.468979	0.850	0.0313	0.1090
97	Dawes	Crack The Case	0.6290	0.531	0.476	0.462724	0.002570	0.252136	0.239	0.0345	0.0801
98	Trousdale	Lov									
99	Henry Jamison, Ed Droste	Green Room (feat. E Drost									

	Title	Artist	genre	genre_folk	genre_rock	genre_indie	genre_pop	genre_songwriter
0	Evergreen	Richy Mitch & The Coal Miners	modern folk rock	1	1	0	0	0
1	Letting Go	Angie McMahon	australian indie	0	0	1	0	0
2	Valapriso	Brenn!	singer-songwriter pop	0	0	0	1	1
3	Beige	Yoke Lore	nyc pop	0	0	0	1	0
4	You're Gonna Go Far	Noah Kahan	pov: indie	0	0	1	0	0
...	...	...	...	...	...	...	...	...
95	Green Eyes	JOSEPH	folk-pop	1	0	0	1	0
96	Borderline	The Franklin Electric	canadian indie folk, indie quebecois	1	0	1	0	0
97	Crack The Case	Dawes	indie folk, modern folk rock, new americana, s...	1	1	1	0	0
98	Love	Trousdale	modern indie folk	1	0	1	0	0
99	Green Room (feat. Ed Droste)	Henry Jamison, Ed Droste	indie folk, pop folk, vermont indie	1	0	1	1	0

Source

We can calculate the distance between them, aka the similarity:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$



Source

[1] A Survey of Music Recommendation Systems and Future Perspectives
Song, Yading, Simon Dixon, and Marcus Pearce. "A survey of music recommendation systems and future perspectives." *9th international symposium on computer music modeling and retrieval*. Vol. 4. 2012.

[2] Music recommendation by unified hypergraph: combining social media information and music content
Bu, Jiajun, et al. "Music recommendation by unified hypergraph: combining social media information and music content." *Proceedings of the 18th ACM international conference on Multimedia*. 2010.

[3] Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions
Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." *IEEE transactions on knowledge and data engineering* 17.6 (2015): 734-749.

[4] Client-Based Generation Of Music Playlists via Clustering Of Music Similarity Vectors
Renshaw, Erin, and John Platt. "Client-based generation of music playlists via clustering of music similarity vectors." *U.S. Patent No. 7,571,183*. 4 Aug. 2009.

[5] Data-Driven Music Exploration: Building a Spotify Song Recommender
Chang. "Data-Driven Music Exploration: Building a Spotify Song Recommender". *Medium Website*. 2023.

Competitors Analysis

We decided to compare the following competitors' Algorithms:

	Spotify		Youtube Music		SoundCloud		Our Product	
User Interface	User-friendly and seamless experience	5	Friendly experience	4	A little complicated	3	Minimal and friendly UI	3
Platforms	Computer, Mobile	5	Computer, Mobile	4	Computer, Mobile	4	Computer	3
Social Features[1]	Advanced features	5	Some features	2	Multiple features	4	None	1
Analysis sAlgorith m	CF, CBM, NLP	5	CBM, UM, Feedback	4	CBM, UM, Promotion	3	CF, CBM, Hybrid	4
Total users	574M+	5	100M+	5	76M+	5	~100	1
Total Tracks	100M+	5	100M+	5	375M+	5	1M	1
Output	Personalised themed playlists	5	Recommended playlists	4	Suggested tracks	2	Personalised themed playlists	5
Total Score		35		28		26		18

Shortcuts used:

CF Collaborative Filtering
CBM Content-Based Model
NLP Natural Language Processing
UM User Modelling (interests etc.)

Our analysis algorithms will be processing a data sample as part of our Proof of Concept music recommender project.

It's worth noting that this is only a preliminary application, and it doesn't include the full range of features that a real-life music recommendation system would have, such as more users, songs, playlists, social features, and a seamless user interface.

