**CO303 DATABASE MANAGEMENT SYSTEMS LAB**

**A REPORT ON THE PROJECT ENTITLED**

**“GREY VIBRANT”**



**SUBMITTED BY**

**NAME1 – Shrinidhi Anil Varna** (171CO145)

**NAME2 - Sayan Biswas** (171CO141)

**V SEMESTER B-TECH**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA, SURATHKAL**

**2019-2020**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **CONTENT** | | **PAGE NUMBER** | |
| 1 | Abstract | | 3 | |
| 2 | Introduction | | 4 | |
| 3 | Database Design | | 5 | |
| 3.A | ER Diagram | | 5 |
| 3.B | Relational Schema | | 6 |
| 3.C | Normalized tables | | 7 |
| 4 | Modules | | 9 | |
| 5 | Tools | | 16 | |
| 6 | Results | | 19 | |

**1. ABSTRACT**

Grey Vibrant is a music streaming android application.

There are two prototypes User and Artist. Artists can upload their own songs and manage them. Users can create their own playlist, follow desired artists and listen to songs of their choice by adding them to a queue.

**2. INTRODUCTION**

Describe your idea here.

Describe what is the use of your project in 1- 2 pages.

Grey Vibrant is a music streaming android application that lets you play songs that you desire. It lets you follow artists of your choice and recommends you the songs of the followed artists. The remaining songs which come under unfollowed artists are also shown and can always be listened to. As an artist you can manage your album by adding a song, along with its description and view them in your personal album.

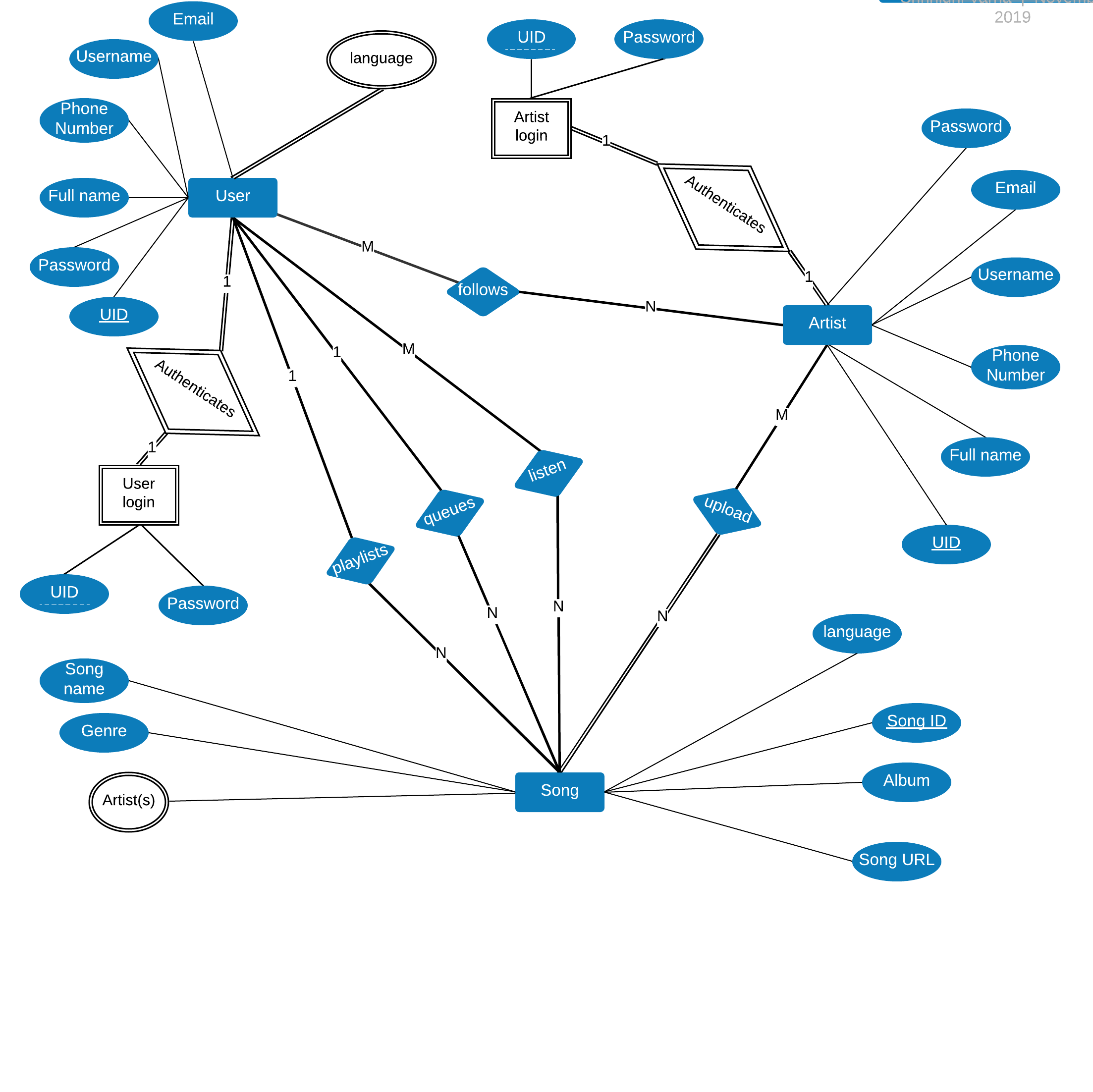
For a user, some special features that are provided include, adding a song to a personal playlist, adding it to a queue to listen later, and maintaining a history of the songs that you have heard before.

