

## Section 1: Metadata

*to be filled by the student*

### 1.1. Project Information to be filled by the student

Title: Spotify	
Section: L5	Instructor: Dr. Umer Tariq

### 1.2. Student(s) Information

Name: Abeer Masroor	ID: am08734
Section: L5	Batch: 2027

Name: Munir Tariq	ID: mm10067
Section: L5	Batch: 2028

Name: Ali Mahmoodi	ID: am09623
Section: L5	Batch: 2028

**Submission guideline:** Save your project proposal as a pdf file and rename as Section<Number>-Group<Number>.pdf



## Section 2: The Project

*to be filled by the student*

**1. Project Description:** *Please provide a brief introduction of the project including its scope.*

A toy version of Spotify.



## 2. Functional Requirements

*This section describes each function/feature provided by your system.*

### **User Types:**

**User A:** Listener

**User B:** Uploader

### **Functional Requirement 1: Register and Login**

Listeners and Uploaders can create an account, log in, and access their respective dashboards.

### **Functional Requirement 2: Upload and Monetize Songs**

Uploaders can upload new songs and choose whether they are free or paid.

### **Functional Requirement 3: View Song Analytics**

Uploaders can view total plays and likes for all their uploaded songs.

### **Functional Requirement 4: Search and Play Songs**

Listeners can search for creators, view their songs, play free tracks while updating view and like counts.

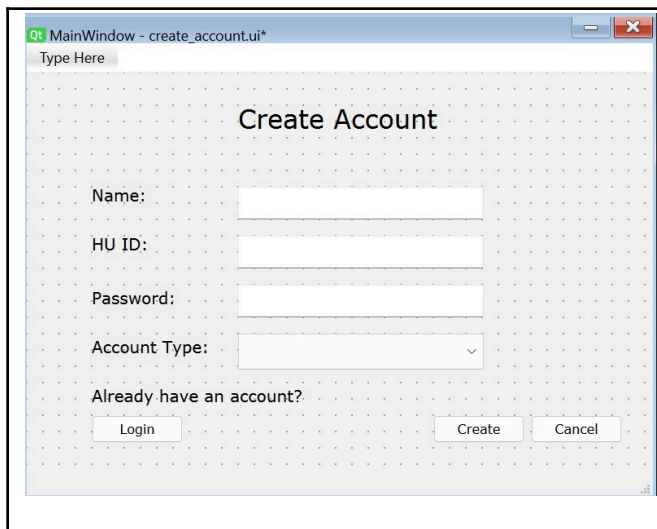
### **Functional Requirement 5: Purchase Paid Songs**

Listeners can purchase paid songs, after which playback is enabled and the purchase is recorded.

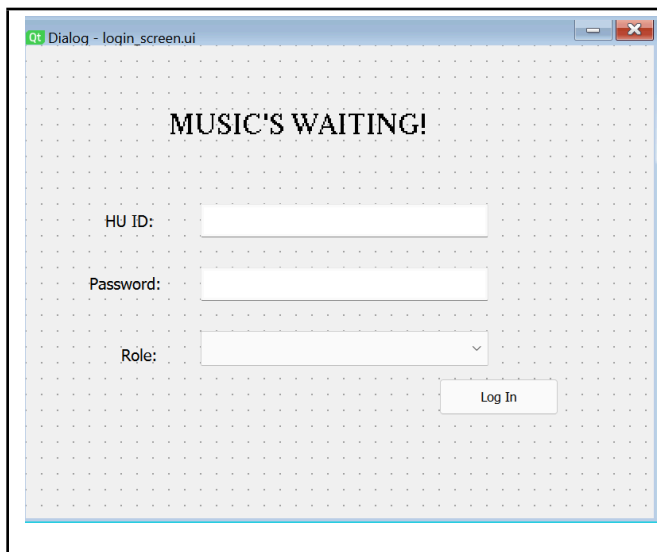
**3. UI Screens:** Provide images of all application screens, showcasing clear input and corresponding outputs. Ensure each image includes a concise caption explaining user action and expected/observed output. You can create these screens using Qt Designer.