[1] Social features: such as seeing what friends are listening to, creating and sharing playlists, joining jamming sessions, and integrating with social media platforms.

Goals, Objectives, Metrics

Goals

The purpose of the project is to provide a solution to music lovers who are looking for novelty music. The system will know how to characterize the songs and the users and learn their preferences. Using the algorithms chosen, the system will suggest new songs to the user. The system will identify trends in the application's usability and will improve its recommendations to the user each time.

Objectives

- Establishing a characterization logic of tracks and users.
- Development of a recommendation algorithm based on similarity between songs and similarity between users.
- Establishing a website with a simple interface for the purpose of receiving data and recommending songs.
- Interactive presentation of the algorithm's recommendations that are updated according to the user's choices.

Metrics

- There are more *positive* than *negative* interactions between the user and the system.

  Positive interactions: clicking songs and marking "like/dislike"...

  Negative interactions: refreshing the recommendation list without marking anything, abandoning...
- At least 80% of the users will be recurring users of the system.

Requirements

Functional - Algorithm Requirements

- The algorithm will vectorize users and tracks by to their features.
- The algorithm will find songs similar to user’s liked tracks.
- The algorithm will find songs liked by similar users.
- The algorithm will improve the recommendations according to the user’s interaction with the system.
- The algorithm will identify user preferences and create Concept Playlists based on the characteristics: artists, genre...

Functional - Interface Requirements

- The interface will log in a registered user by username & password.
- The interface will allow to import tracks from *csv* file.
- The interface will display an interactive lists of recommendations, with the ability to mark "like/dislike" and "refresh list".
- The interface will display recommended Concept playlists.

Functional - System Requirements

- There will be communication between the server (containing the algorithm), the database and the website interface.
- The characterization of the objects (tracks and users) will be performed on the server and saved in the database.
- The system will import the recommendation lists generated by the algorithm from the database.

Non Functional - Design Requirements

- A clear and convenient interface with a minimal design.

Non Functional - Operating Requirements

- The system will run a web interface.

Non Functional - Performance Requirements

- The system will retrieve recommendation playlists in a minimal time (to be defined later).
- The system will be able to work with several users at the same time.

Non Functional - Reliability Requirements

- *Persistent storage* - the system will store all the data in the database in a way that allows the system to be used without losing information. A user will be able to connect to the system and continue from the same point where he stopped.

Detailed Design

Tracks Dataset

The data set we will be using is 1 Million Spotify tracks.

artist_name	string	The artist's name
track_name	string	The track name
track_id	string	Spotify's unique track id
popularity	int64	Track popularity (0 to 100)
year	int64	Year released (2000 to 2023)
genre	string	The genre of the track
danceability	float64	Track suitability for dancing (0.0 to 1.0)
energy	float64	The perceptual measure of intensity and activity (0.0 to 1.0)
key	int64	The key the record is in (-1 to -11)
loudness	float64	Overall loudness of track in decibels (-60 to 0 dB)
mode	int64	Modality of the track (Major '1'/ Minor '0')
speechiness	float64	Presence of spoken words in the track
acousticness	float64	Confidence measure from 0 to 1 of whether the track is acoustic
instrumentalness	float64	Whether track contains vocals (0.0 to 1.0)
liveness	float64	Presence of audience in the recording (0.0 to 1.0)
valence	float64	Musical positiveness (0.0 to 1.0)
tempo	float64	Tempo of track in beats per minutes (BPM)
duration_ms	int64	Duration of track in milliseconds
time_signature	int64	Estimated time signature (3 to 7)

