INDIAN INSTITUTE OF TECHNOLOGY INDORE
Minor Project: CS207

-Amey Patel (160001003) and Gvs Akhil (160001021)

DIGITAL MUSIC DISTRIBUTION

PROBLEM STATEMENT

The Digital Music Distribution System acts as an interface between the seller and buyer of a song, providing them with a virtual market platform to buy and sell music tracks. A customer can choose a song that he wishes to purchase based on many criterion including provided to him which include genre, the name of the artist, the track length and many other such criterion.The customer , if he desires so, can choose a combination of them in order to fulfil his requirements. A customer can also view it's specifications such as genre or the myriad of other data available in the database. If a cuustomer desires, he may add the song to his cart, which can be accessed later to complete the it's purchase.

The Music Library is designed in such a way that it hits two with a stone. Firstly, by storing the details of the songs that have been purchased by the customer earlier, it gathers enough data that suggests them music that suit their preferences. Secondly, by collecting data of the sales and the location of the customers, the supplier can identify his potential targets and take steps that will help him propagate his ideas.

The library,being hosted on the web, acts as an easy host for sellers willing to sell or publicize their songs/albums for free.

FUNCTIONALITIES

CREATE ACCOUNTS

Any seller interested in putting up his/her music on the website can do so. Both individual artist and organizations are allowed to sell music.

While creating an account, the user is asked to enter some basic information including his name, location,email address, phone number. It also incorporates a column which specifies if the user is an artist, a band or an organization.The user is then requested to choose a username. This username must be unique and must follow certain constraints such as its length should at least be 2 characters. After choosing a unique user name, the user is requested to enter a password which should be at least 8 characters in length and containing at least one capital letter to ensure password security.

All the specified data will be added to the database by creation of a new row in the database.The user can now complete their profile by providing details such as their location, their e-mail address, their phone number and their description.

ADDING TRACK

The seller can add a song by completing certain base requirements. He must enter the name of the song, the artist, the genre, duration of the song.

The seller can provide more details about the song such as mode, tempo,liveness,speechiness etc in order to help the user be more specific about his requirements. This helps the user choose the song better and thereby, there is estimated higher sale of the music.

These details shall be requested from the seller itself. However,if the seller is unable to provide these additional sophisticated data,they are,if possible, obtained through Spotify's API, which provides specifications of song based on certain criteria. In case of failure to find any such match, it would leave these fields empty in the database.

REMOVING TRACK

If the seller wishes to add a song for a limited time and then remove it without losing the statistics associated to it, the system provides a solution for it too. It removes the listing of

the song from the website itself,however, it retains the data such as the sale statistics associated to it in the database.

EFFICIENT BROWSING

The customer can browse through the collection of music efficiently with the help of filters. The customer can use the filter to display specific titles based on the filter criteria. The filters will have criteria such as Genre, Artist, Album, Duration, Price etc. The customer is free to combine multiple criteria together when searching. The customer can also sort the displayed music based on audio features of the song such as duration, tempo, loudness etc.

Through the use of filters and sorting, the customer can find exactly the type of music he prefers.

ADJUSTING FOR DISCOUNT

The Distribution system allows the sellers to incorporate special discounts during festive seasons. The seller can modify the price of a song. This aims at improving the sales and publicizing the artist at the expense of the profit per sale.

This is an important feature as most artist look at a festive season as an opportunity to increase the sales of their albums while envisaging and implementing a long-term model.

MANAGING CART

The system provides the buyer with a cart,which would store the songs the buyer is willing to pay for. This cart can store multiple songs, thereby allowing the buyer to purchase all of them at once. Hence,saving him from the trouble of purchasing each music individually.The system allows for the buyer to hop between the web pages without affecting the cart.

The user can then check out and confirm his payment. This payment can be made via multiple modes. It allows for the use of paytm, paypal, credit or debit cards. The details of the card such as the Card Number and CVV are, however not stored in the database to ensure protection of consumer rights.

SALES STATISTICS

Sales statistics provides the seller with crucial information about the performance of his product(s).

This utilitarian statistics provide the sellers with an insight and model to implement further strategies in order to enhance their profitability.The seller can view his total number of sales, total revenue, Revenue Breakdown by Genre/Album etc, Popularity of product in countries etc. The information would also be shown in a convenient graphical format to enhance readability.

The seller can then use these statistics to make informed financial decisions such as deciding the price for a future track release.

CUSTOMER STATISTICS

The customer will have access to view his/her spending statistics. The total number of tracks purchased, total money spent and spending breakdown by genre/album etc. The customer can then track his/her expenditure and adjust their purchasing frequency accordingly.

These features let the customer know that the purchasing process is transparent and hence it would help gain customer trust which in turn will increase customer retention.

Admin Privilleges

The Admin is the owner of the site and the administrator account has full privileges and control over the whole database and has the ability to add/edit/delete any information stored in the database.

The admin can help customers/sellers recover their account in the case of a hijacked account or if they misplaced their login information.

Implementation Details

Front-end:

- HTML
- CSS
- JavaScript

Back-end:

- PHP
- Django

1. **HTML** – Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. HTML defines the structure and semantics of the web application.

2. **CSS** – Cascading Style Sheets(CSS) is a stylesheet language used to describe the presentation of a document written in HTML. It describes how elements are rendered on screen.

3. **JavaScript** – JavaScript (JS) is a full-fledged dynamic programming language that, when applied to and HTML document, can provide dynamic interactivity on websites. It can be used to control web pages on the client side and even server-side programs.

4. **PHP** – PHP is a server scripting language, and a powerful tool for making dynamic and interactive web pages. It can also be embedded into HTML.

5. **Django** – Django is a free,open-source and high-level python web application framework. It is a collection of python libraries which allow us to quickly and efficiently create web applications and is suitable for both frontend and backend.

Data Gathering

To gather data for the tables in the database, we use a **web scraper**.

Web scraping is a computer software technique of extracting information from websites.

The web scraper would be made with the use of python and its libraries such as Scrapy/BeautifulSoup.

Uses -

- Generate list of tracks.
- Communicate with Spotify's API to get audio features of a specific track.
- Gather information related to artists.

E-R DIAGRAMS

Entity relationship between Cart and Customer

1) Customer(E)

- *Customer ID*
- *First Name*
- *Last Name*
- *Mobile Number*

- *Email_ID*
- *Addresss*
- *Username*
- *Password*
- *Gender*
- *Age*

2) Cart(WE)

- *Track IDs*
- *Amount*

3) Adds Products to(R)

This is a relationship between weak entity set 'Cart' and strong entity set 'Person'. Each customer has his/her cart which keeps the previously added products saved in the cart.

Entity Relationship Between Customer and Order

1) Customer(E)

- *Customer ID*
- *First Name*
- *Last Name*
- *Mobile Number*
- *Email_ID*
- *Addresss*
- *Username*
- *Password*
- *Gender*
- *Age*

2) Order(E)

- *Order ID*
- *Track IDs*

- *Date*
- *Original Price*
- *Discount*
- *Price Paid*

3) Places(R)

This is a relationship between the entity set 'Customer' and the entity set 'Order'. Each customer places certain orders which consist of tracks.

<u>Entity Relationship Between Artist and Tracks</u>

1) Artist (E)

- *<u>Artist ID</u>*
- *First Name*
- *Last Name*
- *Home City*
- *Rating*
- *Date Of Birth*
- *Gender*
- *Age*

2) Tracks (E)

- *<u>Track ID</u>*
- *Name*
- *Release Date*
- *Language*
- *Price*
- *Danceability*
- *Energy*
- *Key*
- *Loudness*

- *Mode*
- *Speechiness*
- *Acousticness*
- *Instrumentalness*
- *Liveness*
- *Valence*
- *Tempo*
- *Duration_ms*
- *Time_signature*

3) Makes (R)

This is a relationship between the entity set 'Artist' and the entity set 'Tracks' where the latter is the track created by the Artist.

<u>Entity Relationship Between Album and Track</u>

1) Album(E)

- *<u>Album ID</u>*
- *Name*
- *Release Date*
- *No. of songs*

2) Tracks (E)

- *<u>Track ID</u>*
- *Name*
- *Release Date*
- *Language*
- *Price*
- *Danceability*
- *Energy*
- *Key*
- *Loudness*
- *Mode*
- *Speechiness*

- *Acousticness*
- *Instrumentalness*
- *Liveness*
- *Valence*
- *Tempo*
- *Duration_ms*
- *Time_signature*

3) Contains(R)

The relationship establishes the Tracks that were released as a part of an Album.

<u>Entity Relationship Between Track and Genre</u>

1) Tracks (E)

- *<u>Track ID</u>*
- *Name*
- *Release Date*
- *Language*
- *Price*
- *Danceability*
- *Energy*
- *Key*
- *Loudness*
- *Mode*
- *Speechiness*
- *Acousticness*
- *Instrumentalness*
- *Liveness*
- *Valence*
- *Tempo*
- *Duration_ms*
- *Time_signature*

2) Genre(E)

- *Genre ID*
- *Genre*

3) Categorized By(R)

The relationship establishes that a track belongs to a specific genre.

Entity Relationship Between Seller and Track

1) Seller(E)

- *Seller_ID*
- *Name*
- *Telephone Number*
- *Email ID*
- *Username*
- *Password*

1) Tracks (E)

- *Track ID*
- *Name*
- *Release Date*
- *Language*
- *Price*
- *Danceability*
- *Energy*
- *Key*
- *Loudness*
- *Mode*
- *Speechiness*
- *Acousticness*
- *Instrumentalness*
- *Liveness*
- *Valence*
- *Tempo*
- *Duration_ms*

- *Time_signature*

3) Sells (R)

This relationship establishes that each track is sold by a specific seller.