Apart from this a user can always unfollow an artist, delete a song from a playlist, queue or from song history. An artist can also delete a song from his/her personal album in case it was added by mistake or due to some other reason. A search functionality is also implemented to search a song on the basis of artist name, album, song name, language and genre.

**3. DATABASE DESIGN**

**3.A ER DIAGRAM**

Paste your ER Diagram here.



**3.B RELATIONAL SCHEMA**

Describe about all the tables.

*1. user\_registration (UID, fullname, phNo, username, email, password, playlist\_name)*

Maintains a record of all the users after their registration.

2. *user\_language (UID, language)*

Maintains a record of all the languages that a user wants to listen. (song language)

3. *user\_login (UID, username, password, foreign* *key (UID) references* *user\_registration(UID))*

A table that keeps the information required to authenticate a user.

4. artist\_registration(AID, artistname,email, password, phNo, fullname)

Maintains a record of all the artist after their registration.

5. *artist\_login (AID, artistname, password, foreign* *key (UID) references* *artist\_registration(AID))*

A table that keeps the information required to authenticate an artist.

6. *song (SID, songname, songurl, genre, language, album)*

Maintains a record of all the songs that have been uploaded by various artists registered in the application.

7. *song\_artists (SID, AID)*

A table that links a song and its corresponding artist.

8. *follow\_artists (UID, AID)*

A table that provides a link between a user and the artist(s) he/she follows

9. *listens (UID, SID)*

A table that maintains all the song(s) that a user has already listened to.

10. *playlist (UID, SID)*

A table that maintains all the songs that have been added to the playlist by the user.

11. *queue (UID, SID)*  
Contains all the songs that are currently queued for streaming for the user by the music player.

**3.C NORMALIZED TABLES**

Describe about normalization briefly.

Describe your 3NF tables here.

Database normalizationis a database schema design technique, by which an existing schema is modified to minimize redundancy and dependency of data. Normalizationsplit a large table into smaller tables and define relationships between them to increases the clarity in organizing data. It is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

**3NF tables of the database schema mentioned earlier**:

**1. user\_registration**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| UID | fullname | phNo | username | email | password | playlist\_name |

**2. user\_language**

|  |  |
| --- | --- |
| UID | language |

**3. user\_login**

|  |  |  |
| --- | --- | --- |
| UID | username | password |

**4. artist\_registration**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AID | artistname | email | password | phNo | fullname |