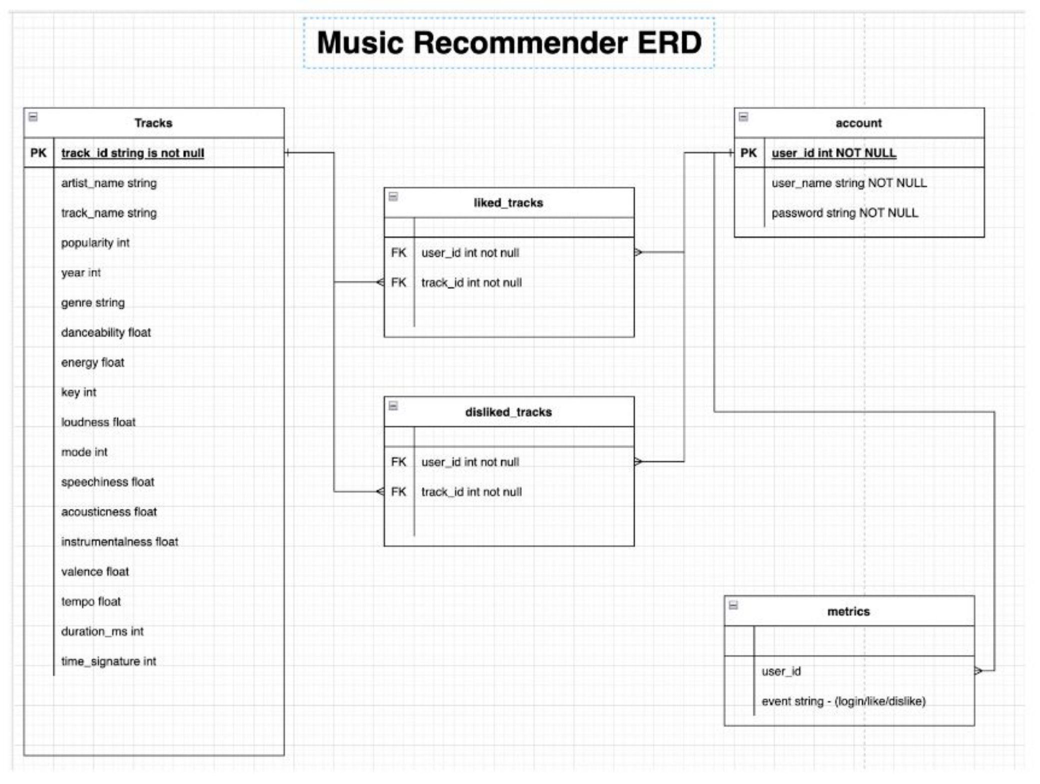
The dataset must be further cleaned:

Hot-Encoding for 'genre' column,

StandardScaler for numeric columns,

Bucketing for 'year' column...

Database ERD



Algorithm

1. Get the tracks the user has liked (=**user_like_df**).
2. Get the tracks the user has disliked (=**user_dislike_df**).
3. Get all tracks table (=**all_tracks_df**).
4. Sort **user_like_df**, **all_tracks_df** columns in alphabetical order so columns match each other.
5. Drop columns: *track_id, track_name, artist_name*.
6. Calculate weighted mean values [1] of **user_like_df** (=**user_like_df_mean**).
7. Calculate similarity_column [3] = sklearn.cosine_similarity(**all_tracks_df**, **user_like_df_mean**).
8. Join similarity_column with **all_tracks_df**.
9. Sort **all_tracks_df** by descending *similarity_column*.
10. Filter **all_tracks_df** of tracks that appear in **user_like_df** & **user_dislike_df**.
11. Filter **all_tracks_df** where *similarity_column* > **THRESHOLD** [2].
12. Present tracks to user.

Algorithm Notes

[1] **weighted mean values** - we want to calculate the mean values from the user's **entire** history of 'likes', but in a weighted way so that the oldests tracks have less influence on the upcoming recommendations. The table below is an example of how we could calculate the weighted mean for all columns.

The specifics (percentiles and weights) are to be determined during development.

