Team Name: C#
Application Name: Harmony

Harmony is a music-based social networking app with the motive to connect people based on their taste in music.

Find a song or a trending artist, discover music and bond with new people over it.
Discuss your favorite artist with thousands of others.
Can't recall that snippet you heard? Maybe your friends do!

Find out with them together.

S.no.	Team Member	Roll No.	Expected Marks (Out of 100)
1.	Advika Singh	2018275	100
2.	Devender Singh	2018334	100
3.	Nikhil Krishnan	2018248	100
4.	Kunal Verma	2018241	100
5.	Pratham Gupta	2018072	100

About the Software:

The software allows people to connect with each other on the basis of music preferences. It also lets you search for artists and get to know their upcoming songs and albums. It also gives you song recommendations based on your preferences.

Bonus Feature:

The bonus feature in our software is to allow upcoming musicians to register themselves and their new songs so as to let people around the world discover them easily. It will allow simple people with no special privileges to showcase their talent to the world.

Future Scope :

The future scope of the app is to build an inbuilt chatting app which allows the users to chat with the person they have been matched with. The users can choose whether they want to be anonymous during the chat or they want to be seen.
This chat feature will also heavily add value to our bonus as now labels can directly scout fresh talent.

Stakeholders:

1. Record Labels -
    - They need to search for musicians to sign based on their number of songs, active listeners, minutes streamed, popularity, genre, and fanbase.
    - Check the artist's previous record labels to compare technicalities.
    - Search other record labels and their artists, to keep a check on the competition.
    - Search unsigned musicians to scout and sign new talents.

2. Established musicians -
    - How many people like their songs and connect over it.
    - Record labels looking for musicians.
    - The number of people following them.
    - Search upcoming artists to collaborate with/support them.

3. Customers looking for different genres of music -
    - Find different music by genre, times played, song name, musician name, producer name.
    - Find established musicians by name, song name, a record label they are signed with.
    - Find upcoming musicians.
    - Know their friend's activity.
    - Browse record labels and musicians signed by them.

4. Upcoming artists-
    - They need to search record labels by name of artists, their signed musicians by name, fanbase.
    - Check if they are in the most played songs playlist.
    - Browse their songs by name, time played, genre, etc
    - Browse other artists by name, songs, genre to collaborate with them.

5. Consumers
    - Looking to connect with people who have the same taste in music.
    - Connect with people by genre, favorite artists, common songs, favorite songs, etc.
    - To check how many people have been connected successfully.

Questions for stakeholders:-

Record labels

How to find upcoming artists?
How to find artists to sign based on their fanbase?
How to search for an artist's record history?
How to compare artists to find the best one?
How to find other competitors and see their progress?

Established musicians

How many people have searched for their songs?
How many people have connected over their songs?
How many people have started following them?
How to find established record labels?
How to find its competitor's progress?
How to look for upcoming artists?

Customers

How to find more information on a song?
How to look for more artists based on a genre?
How to look for upcoming artists?
How to find new music?
How to connect with people based on similar tastes in music.
How to connect with people who listen to songs by the same artist.
To find out how many people have connected over a song
To find out how many people you have been connected with.
To find out how many are listening to the same song.

Upcoming Artists

How to look for new record labels?
How to look for more information on record labels?
How to find out how many people searched for their songs?
How to look for other artists?
How to connect with other artists?

Advertisements

To look for most searched songs
How to find out most searched artists
How to connect with artists
To connect with record labels
To gain more popularity

TABLE SCHEMAS

1. **User Table:**

```
create table users(
        user_id         varchar(20)     primary key,
        name            varchar(20)     not null,
        email           varchar(20)     unique,
        passwd          varchar(20)     not null,
        dob             date            not null,
        gender                  char(1)         not null,
        CHECK gender in ('m','f','o'),
        INDEX (name) );
```

2. **Artist Table:**

```
create table artists (
        artist_id       varchar(20)     primary key,
        name            varchar(20)     not null,
        email           varchar(20)     unique,
        passwd          varchar(20)     not null,
        dob             date            not null,
        signed          int             default 0,
        gender                  char(1)         not null,
        followers       int             default 0,
        CHECK gender in ('m','f','o'),
        CHECK signed in ('1', '0'),
        INDEX (name) );
```

3. **Record Label Table:**

```
create table record_labels(
        label_id        varchar(20)     primary key,
        name            varchar(20)     not null,
        email           varchar(20)     unique,
        passwd          varchar(20)     not null,
        no_of_artists   int             default 0,
        INDEX (name) );
```

4. **Song Table:**

```
create table songs(
        song_id         varchar(20),
        name            varchar(20)     not null,
        artist_name     varchar(20)     not null,
        label_name      varchar(20),
        album_name      varchar(20),
        release_date    date            not null,
        views           int             default 0,
        genre           varchar(10),
        language        varchar(10),
        PRIMARY KEY(song_id, artist_name),
        INDEX (name),
        INDEX (artist_name),
        INDEX (album_name) );
```

5. **Album Table:**

```
create table album(
        album_id        varchar(20),
        album_name      varchar(20)     not null,
        artist_id       varchar(20)     unique,
        artist_name     varchar(20)     not null,
        label_name      varchar(20),
        no_of_songs     int             default 0,
        release_date    date            not null,
        PRIMARY KEY(album_id),
        FOREIGN KEY(artist_id) references artists,
        INDEX (album_name),
        INDEX (artist_name) )
```