## Functional Requirement 1: Register and Login

The system shall allow Listeners and Uploaders to create an account and login



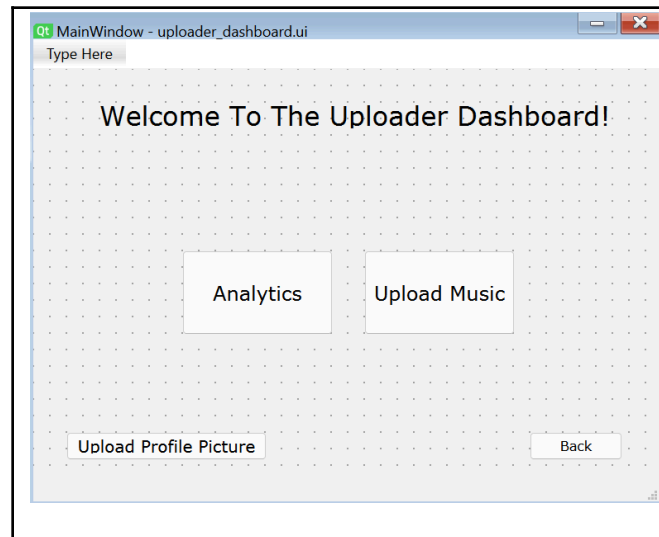
UI Screen 1.A: Create Account screen



UI Screen 1.B: Log in screen

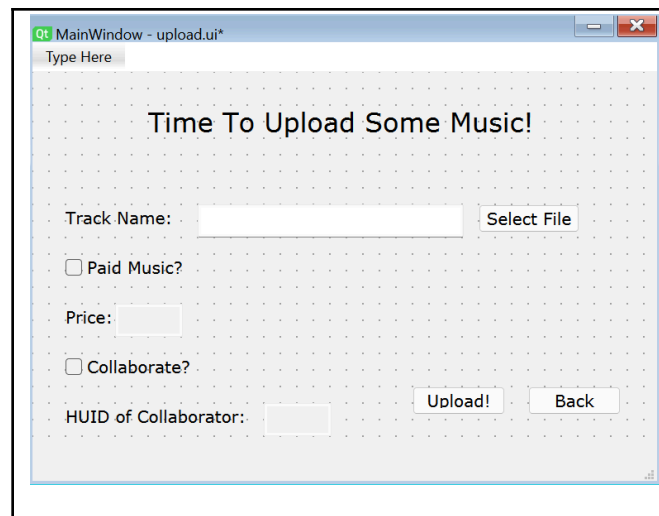
## Functional Requirement 2: Upload and Monetize songs

The system shall allow Uploaders to upload a new song and monetize it if they desire



UI Screen 2.A: Uploader dashboard

The above screen comes once the user logs in as an Uploader



UI Screen 2.B: Music upload screen

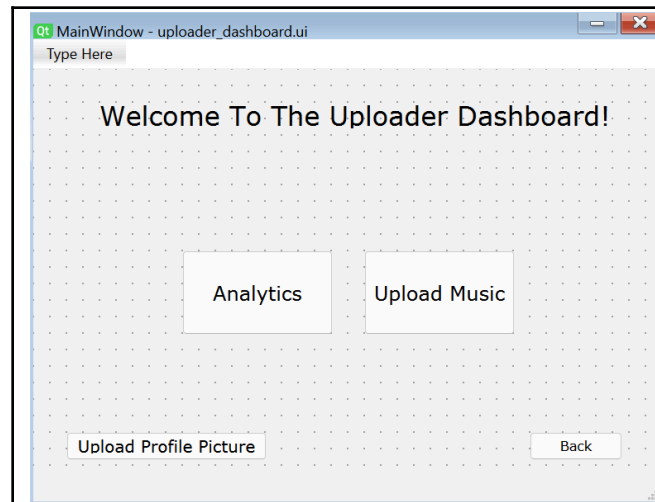
The above screen comes when you press the Upload Music button on Uploader Dashboard