<u>Entity Relationship Between Order and Seller</u>

1) Order (E)

- *<u>Order ID</u>*
- *<u>Track IDs</u>*
- *Original Price*
- *Discount*
- *Price Paid*

1) Seller (E)

- *<u>Seller_ID</u>*
- *Name*
- *Telephone Number*
- *Email ID*
- *Username*
- *Password*

3) Fulfilled By (R)

This relationship establishes that orders are fullfilled by sellers.

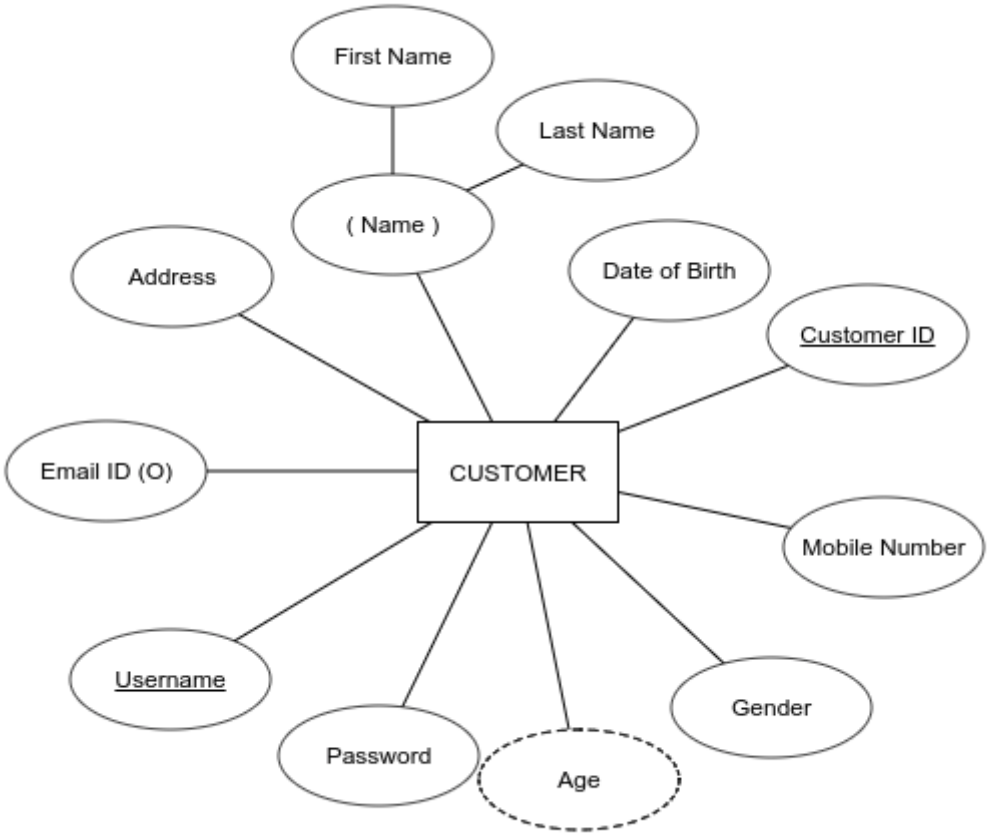
ER DIAGRAMS

ATTRIBUTE DESCRIPTION

Attributes Of Entity Track

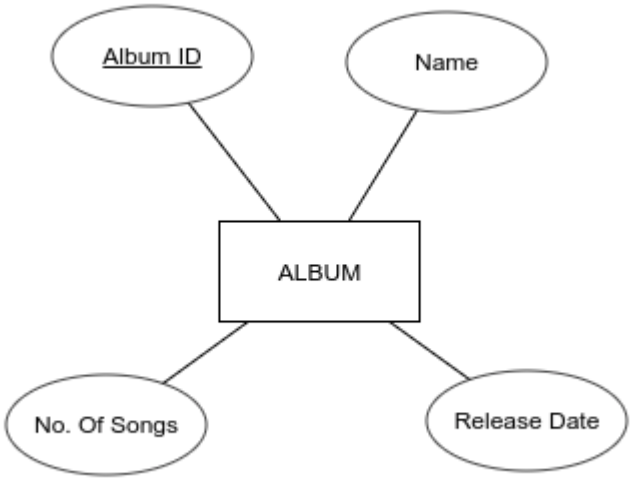
The entity set ‘Tracks’ has attributes such as the length of the track, the language it is recorded in and many other such attributes. Each of the track has it’s own Track ID which uniquely identifies it.

Attributes Of Entity Customer



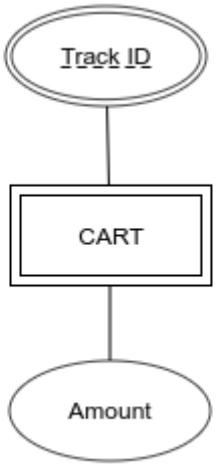
The entity set 'Customer' contains attributes such as User name and password which are stored in the database to ensure authorization, allowing only the customer to access his own account. It has other attributes which contain the details about the customer himself, including the Customer ID which is uniquely apportioned to each person.

Attributes Of Entity Album



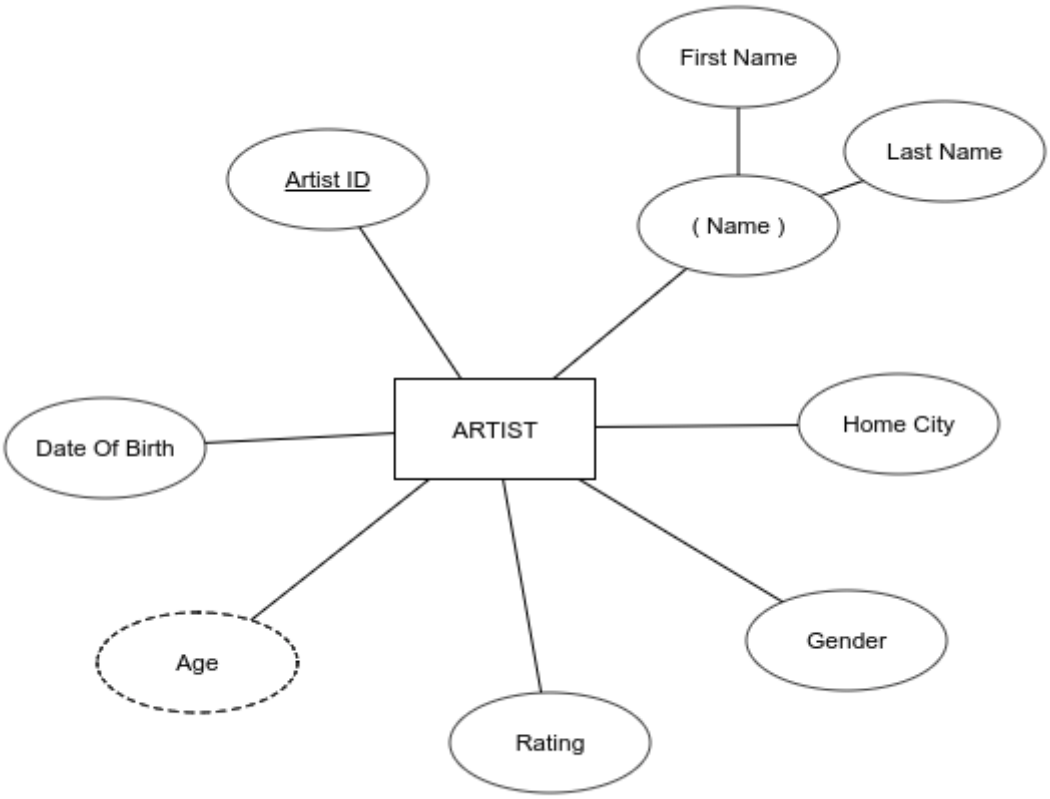
The entity set Album is directly related to the tracks which it contains. It has its own unique Album ID and contains attributes such as the no. of song and it's release date too.

Attributes Of Entity Cart



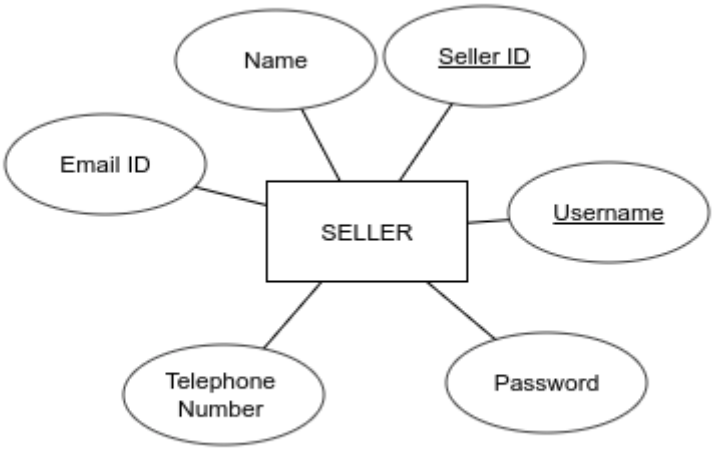
The entity set ‘Cart’ is treated as a weak entity set which is dependent on the strong entity set ‘Customer’ for its existence. A cart will exist only if the customer has an account on the website.Further, the attributes of Cart include the Track ID of tracks that it contains and the total cost which the customer has to pay in order to complete the transaction.

Attributes Of Entity Artist



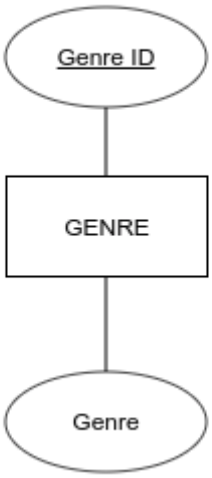
An artist is alloted an unique Artist ID to identify him uniquely. Each artist is given an rating based on his popularity and the average performance of his tracks.