6. **Favorite Albums Table:**

```
create table fav_albums (
        user_id         varchar(20),
        album_id        varchar(20),
        PRIMARY KEY(user_id, album_id),
        FOREIGN KEY(user_id) references users,
        FOREIGN KEY(album_id) references album );
```

7. **Favorite Artists Table:**

```
create table fav_artist (
        user_id         varchar(20),
        artist_id       varchar(20),
        PRIMARY KEY(user_id, artist_id),
        FOREIGN KEY(user_id) references users,
        FOREIGN KEY(artist_id) references artists );
```

8. **Playlist Table:**

```
create table playlists (
        user_id         varchar(20),
        list_id         varchar(20),
        list_name       varchar(20),
        song_id         varchar(20),
        PRIMARY KEY(user_id, list_id, song_id),
        FOREIGN KEY(user_id) references users );
```

9. **Connections Table:**

```
create table connections (
        u_id1           varchar(20),
        u_id2           varchar(20),
        co_artists      varchar(20),
        PRIMARY KEY(u_id1, u_id2),
        FOREIGN KEY(co_artists) references artists );
```

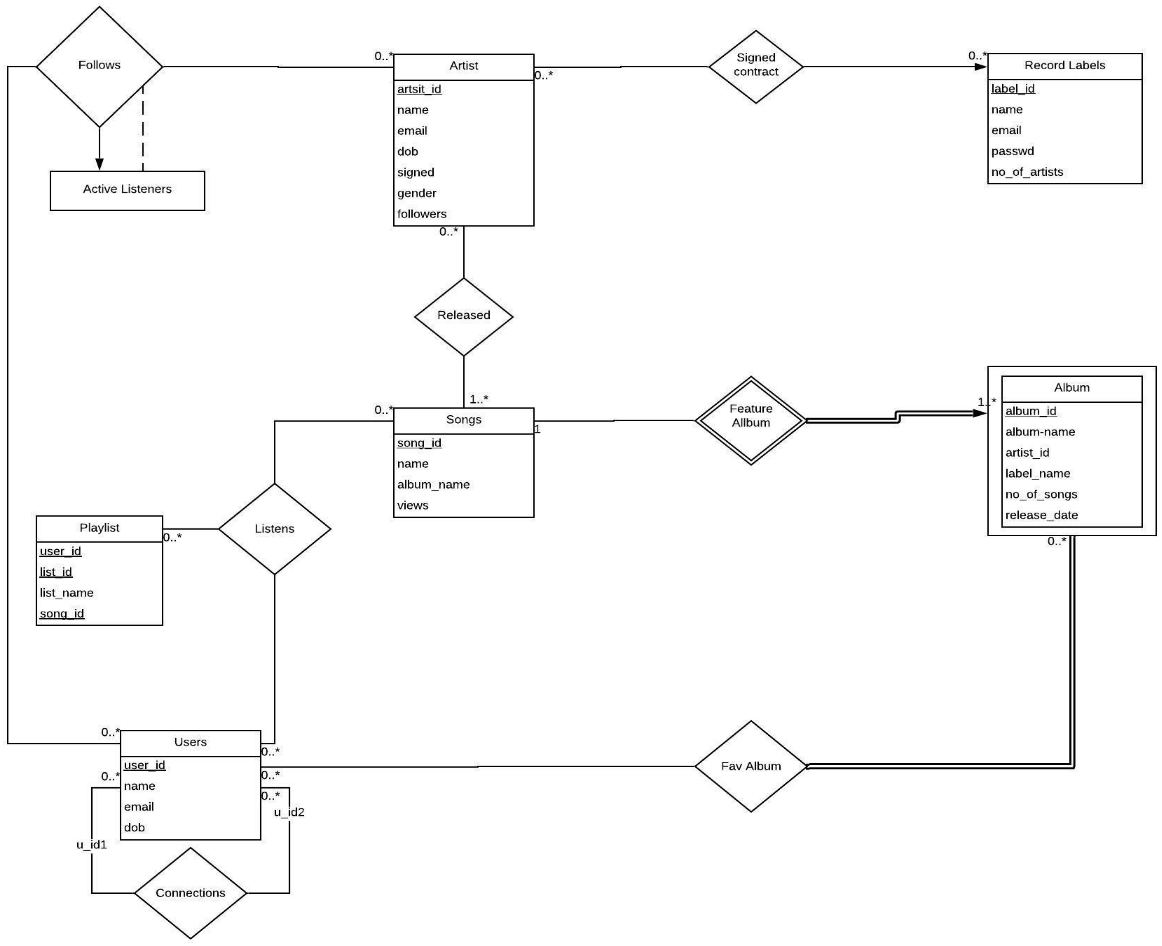
10. **Signed Artists Table:**

```
create table signed(
        artist_id       varchar(20),
        label_id        varchar(20),
        date_signed     date            not null,
        s_released      date            not null,
        PRIMARY KEY(artist_id, label_id),
        FOREIGN KEY(artist_id) references artists,
        FOREIGN KEY(label_id) references record_labels );
```

11. **Recommended Artists Table:**

```
create table recommended(
        genre           varchar(20),
        artist_id       varchar(20),
        song_id         varchar(20),
        artist_name     varchar(20),
        PRIMARY KEY(genre, song_id),
        FOREIGN KEY(artist_id) references artists );
```

ER DIAGRAM



Advika Singh (2018275) *Devender Singh (2018334)* *Kunal Verma (2018241)* *Nikhil Krishnan (2018248)* *Pratham Gupta (2018072)*