**5. artist\_login**

|  |  |  |
| --- | --- | --- |
| AID | artistname | password |

**6. song**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SID | songname | songurl | genre | language | album |

**7. song\_artists**

|  |  |
| --- | --- |
| SID | AID |

**8. follow\_artists**

|  |  |
| --- | --- |
| UID | AID |

**9. listens**

|  |  |
| --- | --- |
| SID | UID |

**10. playlist**

|  |  |
| --- | --- |
| UID | SID |

**11. queue**

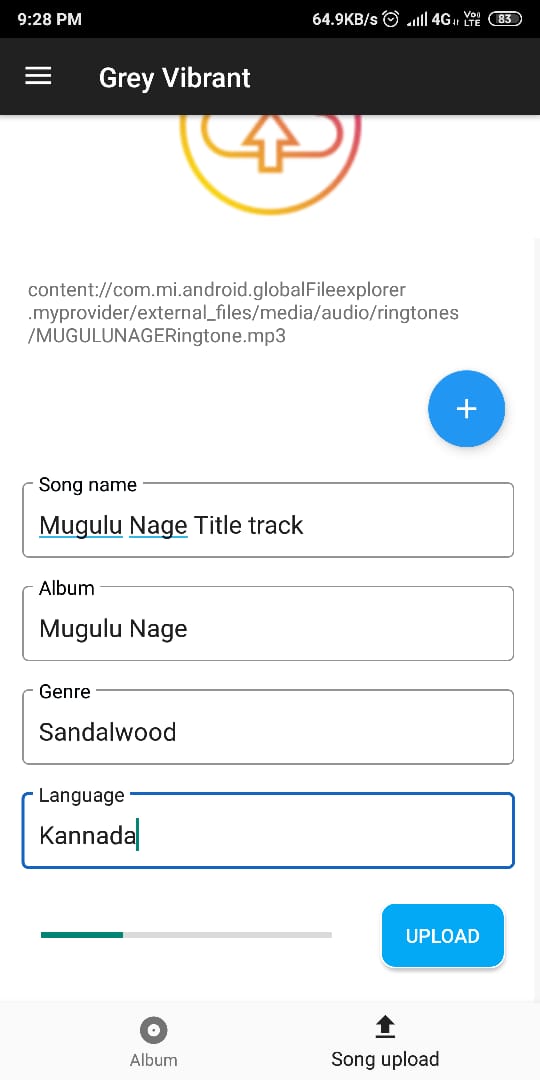
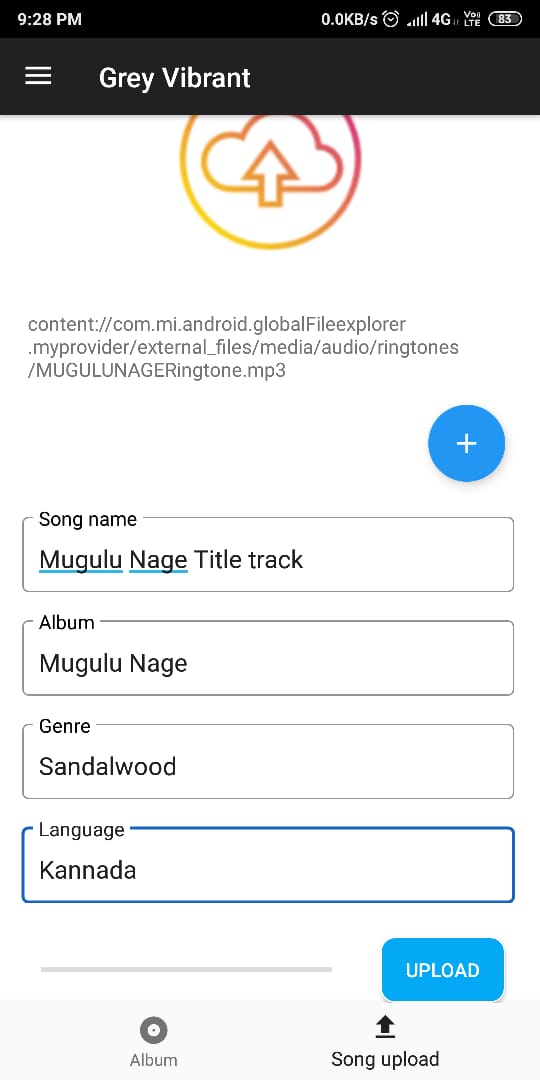
|  |  |
| --- | --- |
| UID | SID |

**4. MODULES**

Describe all the modules present in your project with screenshot.

