Tracks Dataset

The data set we will be using is 1 Million Spotify tracks. The tracks dataset includes the following columns:

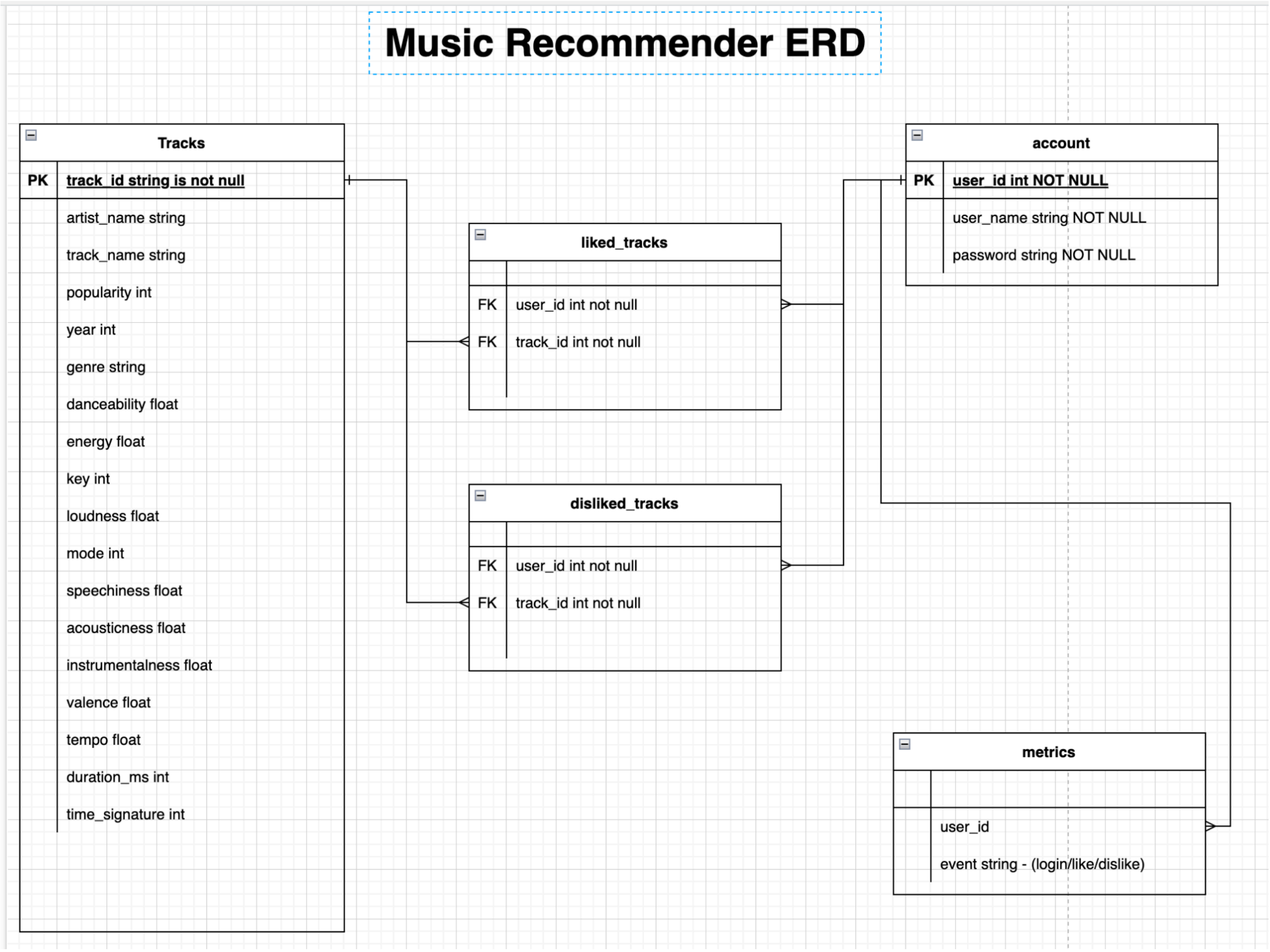
artist_name	string	The artist's name
track_name	string	The track name
track_id	string	Spotify's unique track id
popularity	int	Track popularity (0 to 100)
year	int	Year released (2000 to 2023)
genre	string	The genre of the track
danceability	float	Track suitability for dancing (0.0 to 1.0)
energy	float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

It's important to notice: the dataset must be further cleaned: Hot-Encoding for 'genre' column, StandardScaler for numeric columns, Bucketing for 'year' column.