Queries involving Relational Algebraic Operations

1. SELECT * FROM users WHERE email= ' ***email id*** ' and passwd= ' ***password*** ';
2. SELECT * FROM artists WHERE name = ' ***artist name*** ' and passwd = ' ***password*** ';
3. SELECT * FROM users;
4. SELECT * FROM artists WHERE name= ' ***artist name*** ';
5. SELECT name FROM artists WHERE name NOT IN (SELECT artist_name FROM recommended);

6. SELECT user_id FROM users WHERE user_id NOT IN (SELECT user_id FROM fav_artists);
7. SELECT name FROM songs WHERE genre= " ***genre 1*** " OR genre = " ***genre 2*** ";
8. SELECT signed, followers FROM artists WHERE gender = "f" OR gender = "o";
9. SELECT DISTINCT a2.artist_name FROM songs as a1, songs as a2 WHERE a1.song_id != a2.song_id and a1.name = a2.name;
10. SELECT DISTINCT l2.name FROM labels as l1, labels as l2 WHERE l1.no_of_artists < l2.no_of_artists;

1. $\sigma_{email='id' \cap passwd='pswd'}$ (users)
2. $\sigma_{name='nm' \cap passwd='pswd'}$ (artists)
3. σ (users)
4. $\sigma_{name='nm''}$ (artists)
5. $\prod_{name}$ (artists) - $\prod_{name}$ (recommended)
6. $\prod_{user\_id}$ (users) - $\prod_{user\_id}$ (fav_artists)
7. $\prod_{name}$ ($\sigma_{genre="g1" \cup genre="g2"}$ (songs))
8. $\prod_{signed, followers}$ ($\sigma_{gender="f" \cup gender="o"}$ (artists))
9. $\prod_{a2.artist\_name}$ ($\sigma_{a1.song\_id != a2.song\_id \cap a1.name=a2.name}$ (a1(songs) a2(songs)))
10. $\prod_{l2.name}$ ($\sigma_{l1.no\_of\_artists < l2.no\_of\_artists}$ (l1(labels) l2(labels)))