Attributes Of Entity Seller



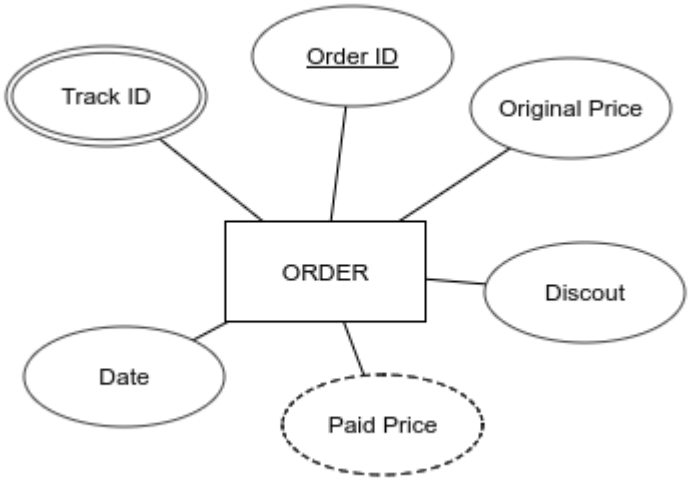
The entity set ‘Seller’ contains attributes such as User name and password which are stored in the database to ensure authorization, allowing only the seller to access their own account. It has other attributes which contain the details about the seller, including the Seller ID which is uniquely apportioned to each seller.

Attributes Of Entity Genre



The entity set 'Genre' contains a list of all Genres along with a unique ID which is associated with each genre.

Attributes Of Entity Order



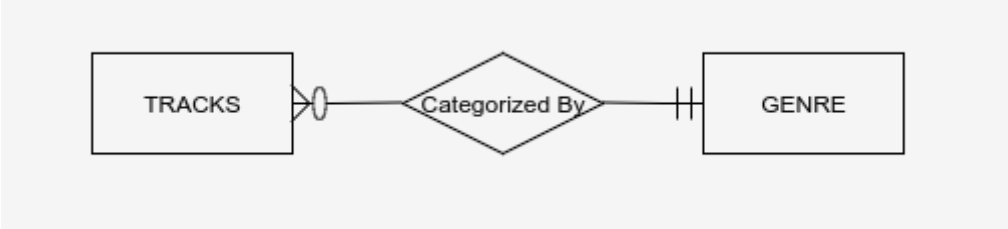
The entity set order contains a list of orders placed. Each row is defined by a unique value Order ID and consists of Track ID, Date, Original Price, Discount, Paid Prce.

RELATIONSHIPS BETWEEN ENTITIES

Relationship Between Customer and Cart

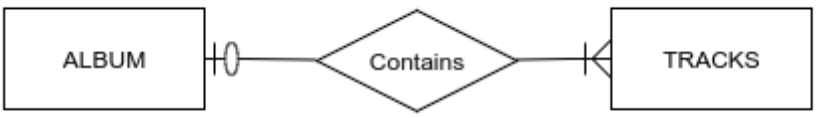
The relationship is an one-to-one relationship from customer to cart. Cart is a weak entity set and is dependent on its parent entity set 'Customer' for existence.The entity set 'Cart' has total participation in the relation.

Relationship Between Tracks and Genre



The relationship is a many-to-one relationship from tracks to genre. Tracks has total participation in the relationship and Genre has optional participation in the relationship.

Relationship Between Album and Tracks

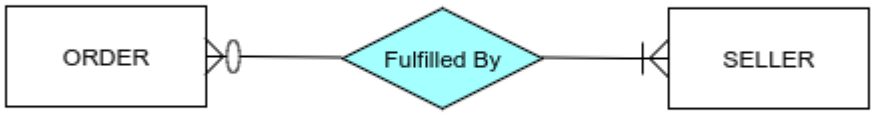


The relationship is an one-to-many relationship from Album to Tracks. Album has mandotary participation and Tracks has optional participation in the relation.

Relationship Between Artist and Tracks

The relationship is a many-to-one relationship from Artists to Tracks. Both Artists and Tracks have mandatory participation in the relation.

Relationship Between Order and Seller



The relationship is a many-to-many relationship from Order to Seller. Order has mandatory participation and Seller has optional participation in the relation.

Relationship Between Customer and Order

The relationship is a many-to-one relationship from customer to order. Customer has optional participation and Order has mandatory participation in the relation.

Relationship Between Seller and Tracks

The relationship is a many-to-one relationship from seller to tracks. Seller has optional participation and Tracks has mandatory participation in the relation.

TRIGGERS AND CONSTRAINTS

1) ***Customer (E)***

- *Each Customer must have an unique mobile Number registered to his account.*
- *Each Customer must have an unique email ID registered to his account.*
- *Each Customer must have an unique username registered to his account.*
- *The password is required to be strong. This is ensured by forcing a constraint that it must contain atleast one uppercase alphabet, one lower case alphabet, a number and must be atleast 6 characters in length.*
- *The gender of an Customer must either be 'M'(Male) or 'F'(Female).*

2) ***Artist (E)***

- *Each artist is apportioned a rating on a scale of 5 based on his popularity and the sale of his track. We enforce a constraint on the 'rating' of an artist that it must be a real number between 0 and 5.*
- *The gender of an artist must either be 'M'(Male) or 'F'(Female).*

3) ***Seller***

- *Each Seller must have an unique mobile Number registered to his account.*
- *Each Seller must have an unique email ID registered to his account.*
- *Each Seller must have an unique username registered to his account.*
- *The password is required to be strong. This is ensured by forcing a constraint that it must contain atleast one uppercase alphabet, one lower case alphabet, a number and must be atleast 6 characters in length.*

4) ***Album (E)***

- *The no. of songs released in each album must be greater than 0.*

5) ***Tracks (E)***

- *The price of each track must be greater than or equal to 0.*
- *The total amount of the tracks in the cart should be greater than or equal to zero.*
- *The Danceability of an Audio Feature must be a real number from 0 to 1.*
- *The Energy of an Audio Feature must be a real number from 0 to 1.*
- *The Loudness of an Audio Feature must be a real number from 0 to 1.*
- *The Speechiness of an Audio Feature must be a real number from 0 to 1.*
- *The Acousticness of an Audio Feature must be a real number from 0 to 1.*

- *The Instrumentalness of an Audio Feature must be a real number from 0 to 1.*
- *The Liveness of an Audio Feature must be a real number from 0 to 1.*
- *The Valence of an Audio Feature must be a real number from 0 to 1.*
- *The Tempo of an Audio Feature must be a real number from 0 to 1.*
- *The Duration_ms of an Audio Feature must be a positive integer number.*

FUNCTIONAL DEPENDENCY

1) ***Customer (E)***

   - *Registration ID , Username , Email_ID ---> First Name ,Last Name, Mobile Number, Address, Password, Gender*

2) ***Artist (E)***

   - *Artist ID ---> First Name, Last Name, Home City, Rating, Date of Birth, Gender.*

3) ***Seller (E)***

   - *Seller_ID , Username --->Name, Telephone Number, Email ID, Password.*

4) ***Album (E)***

   - *Album ID ---> Name, Release Date, No. of songs.*

5) ***Tracks (E)***

   - *Track ID ---> Name,Release Date, Language, Price, Danceability, Energy, Key, Loudness, Mode, Speechiness,*

Acousticness, Instrumentalness, Liveness, Valence, Tempo, Duration_ms, Time_signature

6) ***Genre (E)***

- *Genre ID ---> Genre*

7) ***Cart (WE)***

- *Registration ID , Track ID ---> Amount*

8) ***Order (E)***

- *Order ID , Track ID ---> Date, Original Price, Discount, Price Paid*

Tables Deduced from ER Model

1. Track

Track ID	Name	Length	Price	Language	Release Date
..	..	..	..	..	..

Liveness	Valence	Danceability	Tempo	Energy	Duration_ms
..	..	..	..	..	..

Time_signature	Key	Loudness	Mode	Speechiness	Acousticness
..	..	..	..	..	..

Candidate Keys : {Track ID}

2. Album

Album ID	Name	Release Date	No. of Songs
..	..	..	..

Candidate Keys : { Album ID}

3. Cart

Customer ID	Track ID	Amount
..	..	..

Candidate Keys : { Customer ID, Track ID}

4. Seller

Seller ID	Name	Email ID	Telephone Number	Username	Password
..	..	..	..	..	..

Candidate Keys :{ { Seller ID} , { Username} , {Email ID} }

5. Genre

Genre ID	Genre
..	..

Candidate Keys : { Genre ID}

6. Order

Track ID	Order ID	Date	Original Price	Discount	Paid Price
..	..	..	..	..	..

Candidate Keys : { Track ID, Order ID}

7. Customer

Customer ID	First Name	Last Name	Date of Birth	Mobile Number	Gender
..	..	..	..	..	..

Age	Username	Password	Email ID	Address
..	..	..	..	..

Candidate Keys : {{ Customer ID} , {Username} ,{Email ID}}

8. Artist

Artist ID	First Name	Last Name	Home City	Gender	Rating	Age	Date of Birth
..	..	..	..	..	..	..	..

Candidate Keys : { Artist ID}

9. Artist and Track

Artist ID	Track ID
..	..

Candidate Keys : { Artist ID, Track ID}

10. Album and Track

Album ID	Track ID
..	..

Candidate Keys : { Album ID, Track ID}

11. Seller and Track

Seller ID	Track ID
..	..

Candidate Keys : { Seller ID, Track ID}

12. Seller and Order

Seller ID	Order ID
..	..

Candidate Keys : { Seller ID, Order ID}

13. Track and Genre

Track ID	Genre ID
..	..