Database ERD

Entities:

- Tracks
- Liked/Dislikes Tracks
- Metrics
- Account



Music Recommender ERD

Tracks	
PK	**track_id string is not null**
	artist_name string
	track_name string
	popularity int
	year int
	genre string
	danceability float
	energy float
	key int
	loudness float
	mode int
	speechiness float
	acousticness float
	instrumentalness float
	valence float
	tempo float
	duration_ms int
	time_signature int

liked_tracks	
FK	user_id int not null
FK	track_id int not null

disliked_tracks	
FK	user_id int not null
FK	track_id int not null

account	
PK	**user_id int NOT NULL**
	user_name string NOT NULL
	password string NOT NULL

metrics	
	user_id
	event string - (login/like/dislike)

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.

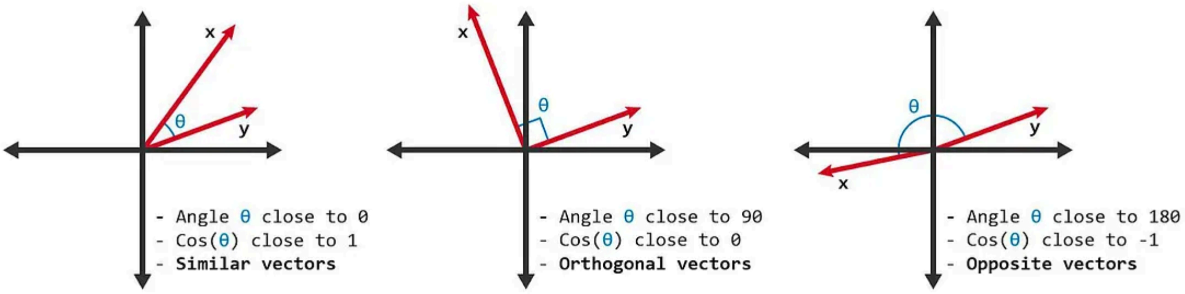
[1] weighted mean values - we want to calculate the mean values from the user's entire history of 'likes', but in a weighted way so that the oldests tracks have less influence on the upcoming recommendations. The table below is an example of how we could calculate the weighted mean for all columns. The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

[2] THRESHOLD - Degree of similarity, 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:

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

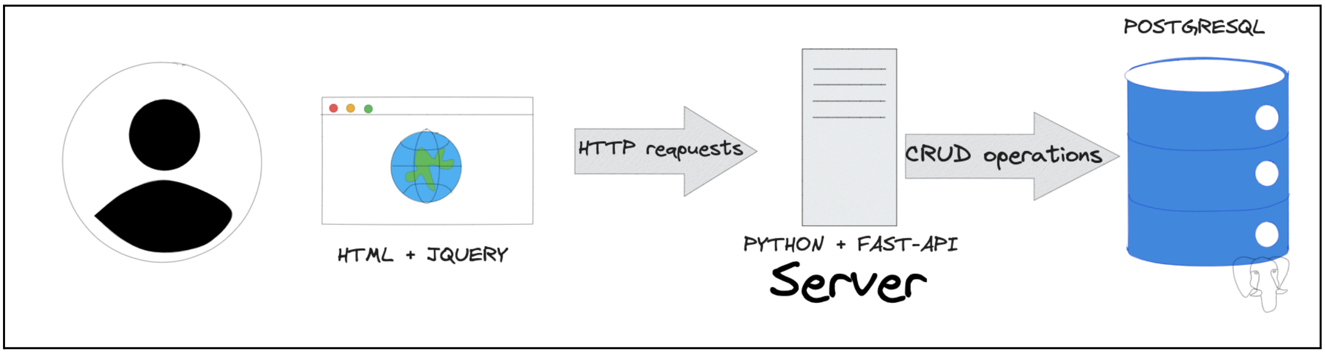


Modules

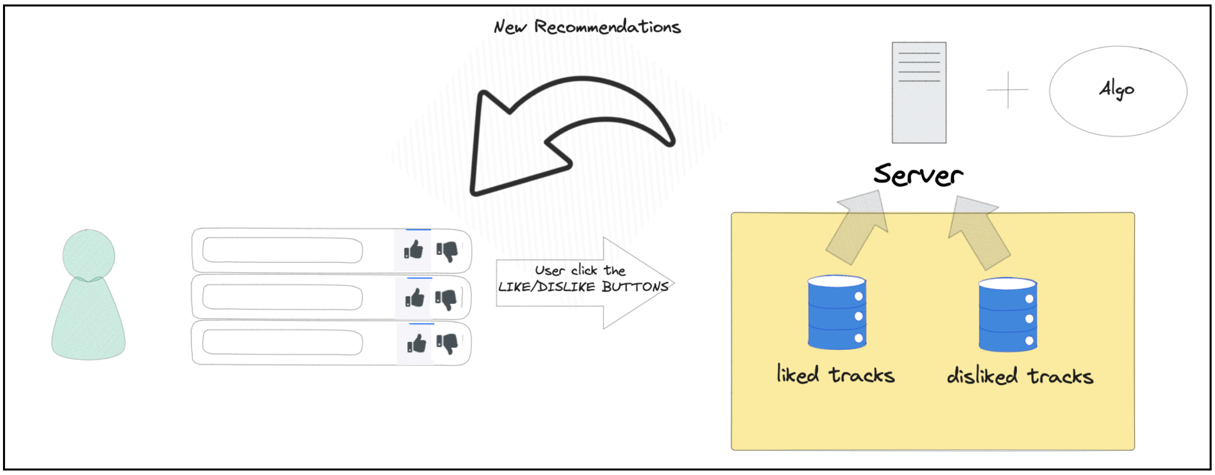
- User Registration Module - Registers and logs the user into the system.
- Music Recommender Module - Generates a recommendation track list for the user, and manages the liked & disliked tracks against the database.
- Metrics Module - Manages the metrics system