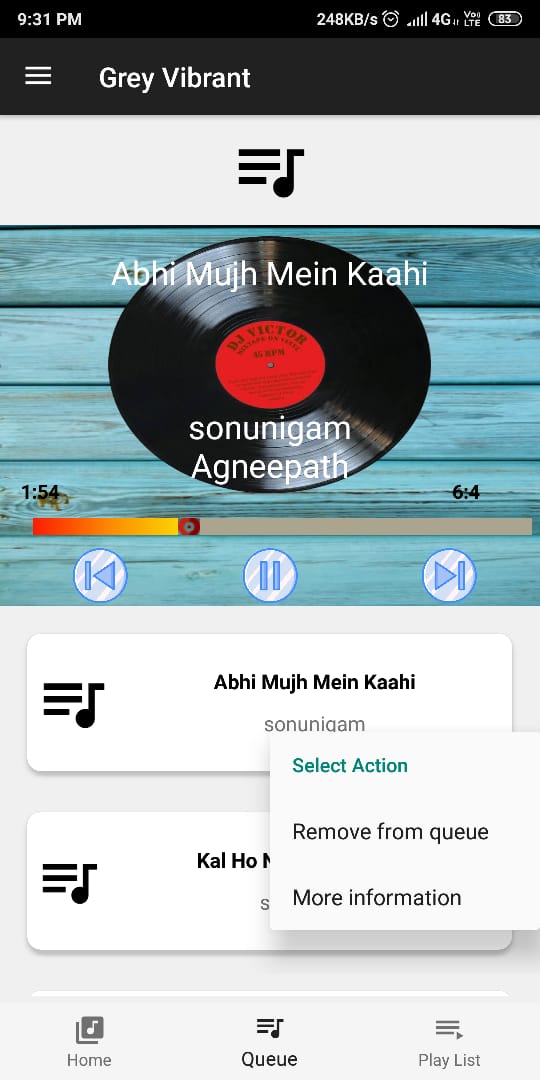
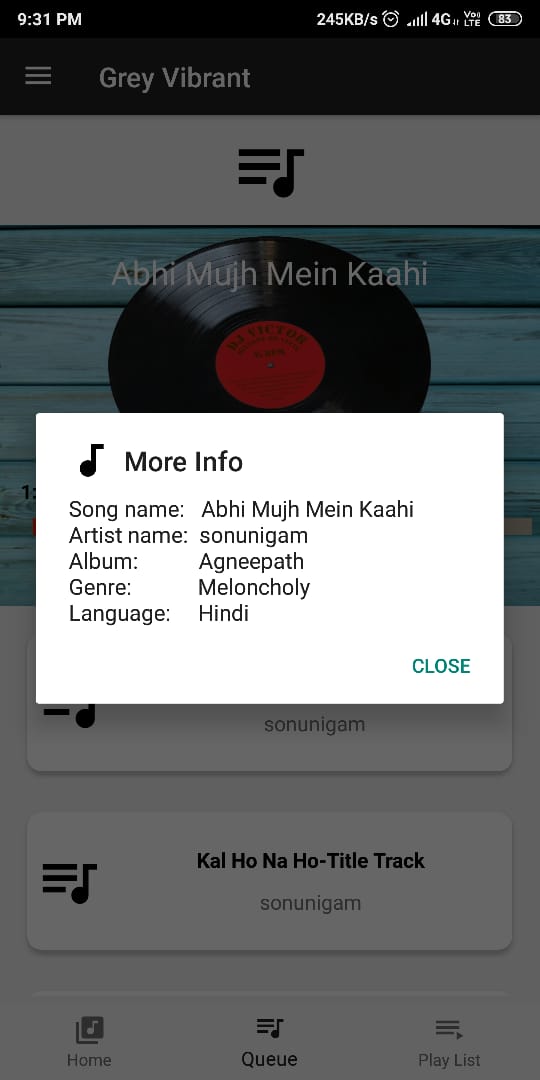
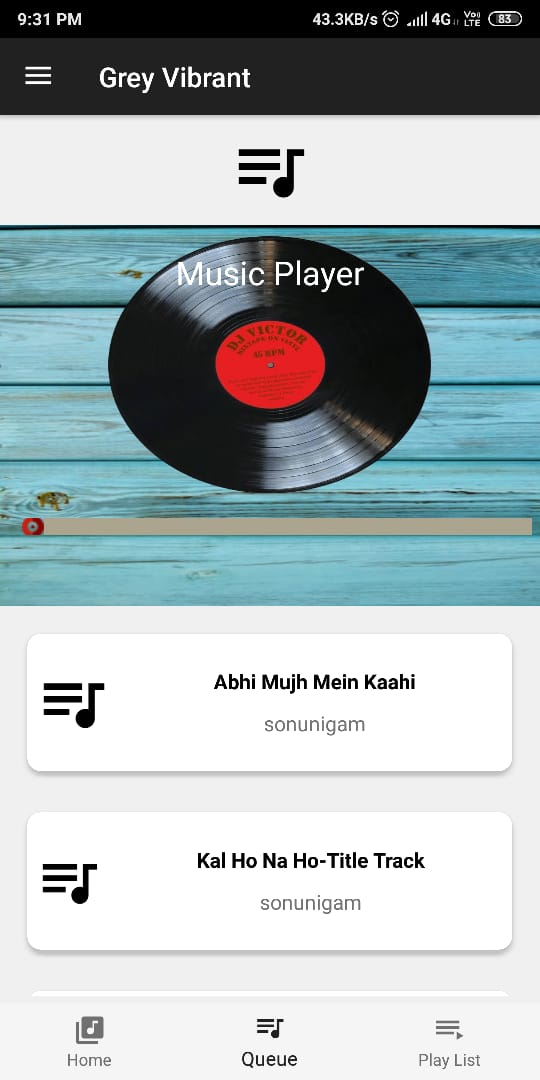
The different modules present in the project are:

1. Upload: In this module, an artist can upload a song (mp3 file) along with its language, genre and songname to the database. A song is uploaded to a storage bucket and its URL is stored in the **song** table of the database. This URL is used for streaming music whenever a user listens to the song.



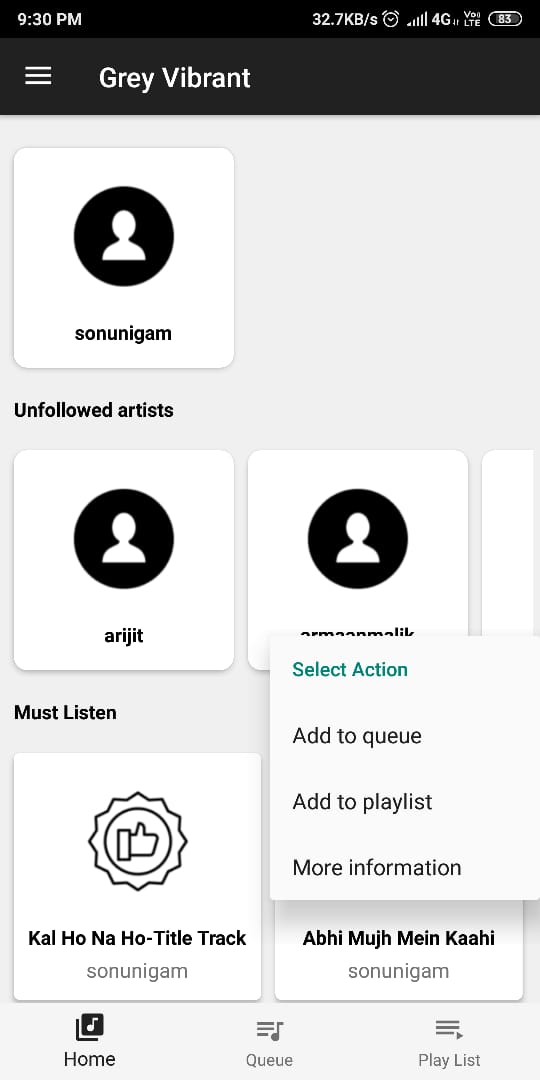
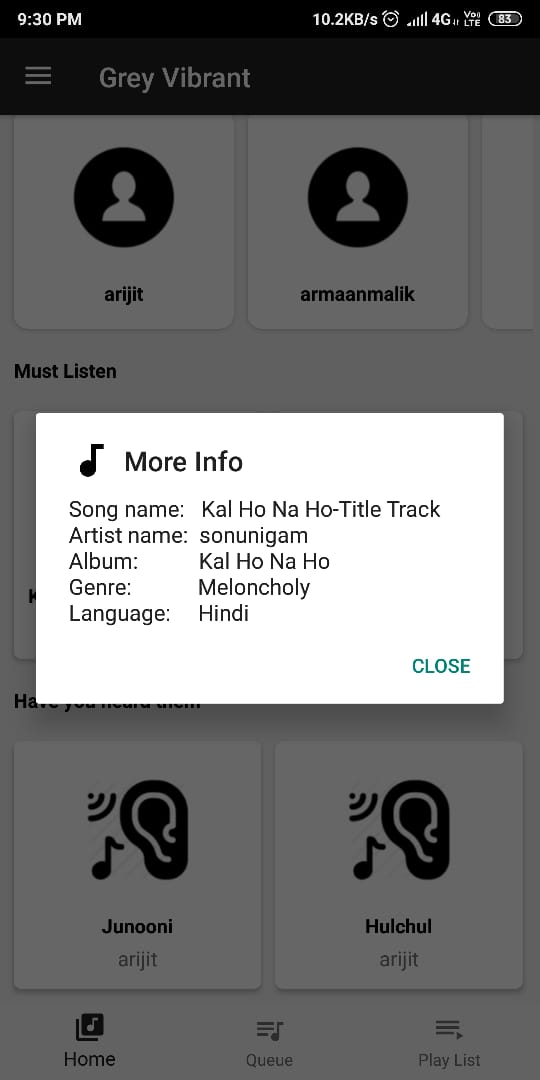
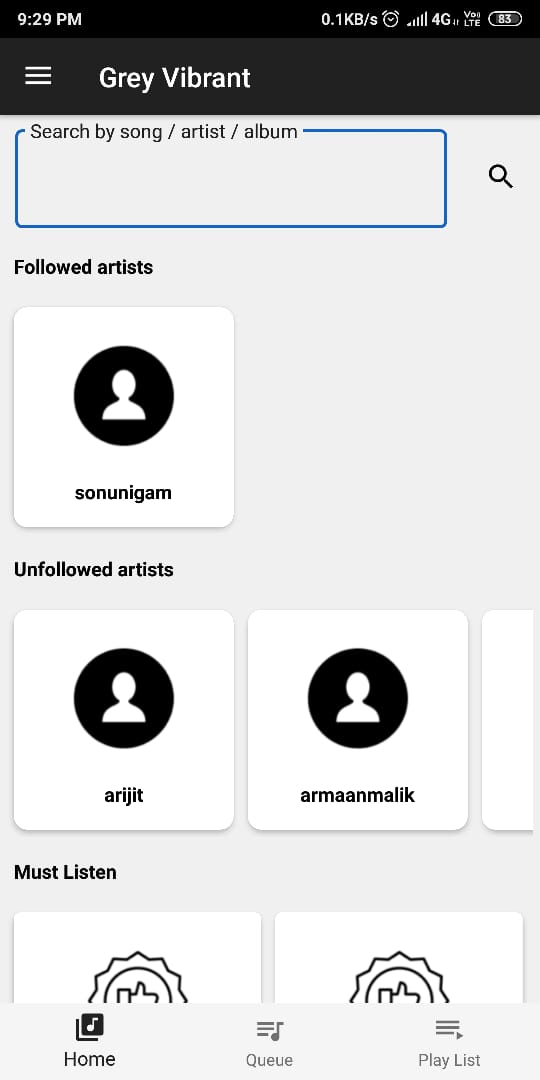
**1.a:** Filling in details **1.b:** Uploading song