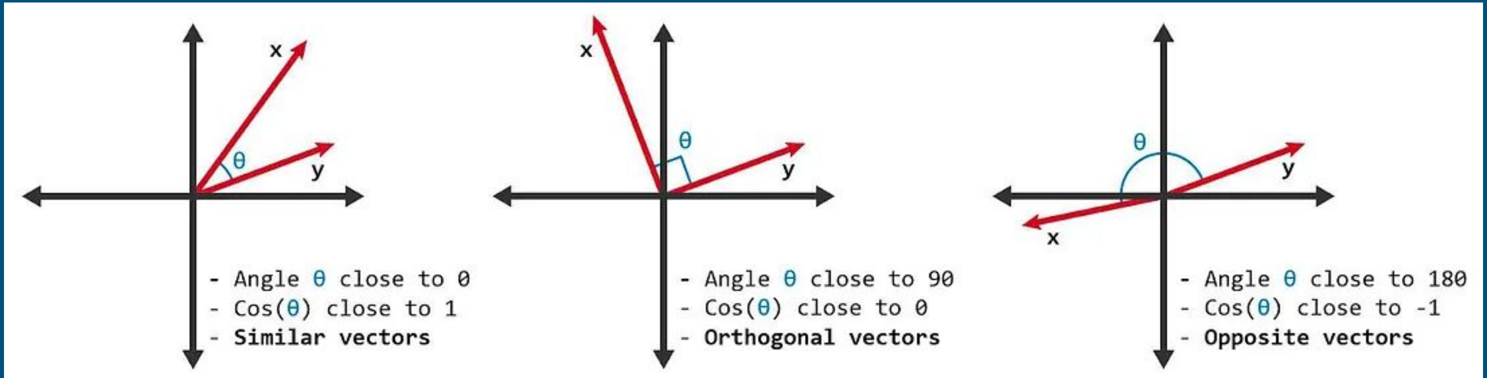
Oldest tracks	**mean of one column**	**weight**
0%-20%	54	1
20%-40%	23	2
40%-60%	87	3
60%-80%	79	4
80%-100%	97	5
Newest tracks		
Regular mean	68.00	
Weighed mean	77.47	

Algorithm Notes

[2] **THRESHOLD** - Degree of similarity(-1 to 1), to be decided later.

[3] **sklearn.cosine_similarity** - The cosine measurement is commonly used to measure the distance between vectors. Below is the formula, and illustrated examples:

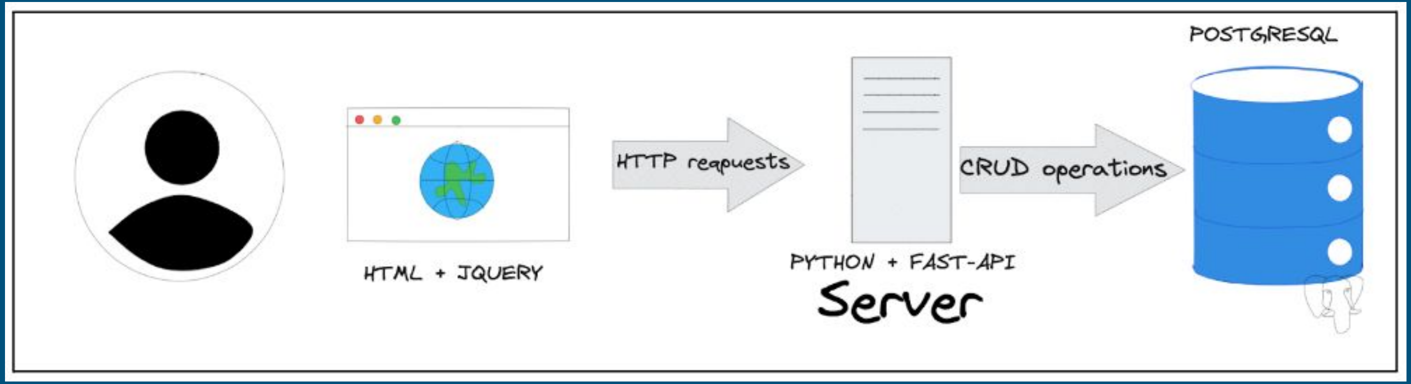
$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