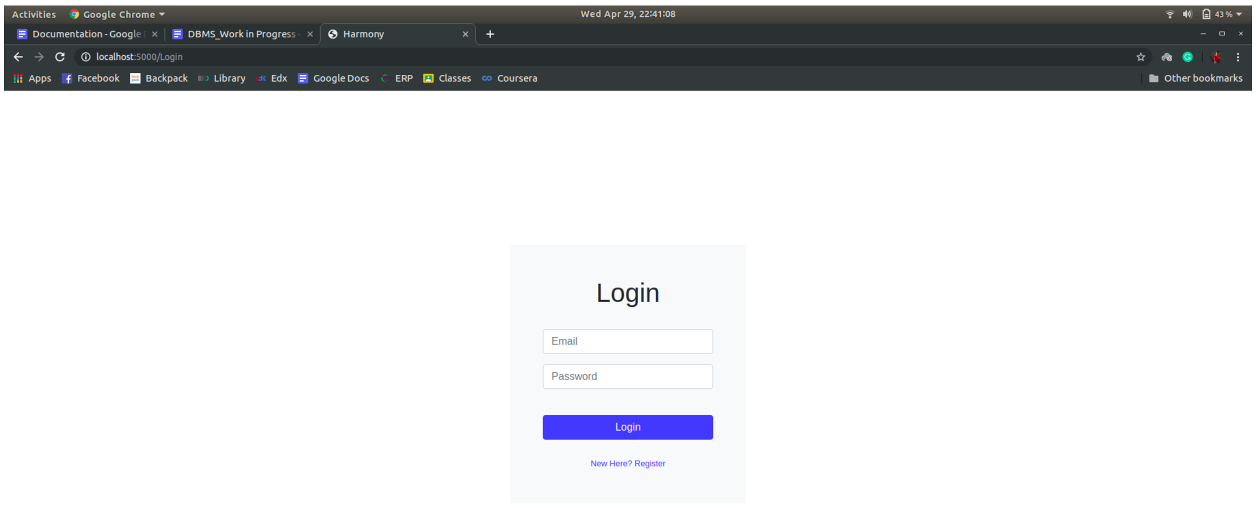
Embedded Queries

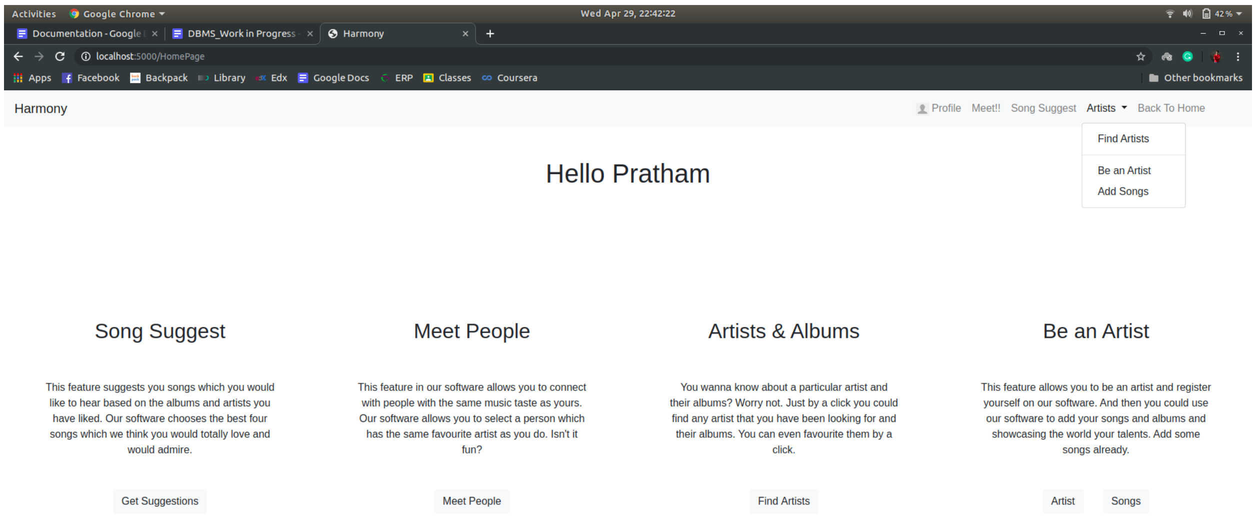
1) cur.execute('SELECT * FROM users')
2) cur.execute('SELECT * FROM users where email=%s and passwd=%s', (email, passwd))
3) cur.execute('SELECT * FROM users WHERE email=%s',(email,))
4) cur.execute('SELECT COUNT(*) FROM users')
5) cur.execute('INSERT INTO users(user_id, Name, email, passwd, gender, dob) VALUES (%s, %s, %s, %s, %s, %s)', (int(cur.fetchone()[0])+1,name, email, passwd, gender, dob))
6) cur.execute('UPDATE users SET passwd=%s WHERE user_id=%s',(request.form['passwd'], userLogged[0]))
7) cur.execute("SELECT * from fav_artists f where f.user_id = %s ", (userLogged[0],))
8) cur.execute(" select * from artists")
9) crr.execute("select * from artists where artist_id = %s ", (list_of_artists[0],))
10) cur.execute("select * from artists where artist_id in %s order by artist_id", (list_of_artists,))
11) cur.execute("select * from artists a where a.name = %s", (artistName,))
12) cur.execute("select * from fav_artists f where f.artist_id = %s", (data[0][0],))
13) cur.execute(" select * from users u where u.user_id in %s", (list_of_user,))
14) cur.execute('select * from users where user_id = %s', (userId,))
15) cur.execute('select * from connections c where c.u_id1 = %s', (userLogged[0],))
16) cur.execute('INSERT INTO connections(u_id1, u_id2, co_artist) VALUES (%s, %s, %s)', (userLogged[0], userId, data[0]))
17) cur.execute("select * from songs where genre in (select distinct genre from songs where artist_name in (select distinct artist_name from albums where artist_id in (select artist_id from fav_artists where user_id=\""+str(user_id)+"\"))) order by rand() limit 4;")
18) cur.execute("select * from songs where genre in (select distinct genre from songs where artist_name in (select distinct artist_name from albums where artist_id in (select artist_id from fav_artists where user_id=\""+str(user_id)+"\"))) order by rand() limit 4;")
19) resultValue = cur.execute(sql_query, (name,))
20) cur.execute(sql, (name,))
21) cur.execute(" select * from fav_albums where user_id = %s", (user,))
22) cur.execute(" select * from fav_artists where user_id = %s", (user,))

Advika Singh (2018275) *Devender Singh (2018334)* *Kunal Verma (2018241)* *Nikhil Krishnan (2018248)* *Pratham Gupta (2018072)*

23) cur.execute("INSERT INTO fav_albums(user_id, album_id) VALUES(%s, %s)",(user, id))
24) cur.execute("INSERT INTO fav_artists(user_id, artist_id) VALUES(%s, %s)",(user, artistID))
25) cur.execute('SELECT * FROM artists WHERE name=%s',(name,))
26) cur.execute('SELECT COUNT(*) FROM artists')
27) cur.execute('INSERT INTO artists(artist_id, name, email, passwd, dob, signed, gender, followers) VALUES (%d, %s, %s, %s, %s, %d, %s, %d)', (int(cur.fetchone()[0])+1,name, email, passwd, dob, 0, gender, 0))
28) cur.execute('select * from artists where name = %s and passwd = %s ', (artist_name, passwd))
29) cur.execute('SELECT * FROM songs WHERE artist_name=%s',(artist_name,))
30) cur.execute('SELECT COUNT(*) FROM songs')
31) cur.execute('INSERT INTO songs(song_id, name, artist_name, label_name, album_name, release_date, views, genre, language) VALUES (%d, %s, %s, %s, %s, %d, %s, %s)', (int(cur.fetchone()[0])+1,name, artist_name, label_name, album_name, dor, 0, genre, lang))

Screenshots of GUI





This feature suggests you songs which you would like to hear based on the albums and artists you have liked. Our software chooses the best four songs which we think you would totally love and would admire.