Architecture

Web System Architecture:

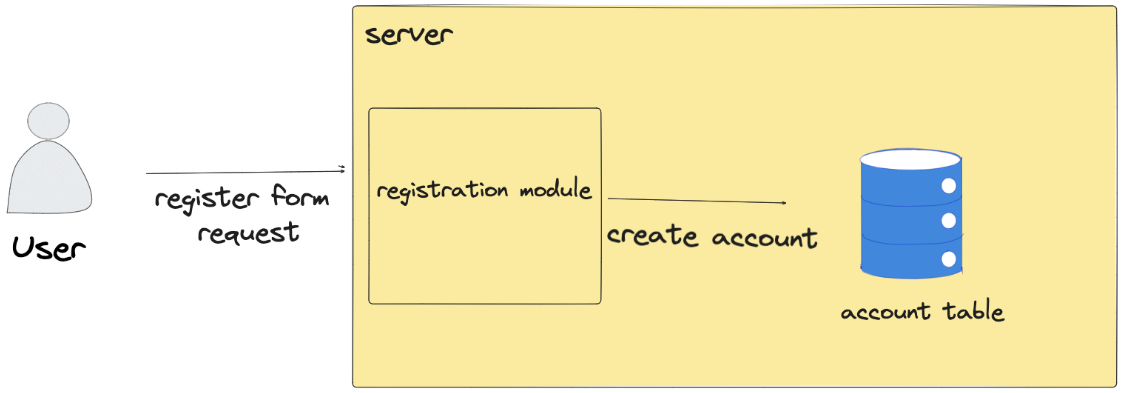


Recommendation Architecture:

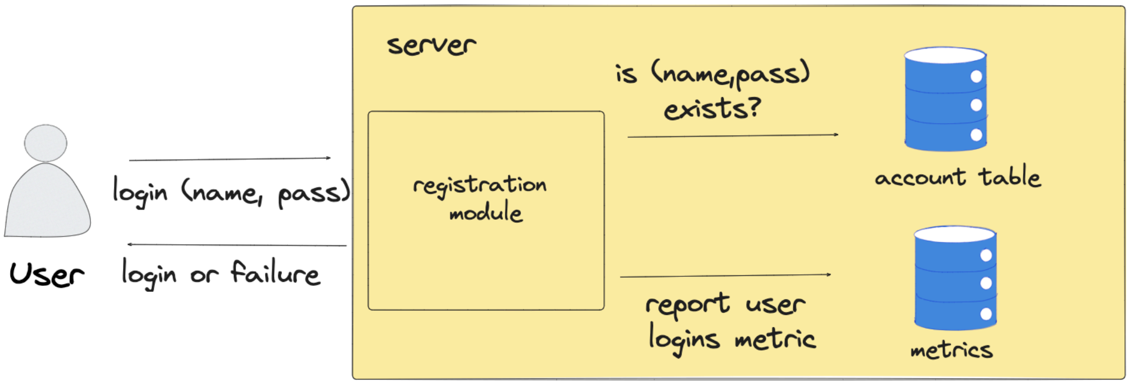


UML Graphs of use cases

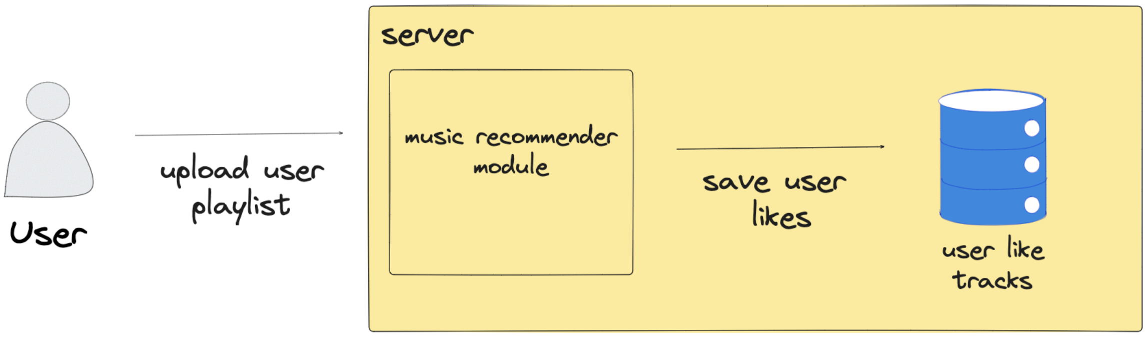
User request to register



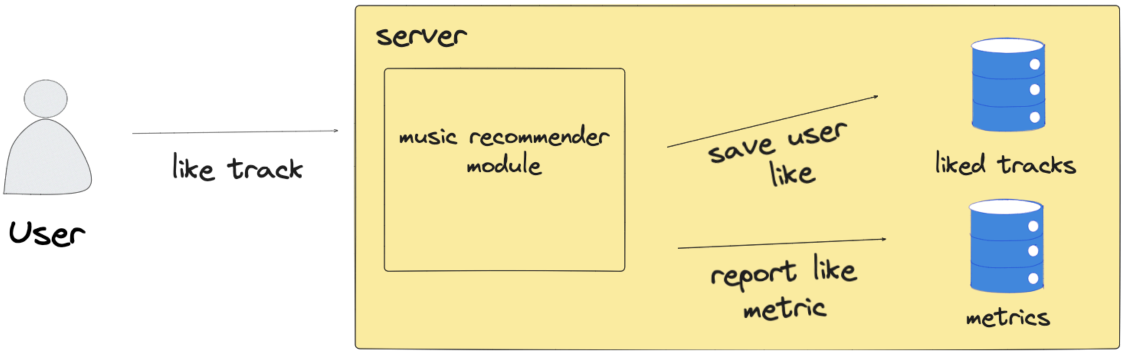
User request to login using (username, pass):



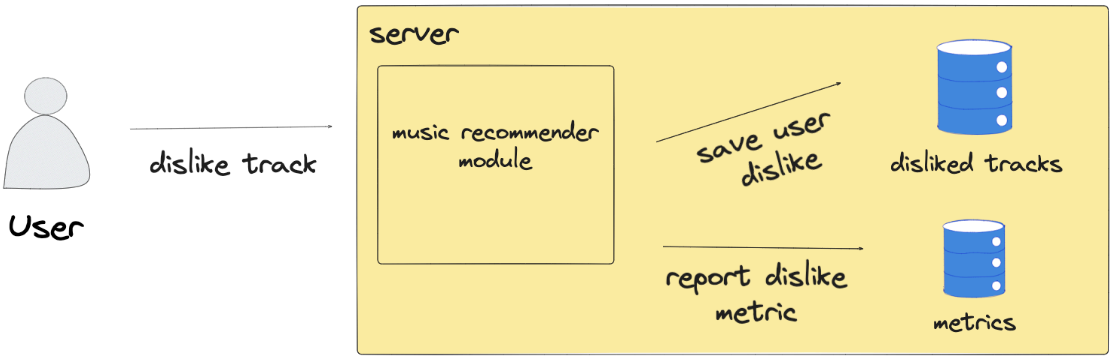
User uploads his favored playlist:
In the format of a CSV, containing a list of Spotify track IDs of his favored tracks.



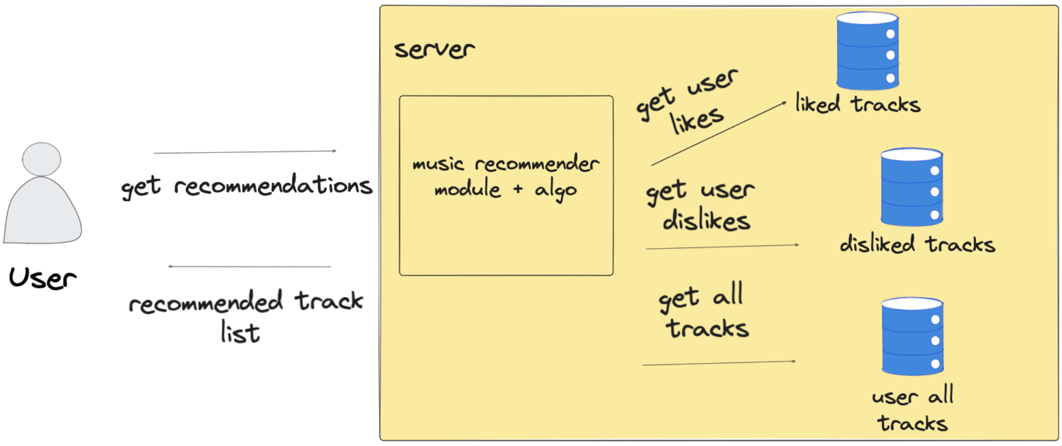
User clicked the like button on a track from the list:



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