Candidate Keys : {Track ID, Genre ID}

TRIGGERS

1) Trigger that inserts the date of the order placed

```
DROP TRIGGER if exists timestamp_on_insert_orders;
DELIMITER //
CREATE TRIGGER timestamp_on_insert_orders
```

```
BEFORE insert ON orders
      FOR EACH ROW BEGIN
        SET NEW.date = CURRENT_TIMESTAMP;
END
//
 DELIMITER ;
```

2) Trigger that updates the timestamp when an order is updated

```
DROP TRIGGER if exists timestamp_on_update_orders;
DELIMITER //
CREATE TRIGGER timestamp_on_update_orders
BEFORE update ON orders
      FOR EACH ROW BEGIN
        SET NEW.date = CURRENT_TIMESTAMP;
END
//
 DELIMITER ;
```

3) Trigger that inserts the timestamp on release of an album

```
DROP TRIGGER if exists timestamp_on_insert_album;
DELIMITER //
CREATE TRIGGER timestamp_on_insert_album
BEFORE insert ON album
      FOR EACH ROW BEGIN
        SET NEW.release_date =CURRENT_TIMESTAMP - INTERVAL
FLOOR(RAND() * 1000) DAY;
END
//
 DELIMITER ;
```

4) Trigger to change the number_of_songs in album after track deletion

```
DROP TRIGGER if exists delete_track_on_album;
DELIMITER //
CREATE TRIGGER delete_track_on_album
AFTER delete ON track
```

```
    FOR EACH ROW
    BEGIN
    UPDATE album
    set album.number_of_songs=album.number_of_songs-1
    where album.album_id=(select album_id from compose where
compose.track_id=old.track_id);

    if album.number_of_songs=0 THEN
    delete from album
    where album.album_id=(select album_id from compose where
compose.track_id=old.track_id);
    end if;

END
//
 DELIMITER ;
```

5) Trigger to update ‘categorisedby’ upon ‘track’ deletion

```
DROP TRIGGER if exists delete_track_on_categorisedby;
DELIMITER //
CREATE TRIGGER delete_track_on_categorisedby
BEFORE delete ON track
    FOR EACH ROW
    BEGIN
    delete from categorisedby
      where old.track_id=categorisedby.track_id;
END
//
 DELIMITER ;
```

6) Trigger to update ‘compose’ upon ‘track’ deletion

```
DROP TRIGGER if exists delete_track_on_compose;
DELIMITER //
CREATE TRIGGER delete_track_on_compose
BEFORE delete ON track
    FOR EACH ROW
```

```
    BEGIN
    delete from compose
      where old.track_id=compose.track_id;
END
//
 DELIMITER ;
```

7) Trigger to update ‘sells’ upon ‘track’ deletion

```
DROP TRIGGER if exists delete_track_on_sells;
DELIMITER //
CREATE TRIGGER delete_track_on_sells
BEFORE delete ON track
    FOR EACH ROW
    BEGIN
    delete from sells
      where old.track_id=sells.track_id;
END
//
 DELIMITER ;
```

8) Trigger to update ‘compose’ upon ‘track’ deletion

```
DROP TRIGGER if exists delete_track_on_customer;
DELIMITER //
CREATE TRIGGER delete_track_on_customer
BEFORE delete ON track
    FOR EACH ROW
    BEGIN
    delete from customer
      where old.track_id=customer.track_id;
END
//
 DELIMITER ;
```

9) Trigger to update ‘makes’ upon ‘artist’ deletion

```
DROP TRIGGER if exists delete_artist_on_makes;
DELIMITER //
```

```
CREATE TRIGGER delete_artist_on_makes
BEFORE delete ON artist
      FOR EACH ROW
      BEGIN
      delete from makes
        where old.artist_id=makes.artist_id;
END
//
 DELIMITER ;
```

10) Trigger to update ‘compose’ upon ‘album’ deletion

```
DROP TRIGGER if exists delete_album_on_compose;
DELIMITER //
CREATE TRIGGER delete_album_on_compose
BEFORE delete ON album
      FOR EACH ROW
      BEGIN
      delete from compose
        where old.album_id=compose.album_id;
END
//
 DELIMITER ;
```

11) Trigger to update ‘filfilledby’ upon ‘order’ deletion

```
DROP TRIGGER if exists delete_order_on_fulfilledby;
DELIMITER //
CREATE TRIGGER delete_order_on_fulfilledby
BEFORE delete ON orders
      FOR EACH ROW
      BEGIN
```

```
      delete from fulfilledby
        where old.order_id=fulfilledby.order_id;
END
//
 DELIMITER ;
```

STORED FUNCTIONS

1) Stored function to calculate the discount when track is present in cart.

```
DROP FUNCTION if exists count_discount;
DELIMITER //
CREATE FUNCTION count_discount(c int)
RETURNS INT
BEGIN
  DECLARE discount int;
   if c>=1000 THEN
       set discount=0.12*c;
    end if;

    if c<1000 and c>=100 THEN
       set discount=0.08*c;
    end if;

    if c<100 THEN
       set discount=0;
     end if;

  return discount;
END
//
DELIMITER ;
```

STORED PROCEDURES

1) Stored procedure that calculates the discount.

```
drop procedure if exists  apply_discount;
delimiter //
create procedure  apply_discount ()
begin
  DECLARE done INT DEFAULT false;
  DECLARE a INT;
  DECLARE b INT;
  DECLARE c INT;
  DECLARE cnt int ;
  DECLARE cur CURSOR FOR SELECT track_id,order_id,original_price
FROM orders;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
true;

  OPEN cur;

  loop1: LOOP
    FETCH cur INTO a,b,c;
    IF done THEN
      LEAVE loop1;

    END IF;
    if c>=1000 THEN
      update orders
        set discount=0.12*c
        where orders.track_id=a and orders.order_id=b;
    end if;

    if c<1000 and c>=100 THEN
      update orders
        set discount=0.08*c
        where orders.track_id=a and orders.order_id=b;
    end if;

    if c<100 THEN
      update orders
        set discount=0
        where orders.track_id=a and orders.order_id=b;
```

```
    end if;

  END LOOP;

end //
 delimiter ;
call  apply_discount();
```

2) Stored procedure to calculate the final price paid by the customer

```
drop procedure if exists  calculate_final_price;
delimiter //
create procedure  calculate_final_price ()
begin
  DECLARE done INT DEFAULT false;
  DECLARE a INT;
  DECLARE b INT;
  DECLARE c INT;
  DECLARE d INT;
  DECLARE cur CURSOR FOR SELECT
track_id,order_id,original_price,discount FROM orders;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
true;

  OPEN cur;

  loop1: LOOP
    FETCH cur INTO a,b,c,d;
    IF done THEN
      LEAVE loop1;

    END IF;
      update orders
        set paid_price=c-d
        where orders.track_id=a and orders.order_id=b;


  END LOOP;
```

```
end //
 delimiter ;
call  calculate_final_price();
```

3) Stored procedure to assign the ‘categorisedby’ table with values

```
drop procedure if exists  random_assign_categorisedby;
delimiter //
create procedure  random_assign_categorisedby ()
begin
  DECLARE done INT DEFAULT false;
  DECLARE a INT;
  DECLARE c INT DEFAULT 0;
  DECLARE d INT DEFAULT 0;
  DECLARE b,e CHAR(200);
  DECLARE cur CURSOR FOR SELECT track_id FROM track;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
true;

  OPEN cur;

  loop1: LOOP
    FETCH cur INTO a;
    IF done THEN
      LEAVE loop1;

    END IF;

    if mod(a,10)=0 THEN
      set b=10;
    else
      set b=mod(a,10);
    end if;
     replace into categorisedby values(a,b);

  END LOOP;

end //
```

```
 delimiter ;
call  random_assign_categorisedby();
```

4) Stored procedure to assign the ‘assign’ table with values

```
drop procedure if exists  random_assign_sells;
delimiter //
create procedure  random_assign_sells ()
begin
  DECLARE done INT DEFAULT false;
  DECLARE a INT;
  DECLARE c INT DEFAULT 0;
  DECLARE d INT DEFAULT 0;
  DECLARE b,e CHAR(200);
  DECLARE cur CURSOR FOR SELECT track_id FROM track;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
true;

  OPEN cur;

  loop1: LOOP
    FETCH cur INTO a;
    IF done THEN
      LEAVE loop1;

    END IF;

    if mod(a,50)=0 THEN
      set b=50;
    else
      set b=mod(a,50);
    end if;
     replace into sells values(a,b);

  END LOOP;

end //
 delimiter ;
```

```
call  random_assign_sells();
```

5) Stored procedure to assign the ‘compose’ table with values

```
drop procedure if exists  random_assign_compose;
delimiter //
create procedure  random_assign_compose ()
begin
  DECLARE done INT DEFAULT false;
  DECLARE a INT;
  DECLARE c INT DEFAULT 0;
  DECLARE d INT DEFAULT 0;
  DECLARE b,e CHAR(200);
  DECLARE cur CURSOR FOR SELECT track_id FROM track;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
true;

  OPEN cur;

  loop1: LOOP
    FETCH cur INTO a;
    IF done THEN
      LEAVE loop1;

    END IF;

    set b=ceil(a/40);
     replace into compose values(a,b);

  END LOOP;

end //
 delimiter ;
call  random_assign_compose();
```

6) Stored procedure to search tracks by artist name

```
DROP procedure if exists search_artist;
DELIMITER //
CREATE procedure search_artist(name varchar(100))
BEGIN

  select artist.first_name,track.track_id,track.name as track_name from
  track,makes,artist
  where
  makes.artist_id=artist.artist_id
  and
  track.track_id=makes.track_id
  and
   concat(artist.first_name," ",artist.last_name) like
concat("%",name,"%") ;

END
//
DELIMITER ;
call search_artist("Future");
```

6) Stored procedure to search tracks by genre

```
DROP procedure if exists search_genre;
DELIMITER //
CREATE procedure search_genre(name varchar(100))
BEGIN

  select genre,track.track_id,track.name as track_name from
  track,categorisedby,genre
  where
  track.track_id=categorisedby.track_id
  and
  categorisedby.genre_id=genre.genre_id
  and
   genre.genre like concat("%",name,"%") ;
```

```
END
//
DELIMITER ;
call search_genre("Pop");
```