### Functional Requirement 3: View song analytics

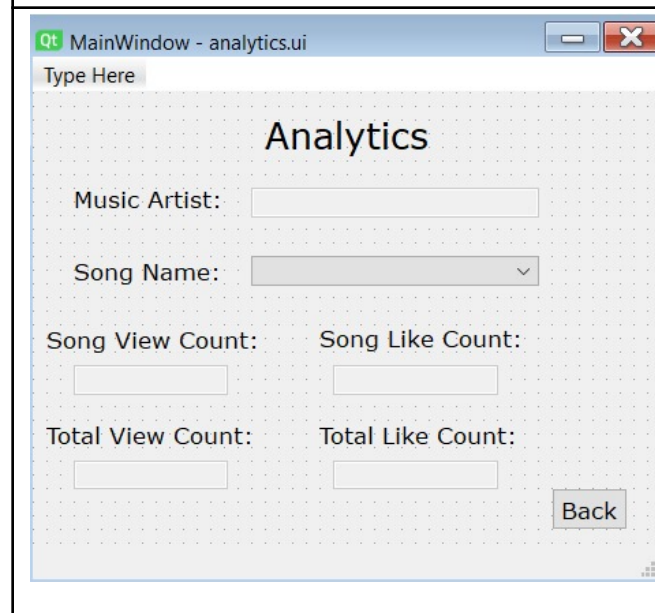
The system shall allow the Uploader to view analytics for their song/s



## CS355 – Database Project Submission Template



UI Screen 3.A: Uploader dashboard



UI Screen 3.B: Analytics screen

The above screen comes when you press Analytics button on Uploader Dashboard

### Functional Requirement 4: Search and play songs

The system shall allow the Listener to search for songs from the artist/genre of their choice and listen to them



Choose A Track!

Artist Name:

Genre:

Song:

Purchased Songs:

Log out Like Play Purchase

UI Screen 4.A: Search & Play song screen

The above screen comes when you login in as a Listener from the Login screen 1.B.

## Functional Requirement 5: Purchase Paid songs.

The system shall allow Listeners to purchase paid songs on the application

Purchase Song Screen

Artist Name

Song Name

Price

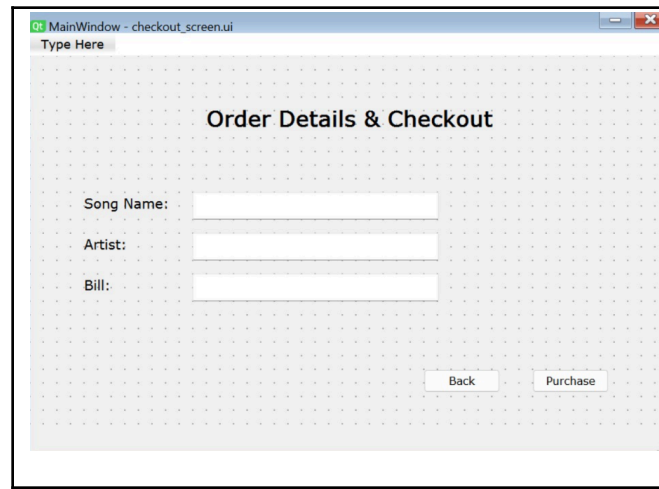
Credit Card Number

Expiry Date

Back Proceed to checkout

UI Screen 5.A: Purchase song screen

The above screen comes when the Listener presses the Purchase button on screen 4.A