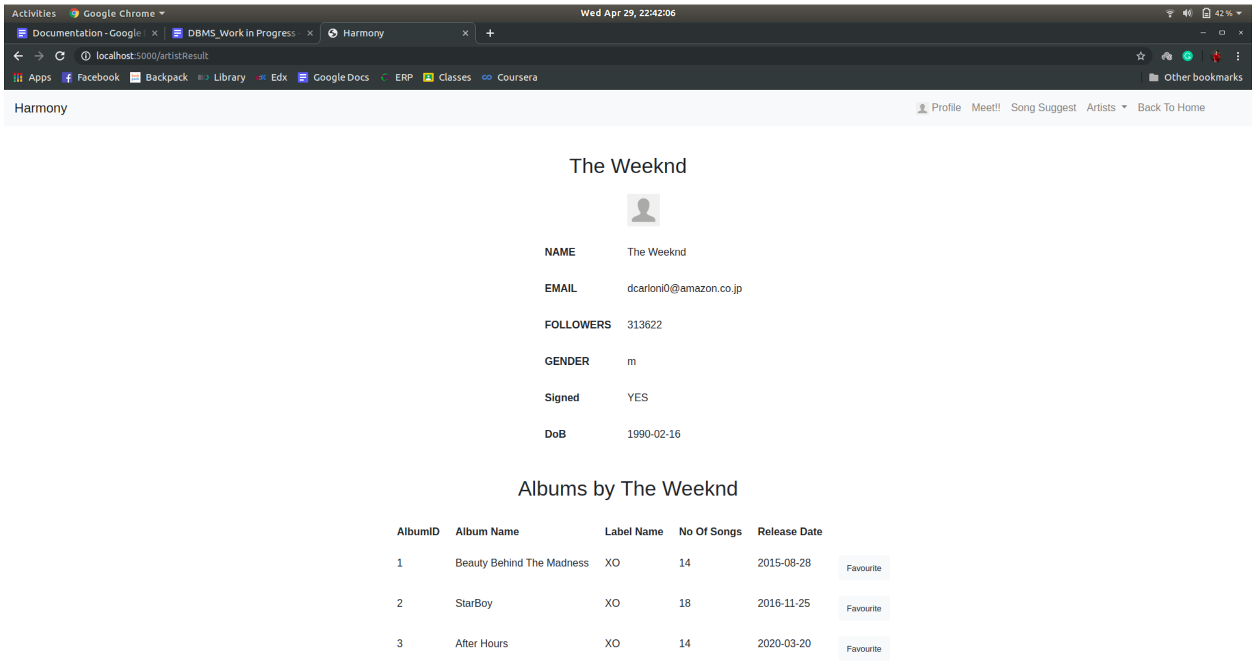
This feature in our software allows you to connect with people with the same music taste as yours. Our software allows you to select a person which has the same favourite artist as you do. Isn't it fun?

You wanna know about a particular artist and their albums? Worry not. Just by a click you could find any artist that you have been looking for and their albums. You can even favourite them by a click.

This feature allows you to be an artist and register yourself on our software. And then you could use our software to add your songs and albums and showcasing the world your talents. Add some songs already.

Advika Singh (2018275) *Devender Singh (2018334)* *Kunal Verma (2018241)* *Nikhil Krishnan (2018248)* *Pratham Gupta (2018072)*

Team: 48 *Team Name: C#*

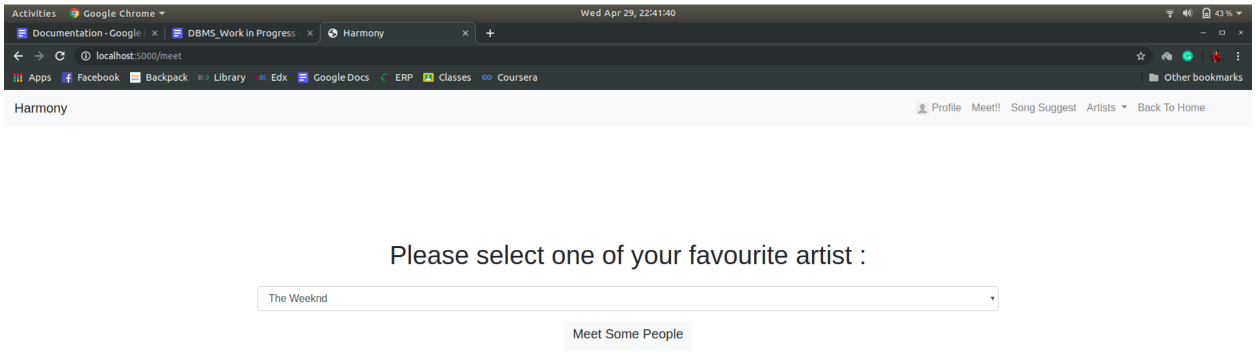


The Weeknd

NAME	The Weeknd
EMAIL	dcarloni0@amazon.co.jp
FOLLOWERS	313622
GENDER	m
Signed	YES
DoB	1990-02-16

Albums by The Weeknd

AlbumID	Album Name	Label Name	No Of Songs	Release Date	
1	Beauty Behind The Madness	XO	14	2015-08-28	Favourite
2	StarBoy	XO	18	2016-11-25	Favourite
3	After Hours	XO	14	2020-03-20	Favourite