6) Stored procedure to search tracks by artist name and genre

```
DROP procedure if exists search_genre_and_artist;
DELIMITER //
CREATE procedure search_genre_and_artist(artist_name
varchar(100),genre_name varchar(100))
BEGIN

  select a.first_name,b.genre,b.track_id,a.track_name from
 (select artist.first_name,track.track_id,track.name as track_name from
  track,makes,artist
  where
  makes.artist_id=artist.artist_id
  and
  track.track_id=makes.track_id
  and
   concat(artist.first_name," ",artist.last_name) like
concat("%",artist_name,"%") )as a,

  (select genre,track.track_id,track.name as track_name from
  track,categorisedby,genre
  where
  track.track_id=categorisedby.track_id
  and
  categorisedby.genre_id=genre.genre_id
  and
   genre.genre like concat("%",genre_name,"%") ) as b
  where a.track_id=b.track_id;

END
```

```
//
DELIMITER ;
call search_genre_and_artist("Future","Pop");
```

7) Stored procedure to find the album details of a given track

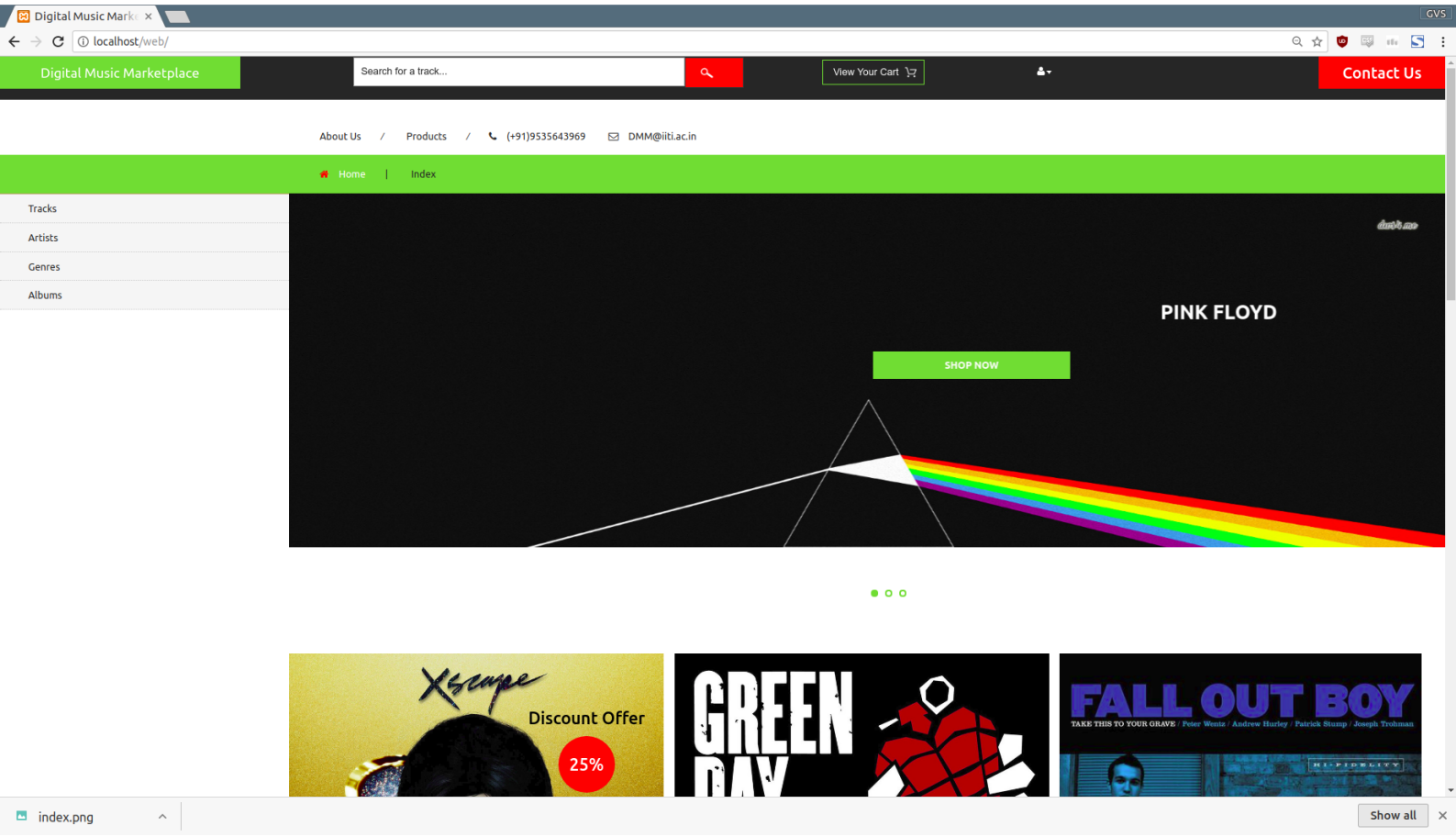
```
DROP procedure if exists find_album_details;
DELIMITER //
CREATE procedure find_album_details(IN id int)
BEGIN
  select * FROM
  album where
  album.album_id=(select album_id from compose where
compose.track_id=id);

END
//
DELIMITER ;
call find_album_details(5);
```

SCREENSHOTS

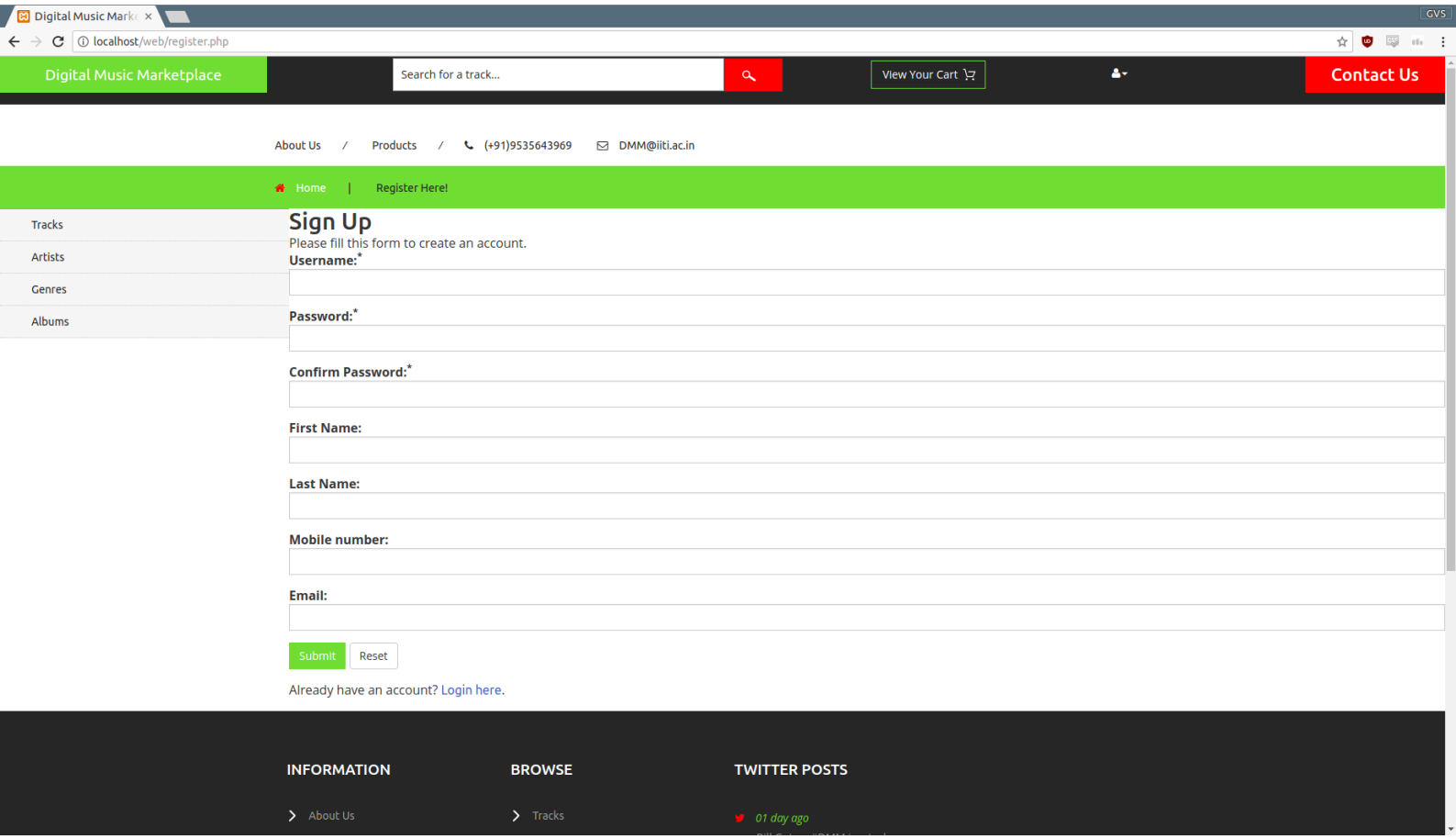
Index Page :-

This is the page which first opens up when you go to the website. It contains links to the product pages, website information pages, a ‘View Cart’ Button and a login/register page link.