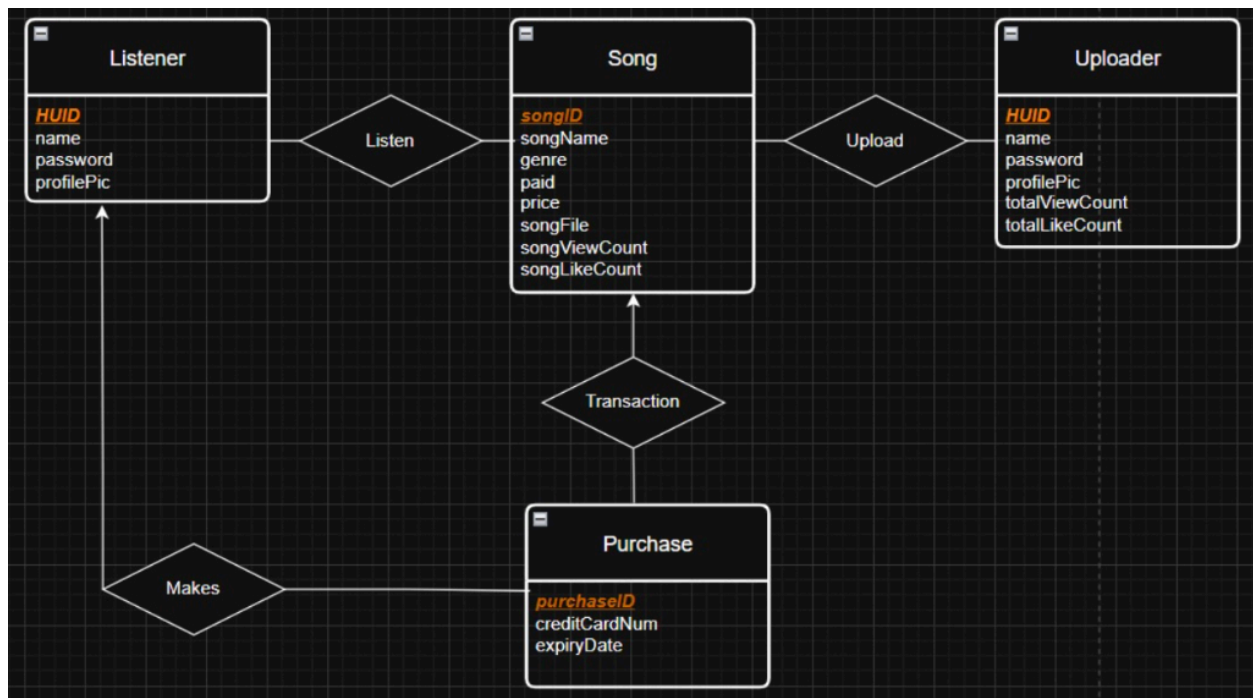


UI Screen 5.B: Order Details and Checkout screen

The above screen comes when the Listener presses the Proceed To Checkout button on screen 5.A