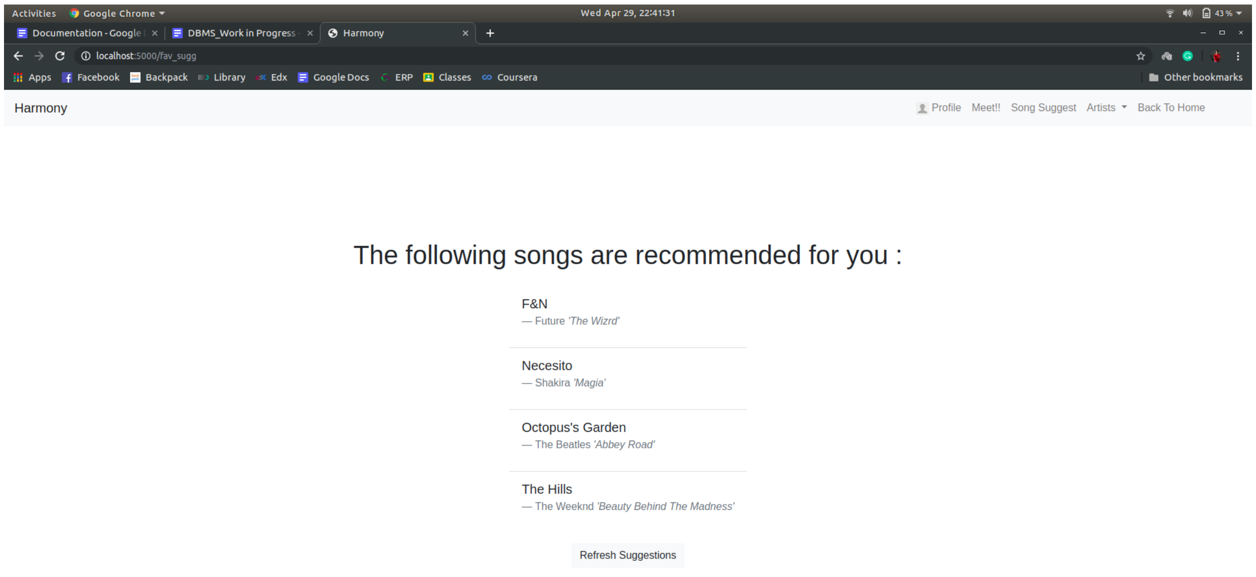


Please select one of your favourite artist :

The Weeknd

Meet Some People

Advika Singh (2018275) *Devender Singh (2018334)* *Kunal Verma (2018241)* *Nikhil Krishnan (2018248)* *Pratham Gupta (2018072)*



Harmony

Profile Meet!! Song Suggest Artists Back To Home

The following songs are recommended for you :

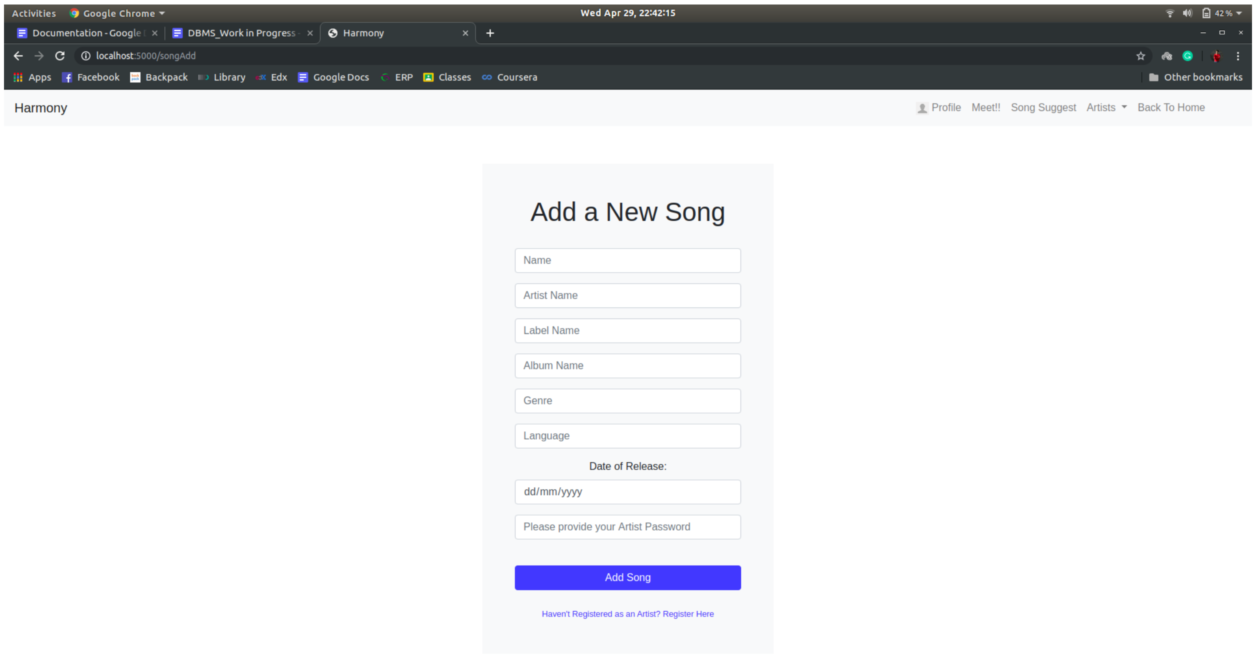
F&N
— Future *'The Wizrd'*

Necesito
— Shakira *'Magia'*

Octopus's Garden
— The Beatles *'Abbey Road'*

The Hills
— The Weeknd *'Beauty Behind The Madness'*

Refresh Suggestions



Harmony

Profile Meet!! Song Suggest Artists Back To Home

Add a New Song

Name

Artist Name

Label Name

Album Name

Genre

Language

Date of Release:

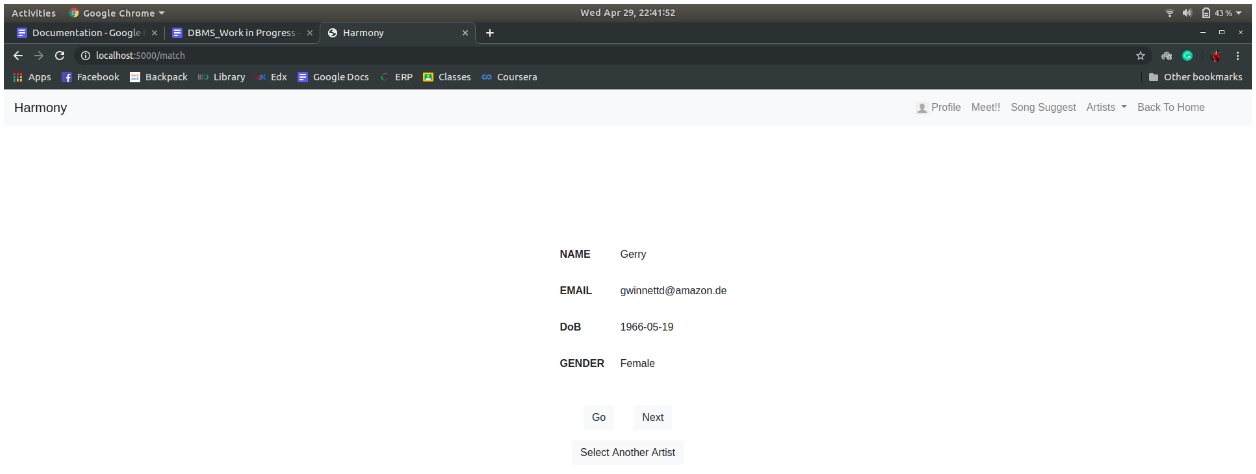
dd/mm/yyyy

Please provide your Artist Password

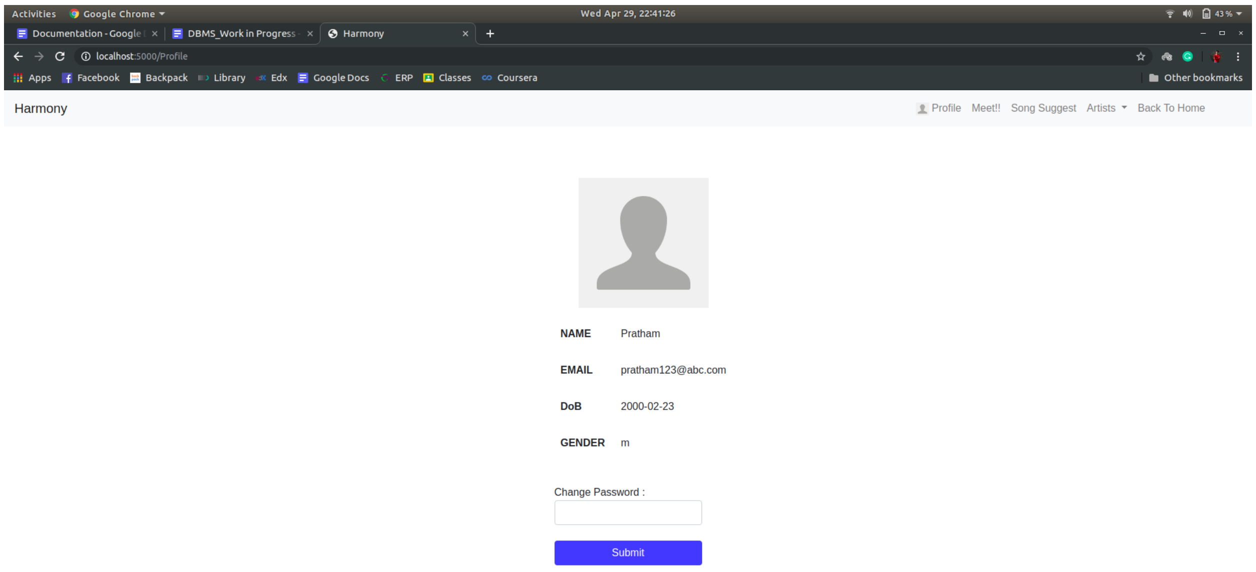
Add Song

Haven't Registered as an Artist? Register Here

Team: 48 Team Name: C#



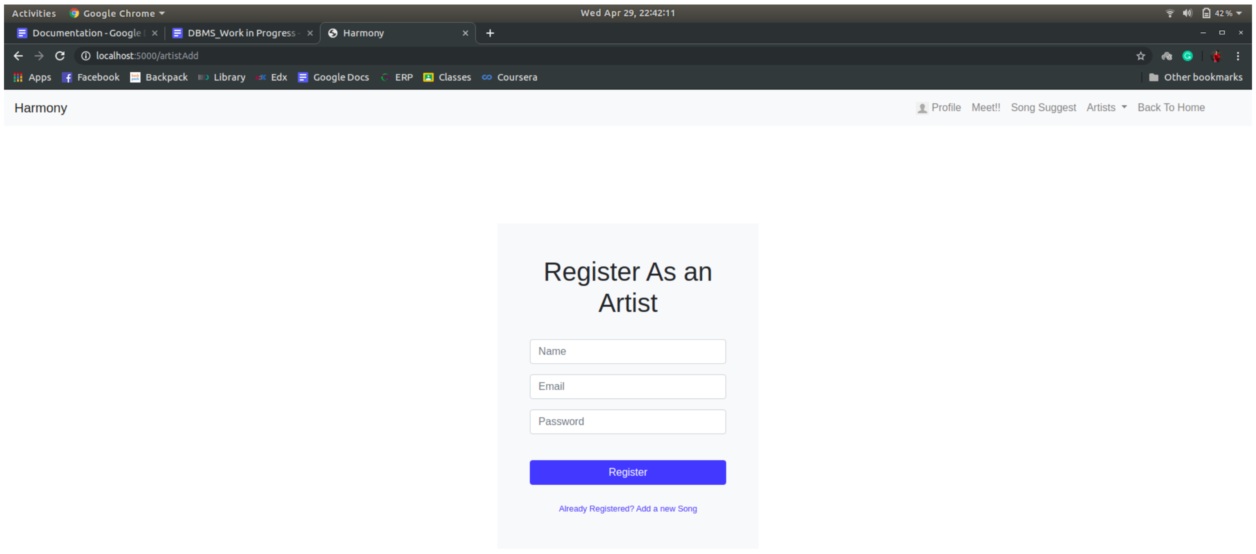
NAME	Gerry
EMAIL	gwinnettd@amazon.de
DoB	1966-05-19
GENDER	Female