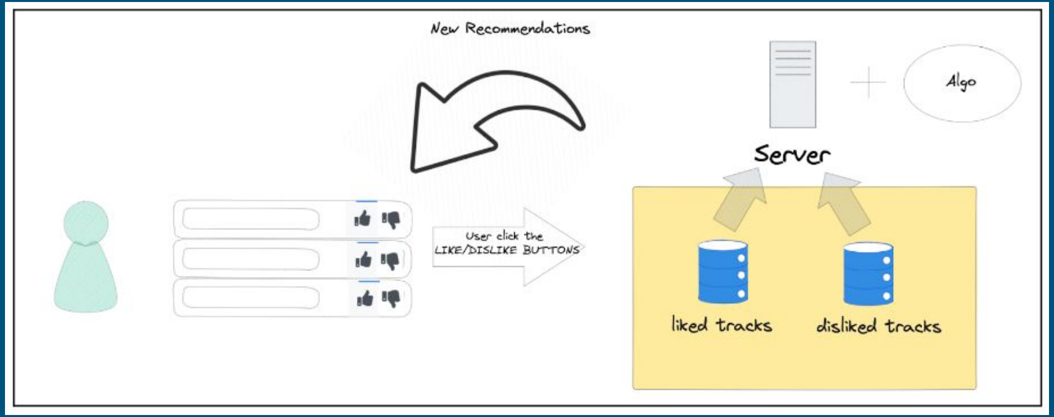


Modules

- ➔ **User Registration Module** - Registers and logs the user into the system.
- ➔ **Music Recommender Module** - Generates a recommendation track list for the user, and manages the liked & disliked tracks against the database.
- ➔ **Metrics Module** - Manages the metrics system

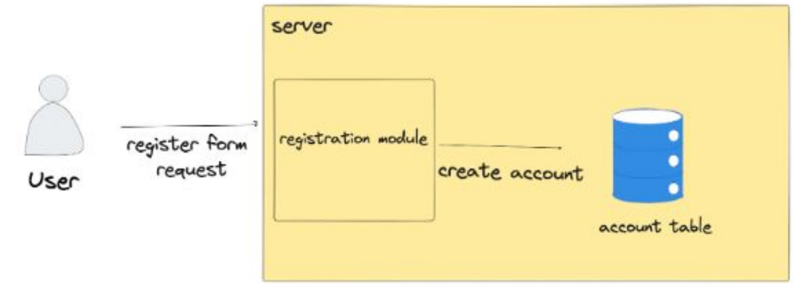
Architecture



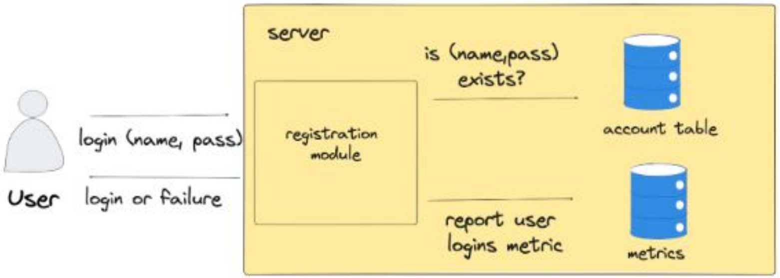


UMLs of Use Cases

User request to register

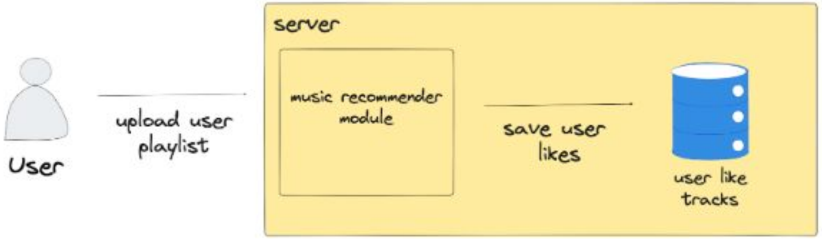


User request to login using (username, pass):