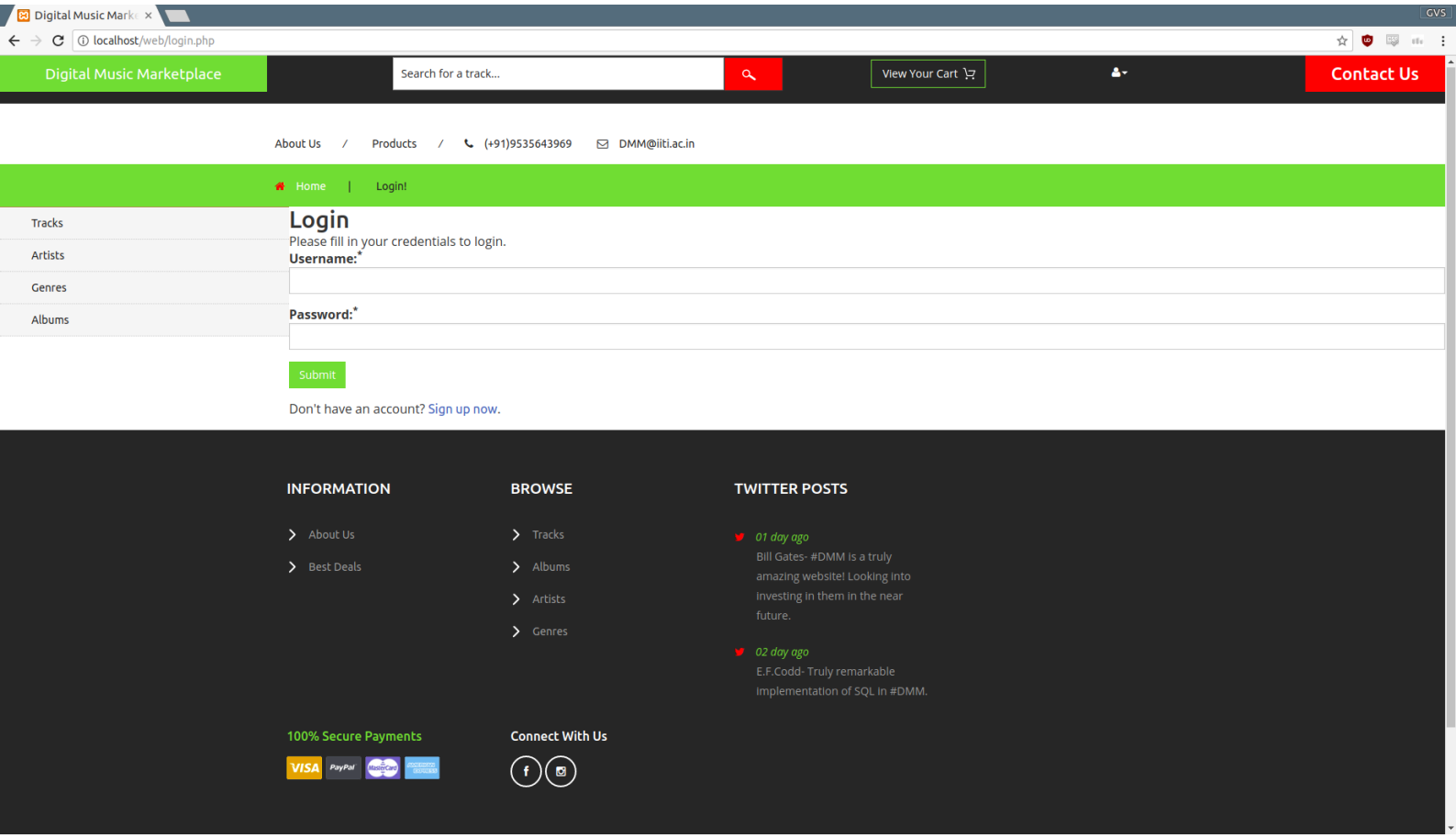
Sign Up: -

To register yourself in the customer database, you can visit the following page and enter your details.



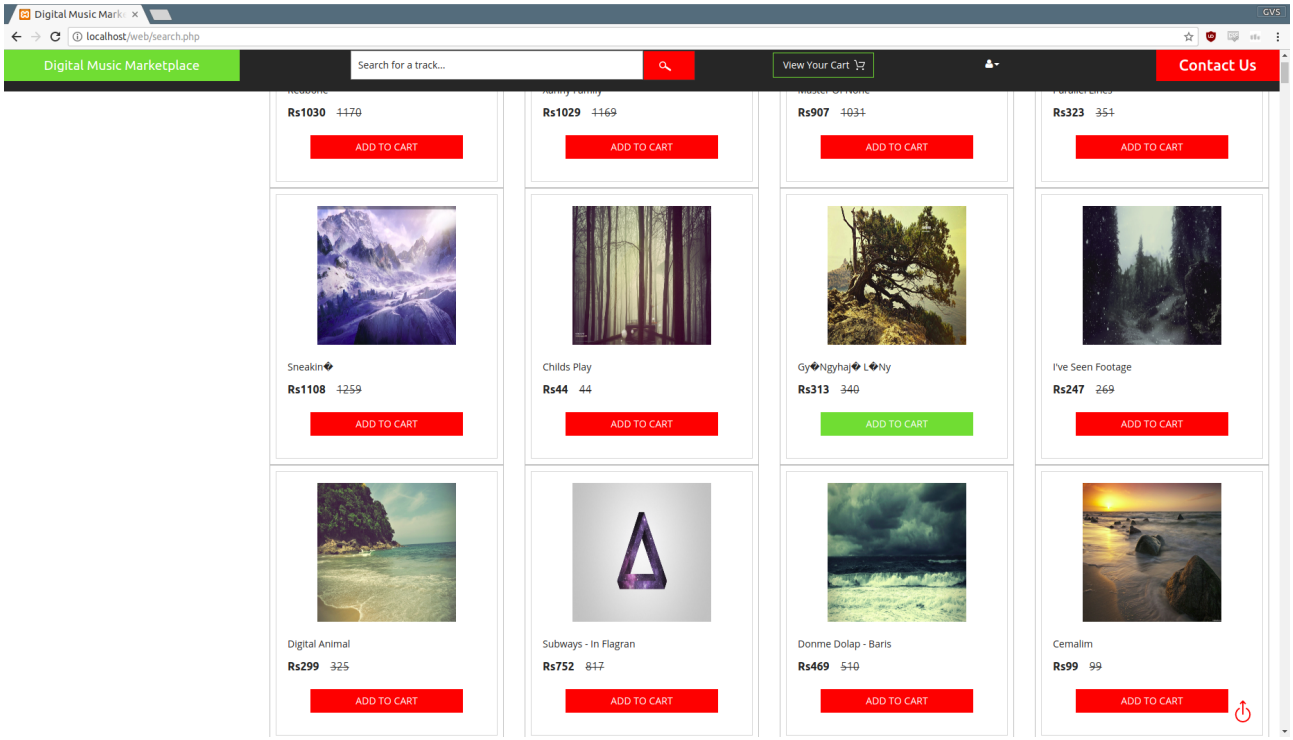
Log In:-

Once you sign up, you can login at the following page.