Listener returns to Search and Play song screen 4.A once purchase is complete

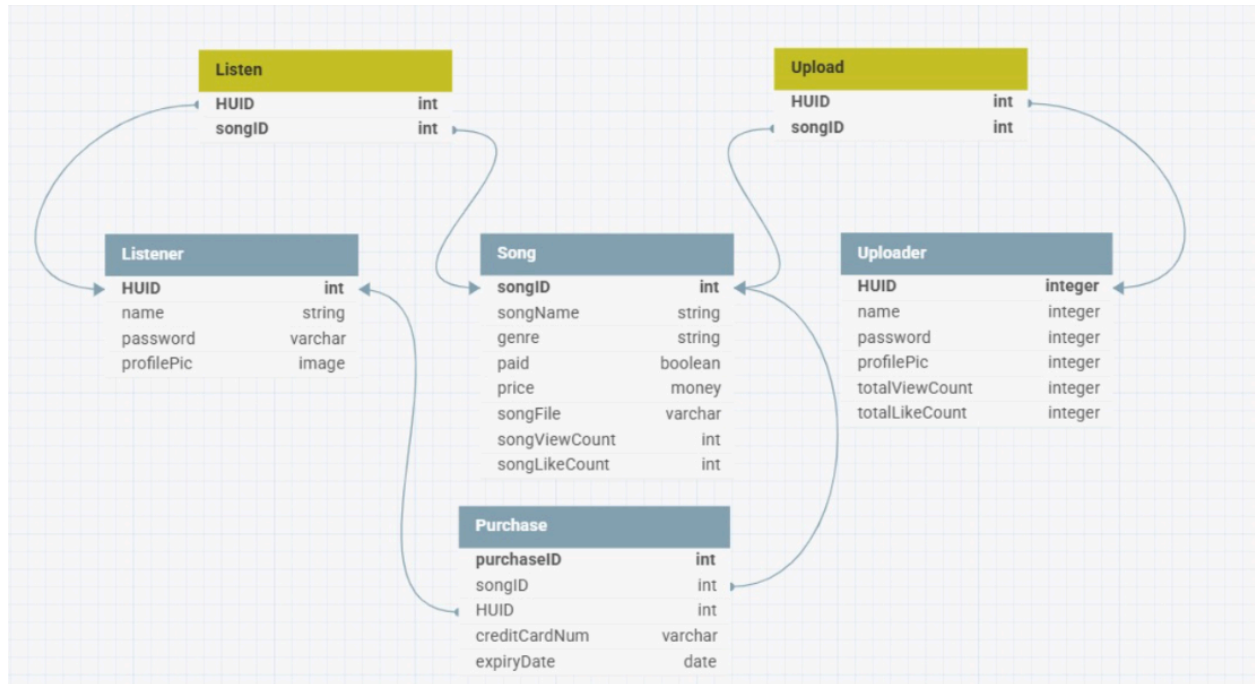
## 4. ERD







## 5. Relation Schema





### 6. SQL

*(Note: '?' in the queries below will be replaced by actual values later on, its a placeholder for now.)*

#### Functional Requirement 1: Register and Login

The system shall allow Listeners and Uploaders to create accounts and login

UI Screen 1.A: Create account screen

#### ***For Listener:***

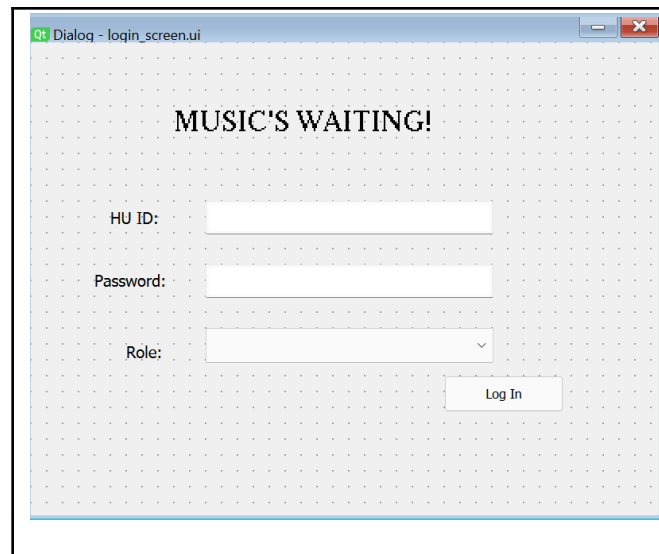
```
INSERT INTO Listener (HUID, name, password, profilePic)
```

```
VALUES (?, ?, ?, NULL);
```

#### ***For Uploader:***

```
INSERT INTO Uploader (HUID, name, password, profilePic, totalViewCount, totalLikeCount)
```

```
VALUES (?, ?, ?, NULL, 0, 0);
```



UI Screen 1.B: Log in screen

***Check Listener:***

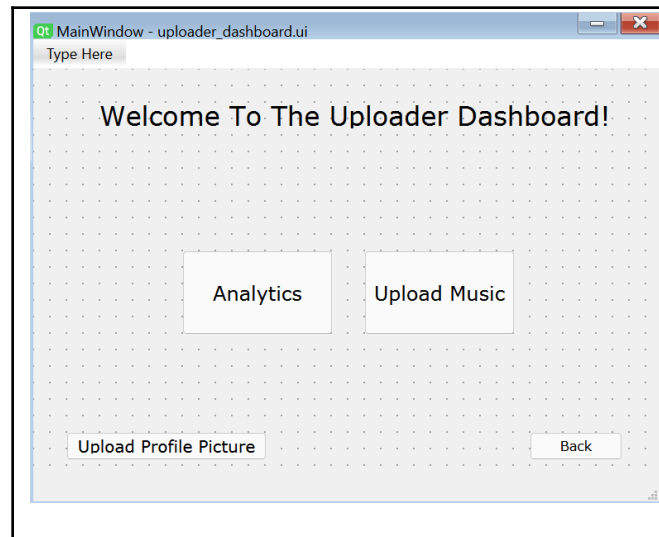
SELECT HUID, name FROM Listener WHERE HUID = ? AND password = ?;