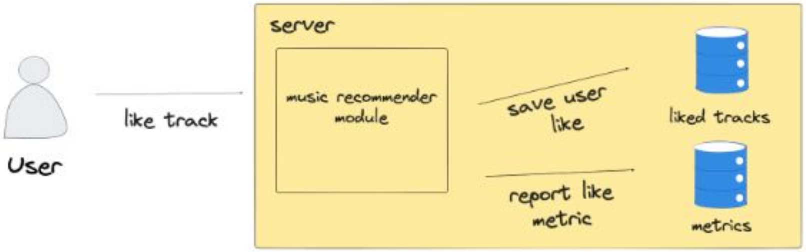


User uploads his favored playlist:

In the format of a CSV, containing a list of Spotify track IDs of his favored tracks.

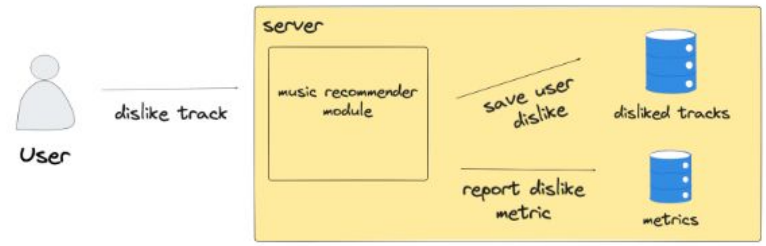


User clicked the like button on a track from the list:

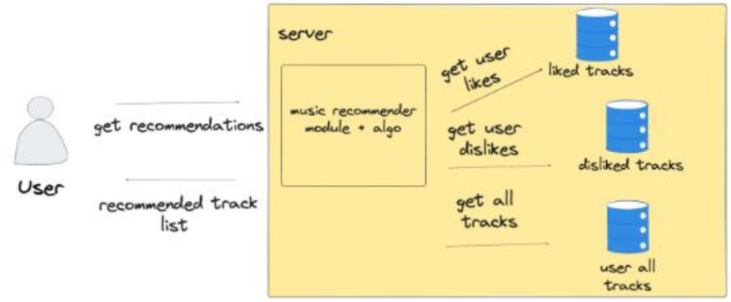


UMLs of Use Cases

User clicked the like button on a track from the list:



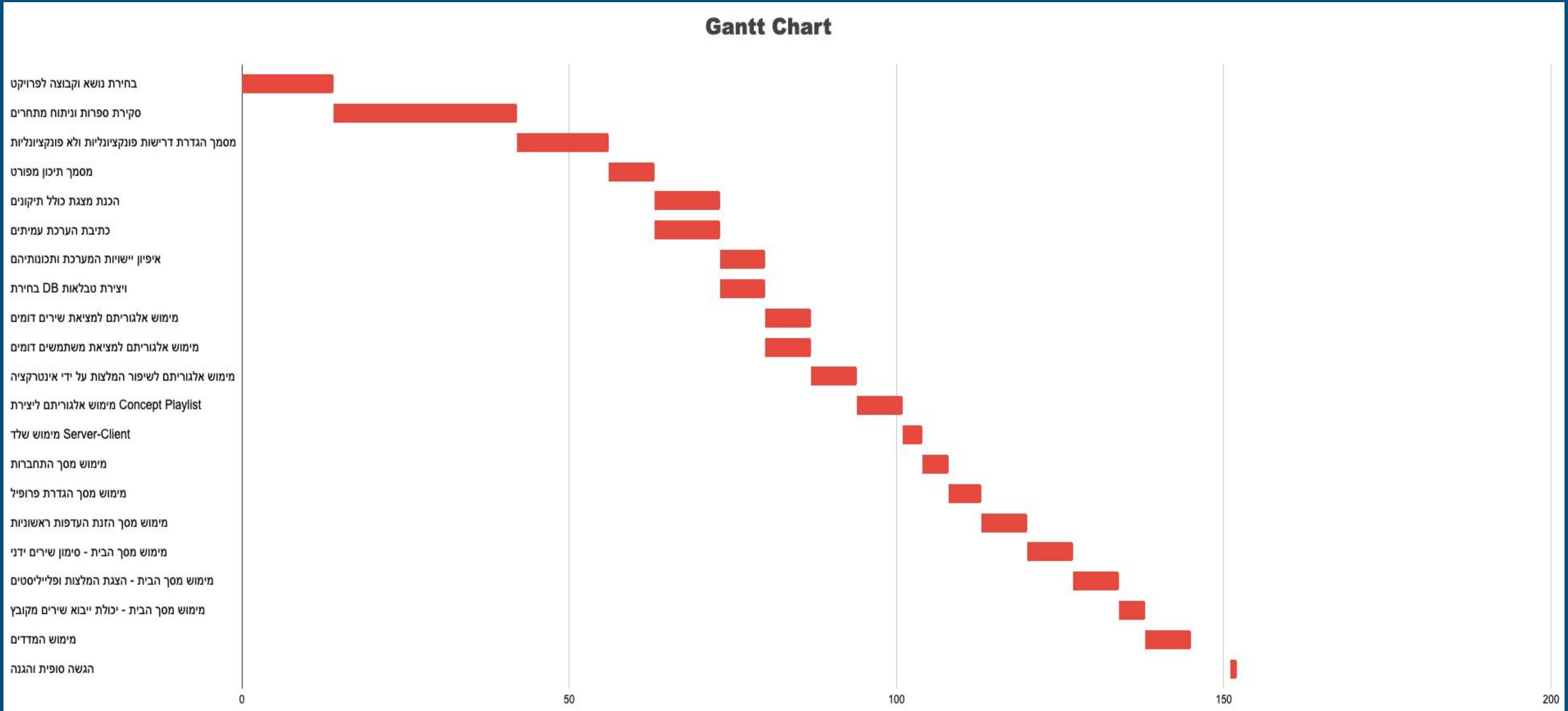
User clicked the ‘Refresh Recommendation list” button



Technologies

- ➔ Frontend - JQuery (Javascript)
- ➔ Backend - FastAPI (Python)
- ➔ Algorithm - libraries: numpy, pandas, sklearn (Python)
- ➔ Database - PostgreSQL

Gantt



Gantt Chart
בחירת נושא וקבוצה לפרויקט
סקירת ספרות וניתוח מתחרים
מסמך הגדרת דרישות פונקציונליות ולא פונקציונליות
מסמך תיכון מפורט
הכנת מצגת כולל תיקונים
כתיבת הערכת עמיתים
איפיון יישויות המערכת ותכונותיהם
ויצירת טבלאות DB בחירת
מימוש אלגוריתם למציאת שירים דומים
מימוש אלגוריתם למציאת משתמשים דומים
מימוש אלגוריתם לשיפור המלצות על ידי אינטרקציה
Concept Playlist מימוש אלגוריתם ליצירת
Server-Client מימוש שלד
מימוש מסך התחברות
מימוש מסך הגדרת פרופיל
מימוש מסך הזנת העדפות ראשוניות
מימוש מסך הבית - סימון שירים ידני
מימוש מסך הבית - הצגת המלצות ופלייליסטים
מימוש מסך הבית - יכולת ייבוא שירים מקובץ
מימוש המדדים
הגשה סופית והגנה
0
50
100
150
200

The END

Thanks for listening