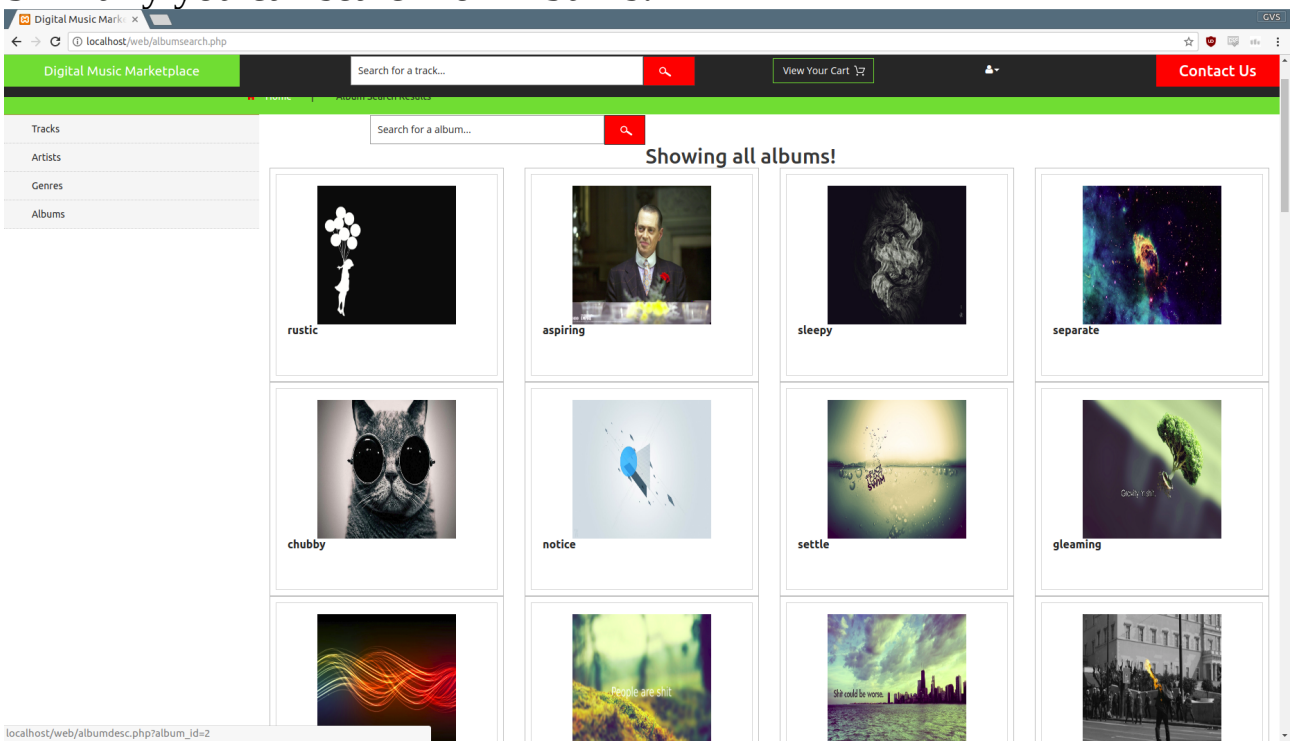


Search for a Track:-

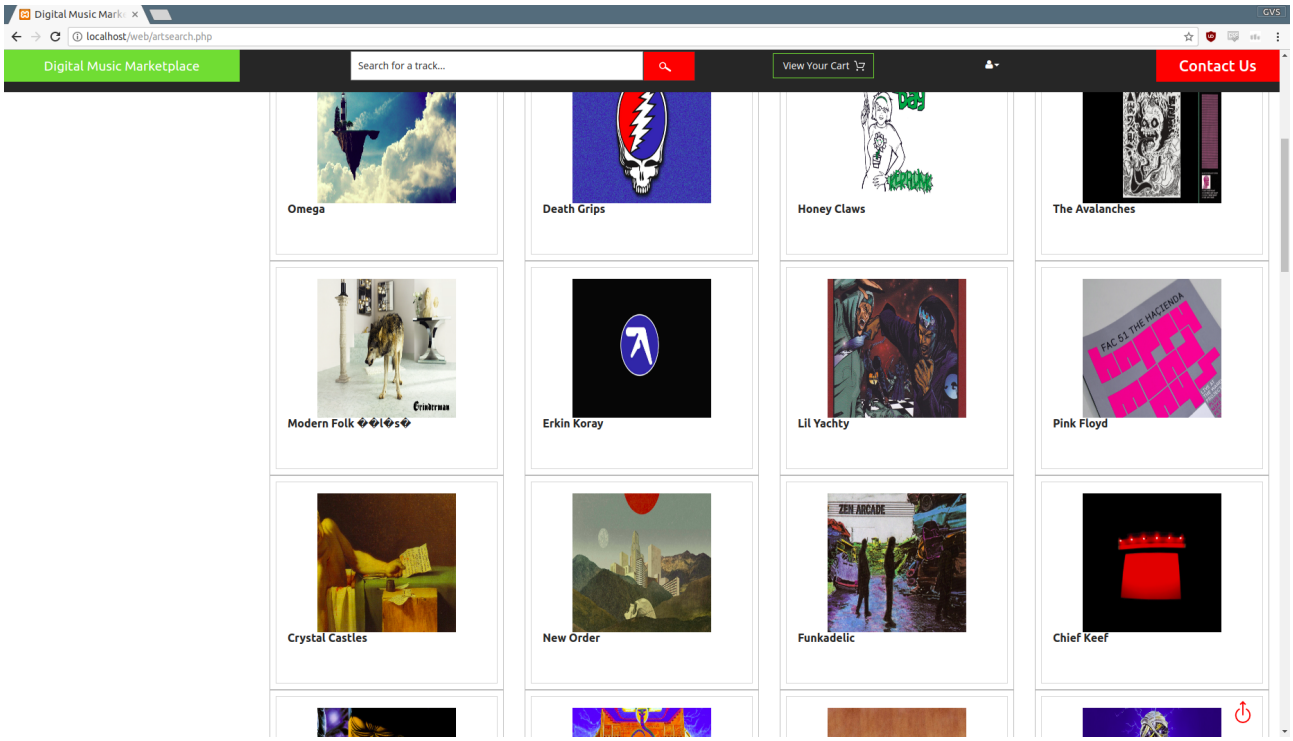
You can search for tracks by entering a search query in the search box or you can view all the tracks in a single page by pressing the ‘Tracks’ menu on the left.



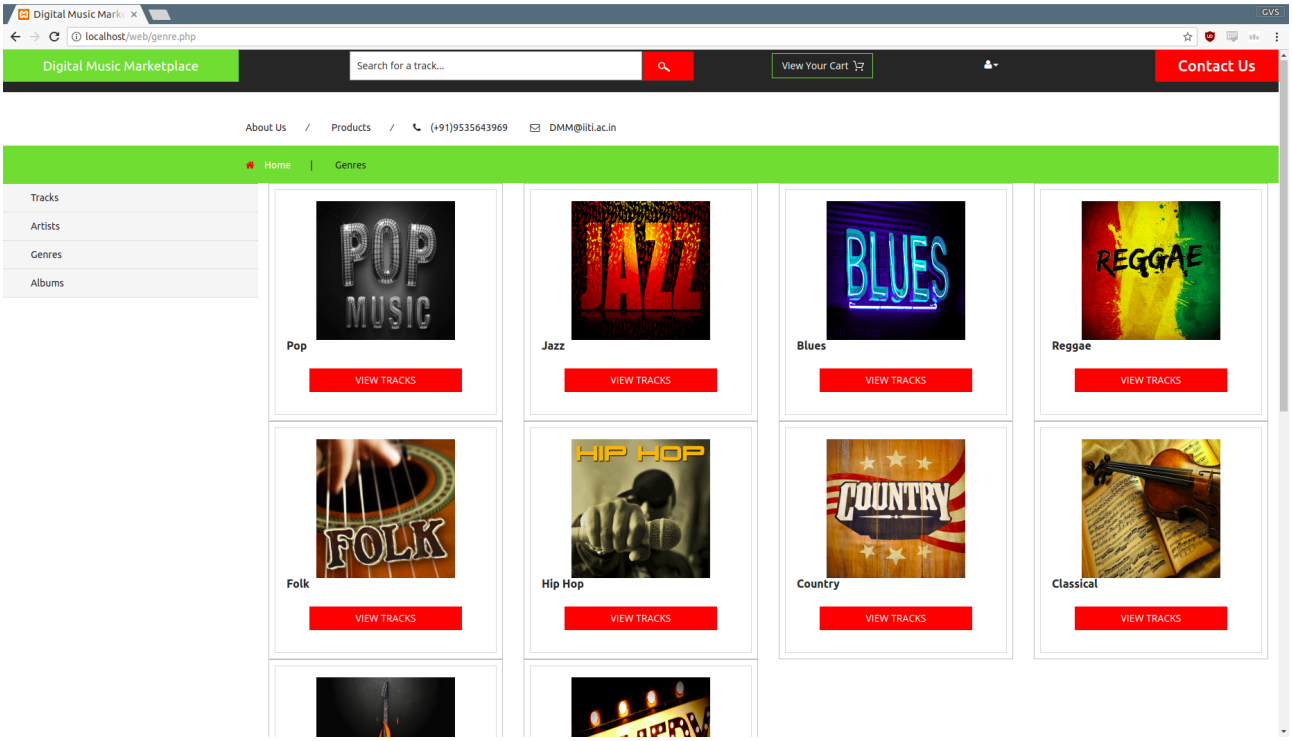
Similarly you can search for **Albums:**



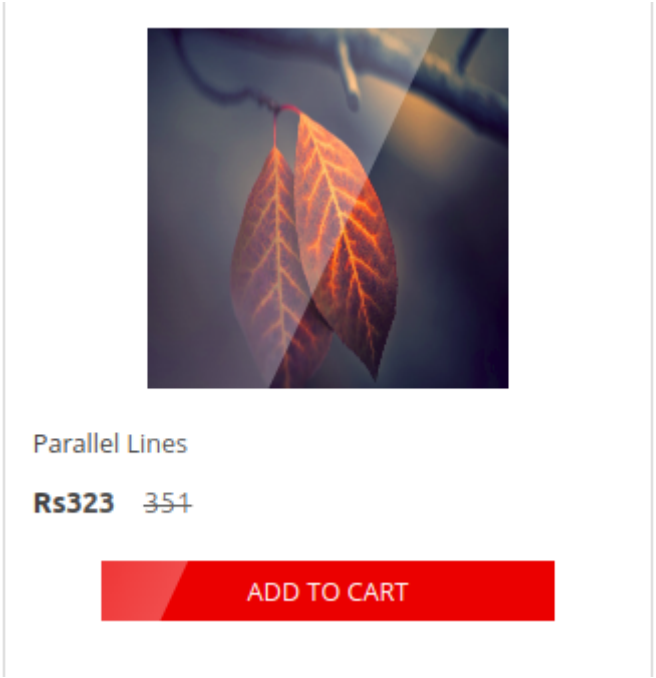
Artists:



Genre:



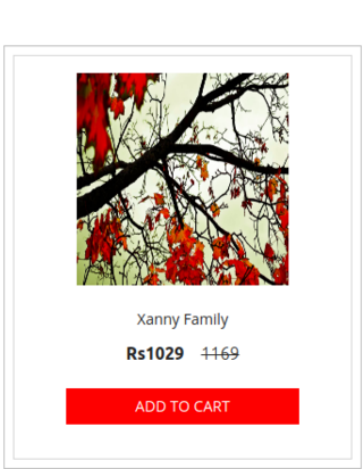
Adding Track to Cart:

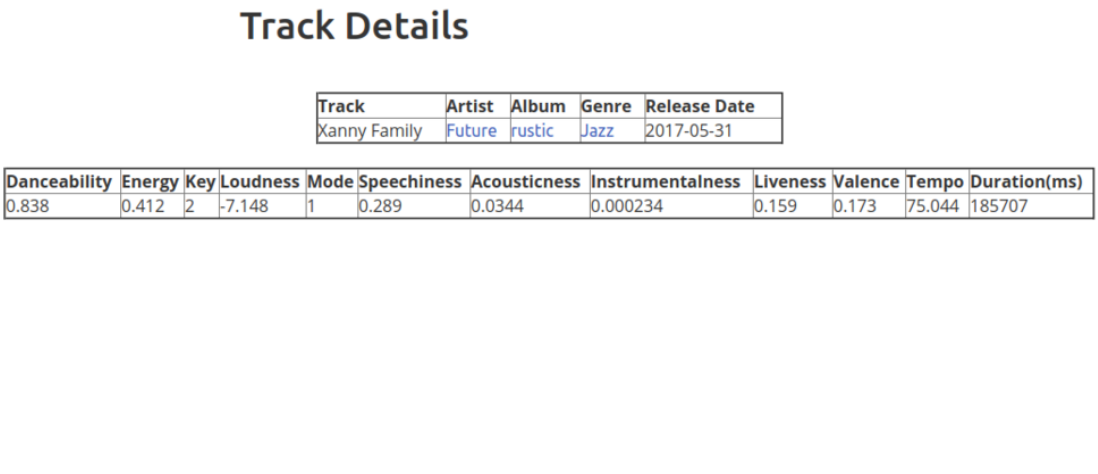


Once you select a track you can either press the ‘ADD TO CART’ button to directly add the item to your cart ot you can click the image to view more details about the track.