***Check Uploader:***

SELECT HUID, name FROM Uploader WHERE HUID = ? AND password = ?;

**Functional Requirement 2: Upload and monetize songs**

The system shall allow an uploader to upload a song and monetize it if they desire



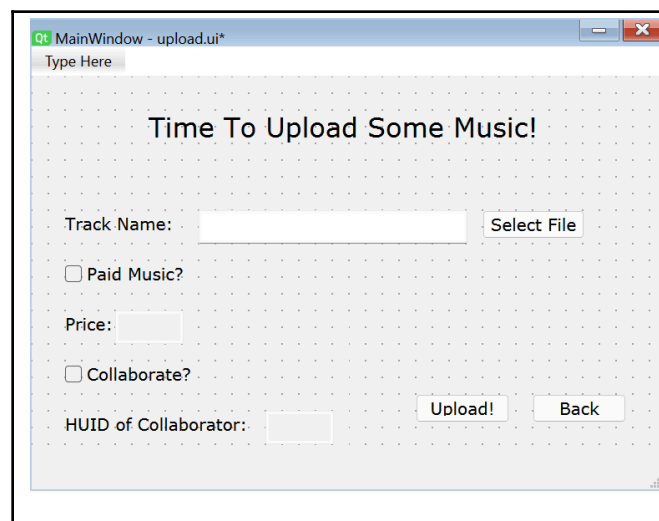
UI Screen 2.A: Uploader dashboard

***Insert song:***

```
INSERT INTO Song (songName, genre, paid, price, songFile, songViewCount, songLikeCount)
VALUES (?, ?, ?, ?, ?, 0, 0);
```

***Create upload relationship:***

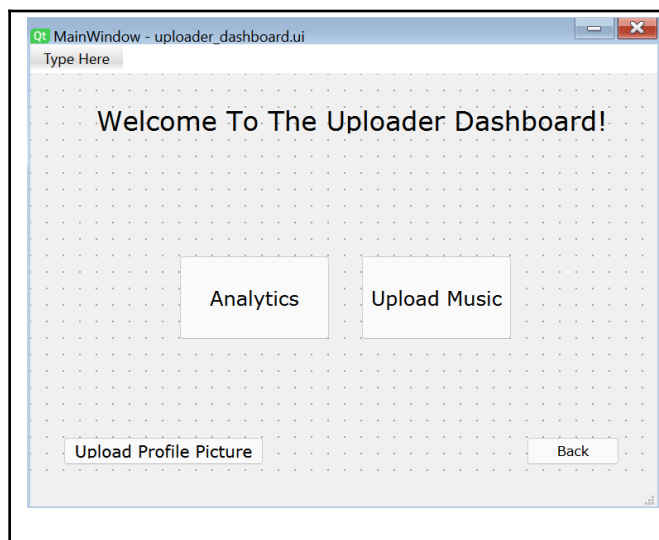
```
INSERT INTO Upload (HUID, songID) VALUES (?, LAST_INSERT_ID());
```