NAME	Pratham
EMAIL	pratham123@abc.com
DoB	2000-02-23
GENDER	m

Advika Singh (2018275) Devender Singh (2018334) Kunal Verma (2018241) Nikhil Krishnan (2018248) Pratham Gupta (2018072)

Team: 48 *Team Name: C#*



Register As an Artist

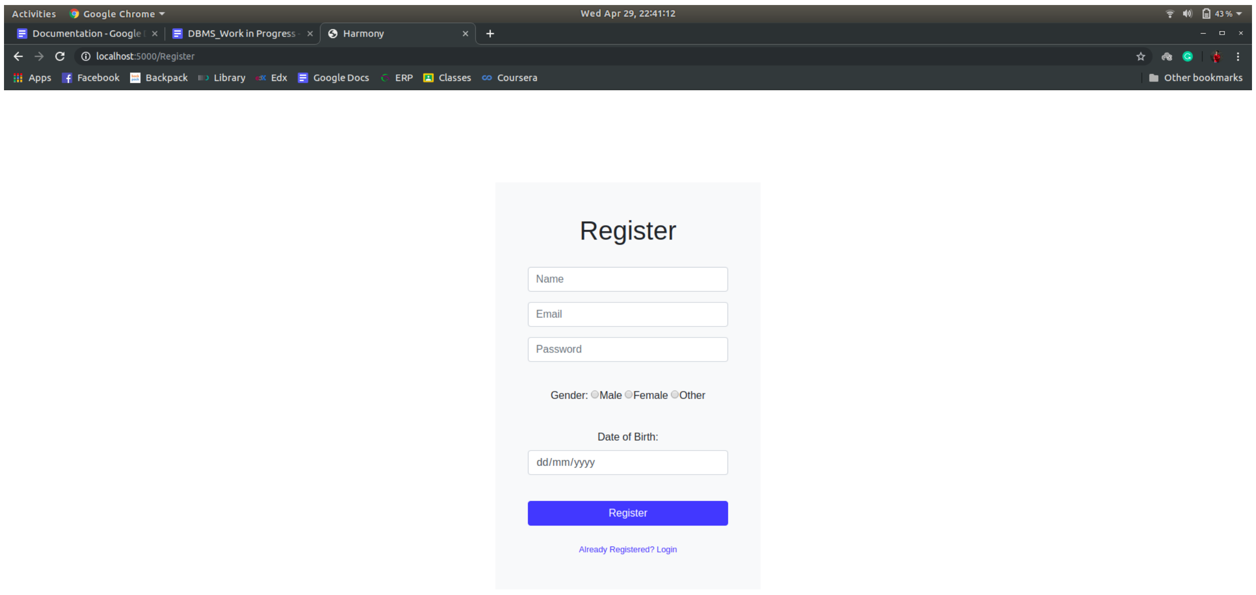
Name

Email

Password

Register

Already Registered? Add a new Song



Register

Name

Email

Password

Gender: Male Female Other

Date of Birth:

dd/mm/yyyy

Register

Already Registered? Login

Advika Singh (2018275)	*Devender Singh (2018334)*	*Kunal Verma (2018241)*	*Nikhil Krishnan (2018248)*	*Pratham Gupta (2018072)*

Software requirements

Python 3.3+

The underlying programming language used here to build web applications and web APIs is Python. Python can also analyze and visualize data and test software, even if the software being tested was not written in Python.
Concepts of python used:
Application dependency handling via the build-in venv (virtual environment) and pip.

MySQL (Relational Database)

A Database is an abstraction over an operating system's file system. MySQL is an open source relational database implementation for storing and retrieving data.

Flask (Web framework)

Flask is a Python web framework built with a small core and easy-to-extend philosophy. Flask depends on the Jinja template engine and the Werkzeug WSGI toolkit.

Flask-MySqldb (Connector)

Accessing MySQL from a Python application requires a database driver (also called a "connector"). The one we used is “mysqlclient” which is a fork of MySQLdb that supports Python 2 and 3. Flask-MySQLdb provides a MySQL connection for Flask. Flask-MySQLdb depends, and will install for you, recent versions of Flask and mysqlclient.

HTML

Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript.

CSS (Web design)

Cascading Style Sheet (CSS) files contain rules for how to display and lay out the HTML content when it is rendered by a web browser. CSS separates the content contained in HTML files from how the content should be displayed, to allow reusability and better management.

AT LAST, INSPIRATION TO CREATE SOMETHING BIG!!