Track Details:

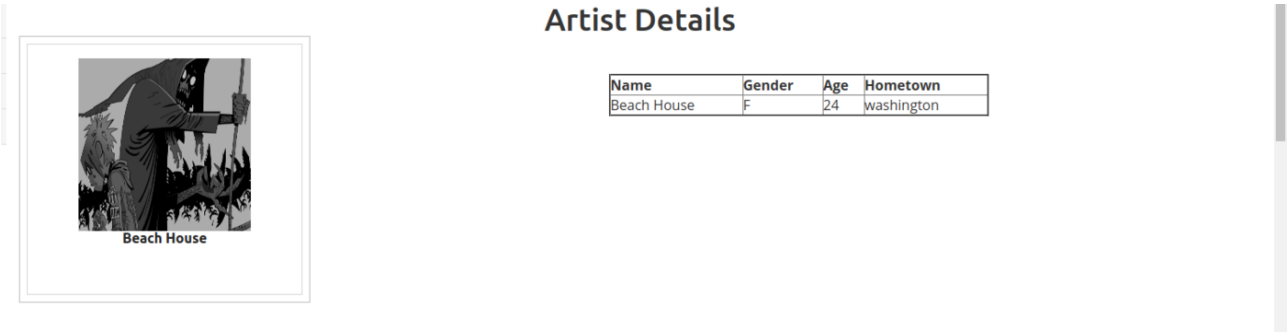
In this page, you can view all the track details to decide if you want to purchase it.



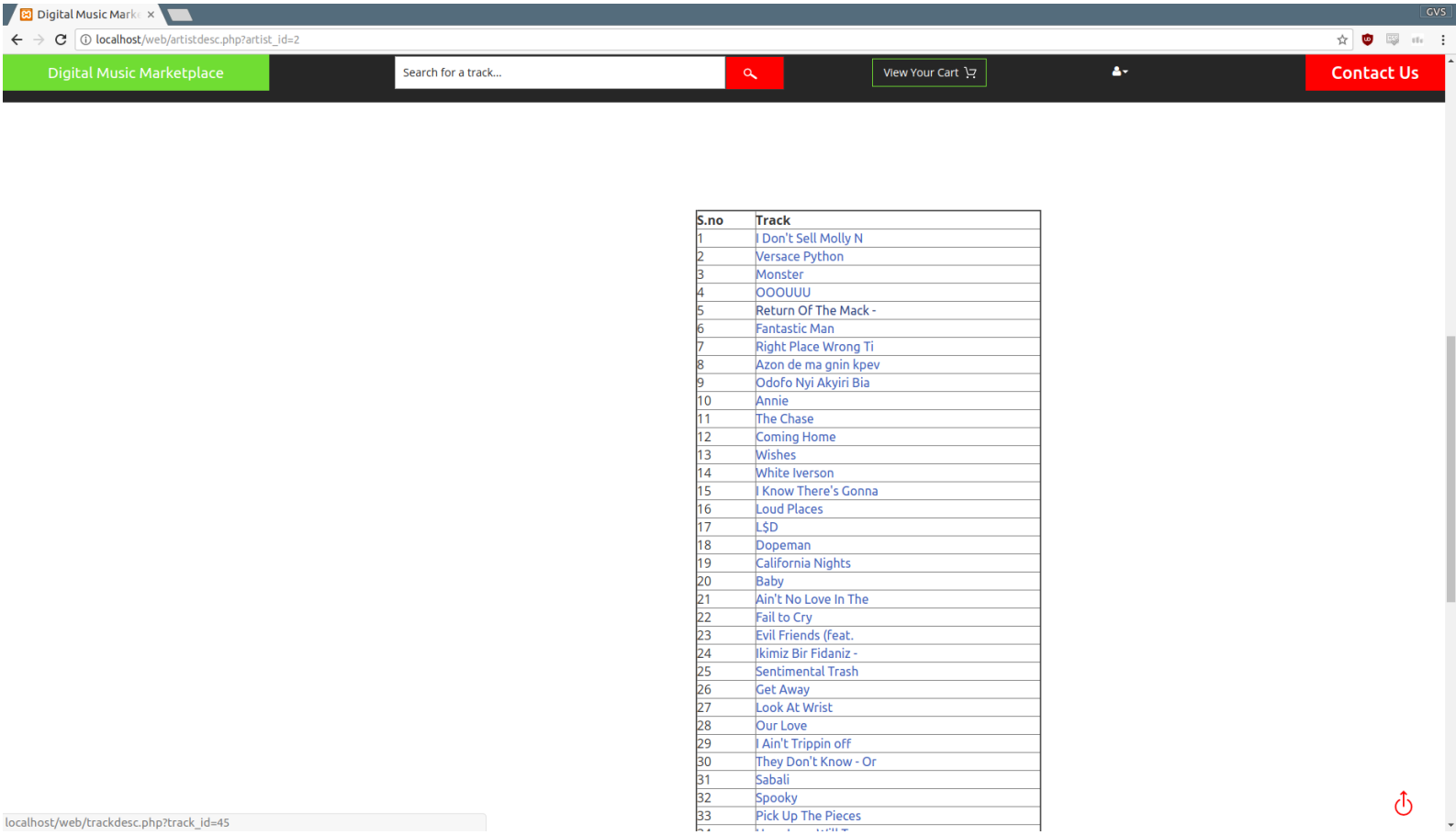


Artist Details:

By clicking on a artist, you can view his details.

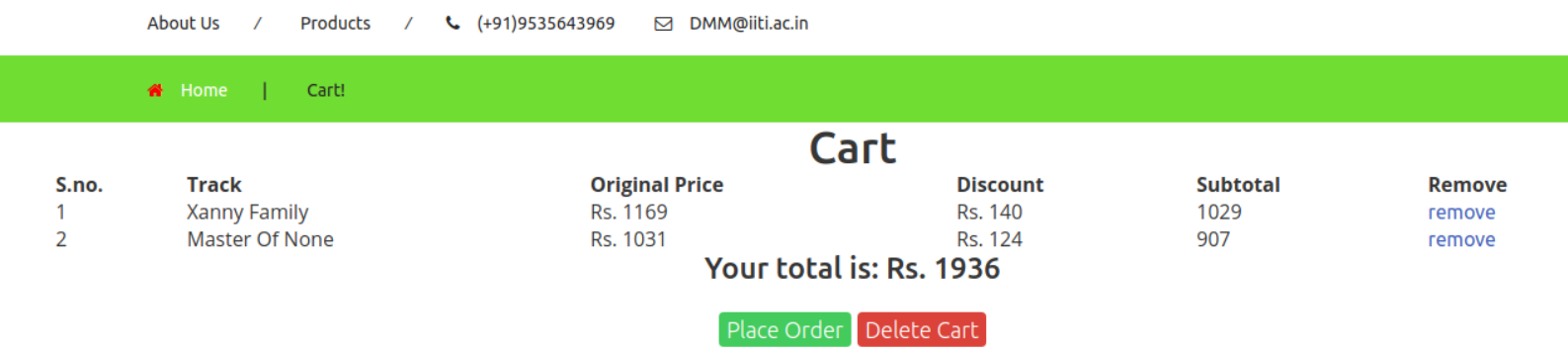


You can also view all the songs he has composed .



Cart:

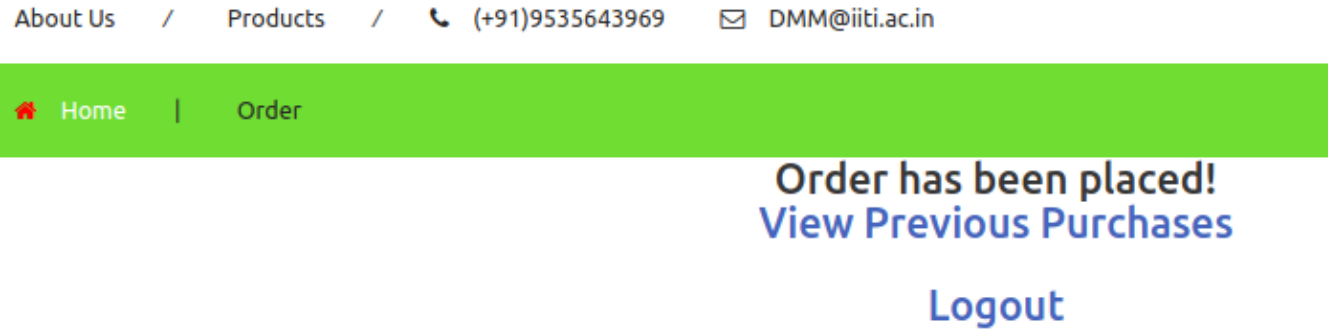
The cart contains all the tracks which the customer has added as well as options to remove tracks, place order and delete cart.



Place Order:

Once the customer clicks ‘Place Order’ button, a page containing the order summary is first displayed.

Upon clicking ‘Place Order’ , the order is finalized and recorded in the database.



Purchase History:

The customer can also view his purchase history by clicking on the ‘View Previous Purchases’ tab.

Contact Us:

If the customer has any problems, he may visit the ‘Contact Us’ page and drop a email.