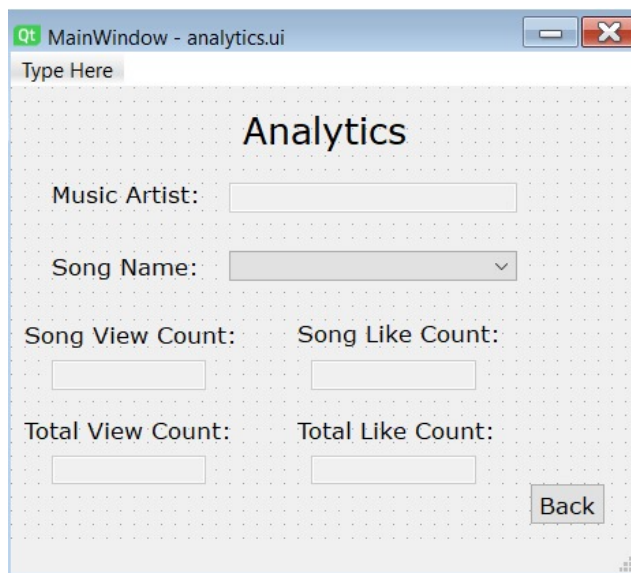
UI Screen 2.B: Upload song screen

## Functional Requirement 3: View song analytics as an Uploader

The system shall allow an Uploader to view analytics for their song/s



UI Screen 3.A: Uploader dashboard



UI Screen 3.B: Analytics screen

**Total views and likes across all songs:**

SELECT

SUM(s.songViewCount) as totalViews,



```
SUM(s.songLikeCount) as totalLikes  
  
FROM Song s  
  
JOIN Upload u ON s.songID = u.songID  
  
WHERE u.HUID = ?;
```

***Per-song breakdown:***

```
SELECT  
  
    s.songID, s.songName, s.songViewCount, s.songLikeCount  
  
FROM Song s  
  
JOIN Upload u ON s.songID = u.songID  
  
WHERE u.HUID = ?;
```

Functional Requirement 4: Search and play songs

The system shall allow listeners to search for songs and play them

Dialog - songselection\_screen.ui\*

Choose A Track!

Artist Name:

Genre:

Song:

Purchased Songs:

Log out Like Play Purchase

UI Screen 4.A: Search and play song screen



***Get all paid songs:***

```
SELECT
    s.songID,
    s.songName,
    s.genre,
    s.songFile,
    u.name as artistName
FROM Purchase p
JOIN Song s ON p.songID = s.songID
JOIN Upload up ON s.songID = up.songID
JOIN Uploader u ON up.HUID = u.HUID
WHERE p.HUID = ?
```

***Search by Artist:***

```
SELECT s.songID, s.songName, s.genre, u.name as artistName, s.paid, s.price
FROM Uploader
WHERE name LIKE CONCAT('%', ?, '%');
```

***Search by genre:***

```
SELECT s.songID, s.songName, s.genre, u.name as artistName, s.paid, s.price
FROM Song s
JOIN Upload up ON s.songID = up.songID
JOIN Uploader u ON up.HUID = u.HUID
WHERE s.genre LIKE CONCAT('%', ?, '%');
```

***Search by song name:***

```
SELECT s.songID, s.songName, s.genre, u.name as artistName, s.paid, s.price
```



```
FROM Song s
```

```
JOIN Upload up ON s.songID = up.songID
```

```
JOIN Uploader u ON up.HUID = u.HUID
```

```
WHERE s.songName LIKE CONCAT('%', ?, '%');
```

***Get creator info:***

```
SELECT name, profilePic FROM Uploader WHERE HUID = ?;
```

***Increment view count:***

```
UPDATE Song SET songViewCount = songViewCount + 1 WHERE songID = ?;
```

***If liked, increment like count (separate query when like button clicked):***

```
UPDATE Song SET songLikeCount = songLikeCount + 1 WHERE songID = ?;
```

### Functional Requirement 5: Purchase song

The system shall allow a listener to purchase a paid song

Dialog - purchaseScreen.ui

Purchase Song Screen

Artist Name

Song Name

Price

Credit Card Number

Expiry Date

Back Proceed to checkout

UI Screen 5.A: Purchase song screen





The screenshot shows a Java Swing window titled "MainWindow - checkout\_screen.ui". Inside the window, there is a text area with a grid pattern. The title "Order Details & Checkout" is centered at the top. Below it, there are three input fields with labels "Song Name:", "Artist:", and "Bill:". At the bottom right of the grid, there are two buttons labeled "Back" and "Purchase".

UI Screen 5.B: Order details and checkout screen

***Making purchase:***

```
SELECT purchaseID FROM Purchase
WHERE songID = ? AND HUID = ?;
INSERT INTO Purchase (songID, HUID, creditCardNum, expiryDate)
VALUES (?, ?, ?, ?);
```

***Purchase confirmation:***

```
SELECT
    s.songName,
    u.name as artistName,
    s.price
FROM Song s
JOIN Upload up ON s.songID = up.songID
JOIN Uploader u ON up.HUID = u.HUID
WHERE s.songID = ?;
```

P.T.O →

***Work division:***

**Abeer:**

- Register and Login
- Search and Play Songs

**Munir:**

- Upload and Monetize Songs
- View Song Analytics

**Ali:**

- Purchase Paid Songs
- Assist with integration and database management



### 7. Final Demo Screenshots

#### Functional Requirement 1: Register and Login

The system shall allow Sellers and Buyers to register an account and login.



Demo Screenshot 1.A: Sign up screen



Demo Screenshot 1.B: Log in screen

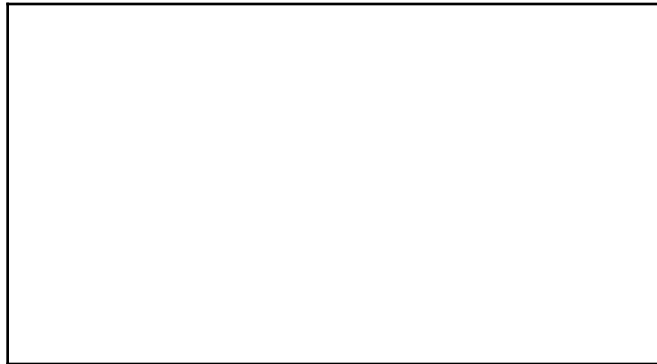
#### Functional Requirement 2: Update the Product List

The system shall allow a Seller to list a new product and modify any existing product.





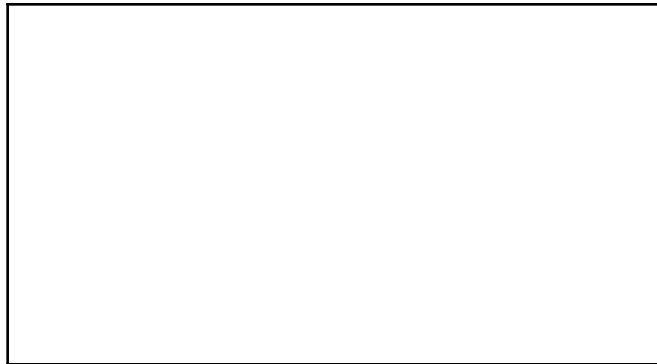
Demo Screenshot 2.A: Description



Demo Screenshot 2.B: Description

### Functional Requirement 3: Search and Purchase a Product

The system shall allow a Buyer to search for a product and place an order for it.



Demo Screenshot 3.A: Description

### Functional Requirement 4: Update the Order Status

The system shall allow a Seller to see the placed orders and update the status.





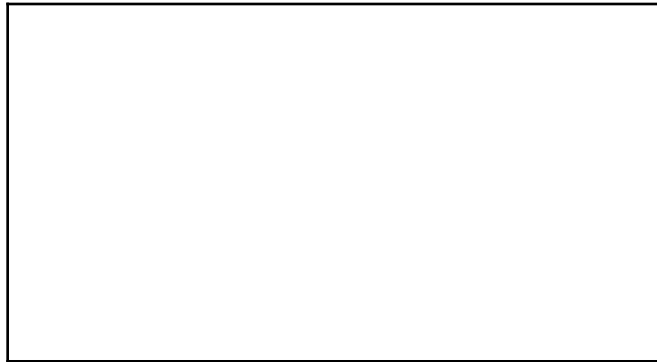
## CS355 – Database Project Submission Template



Demo Screenshot 4.A: Description

### Functional Requirement 5: Monitor the Order Status

The system shall allow a Buyer to monitor the status of past orders.



Demo Screenshot 5.A: Description



Demo Screenshot 5.B: Description