2. Music Stream: This module contains a music player that can play a list of songs back to back by streaming it live from the song URLs stored in the database. The functionalities available in this module are: play next, play previous, start, pause, pick a song from the queue for playing. This module also has a queue of songs shown to the user which were selected for streaming. These songs can be picked at random and played, played sequentially, deleted from the queue, more information can be provided about the songs in queue.

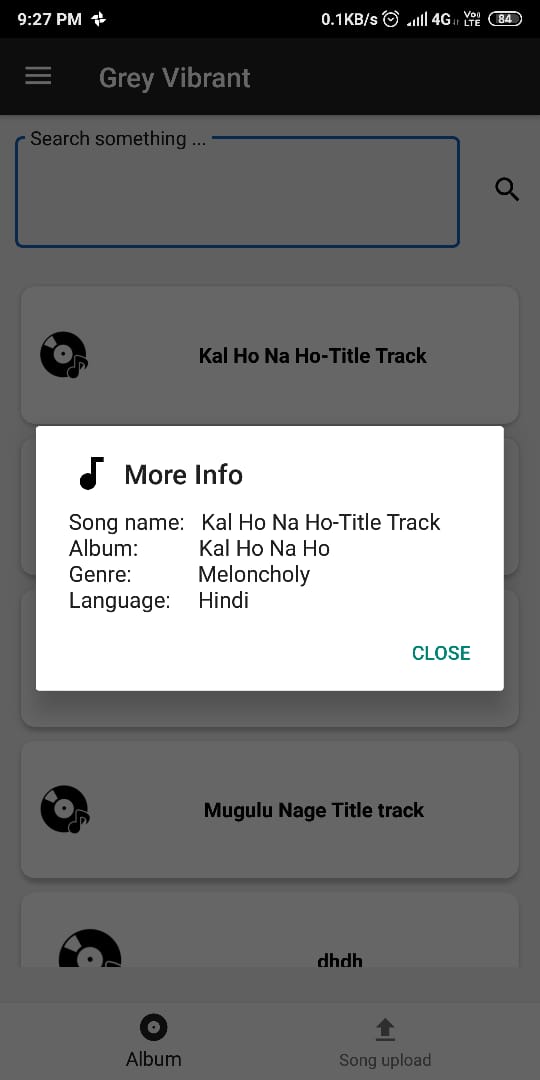
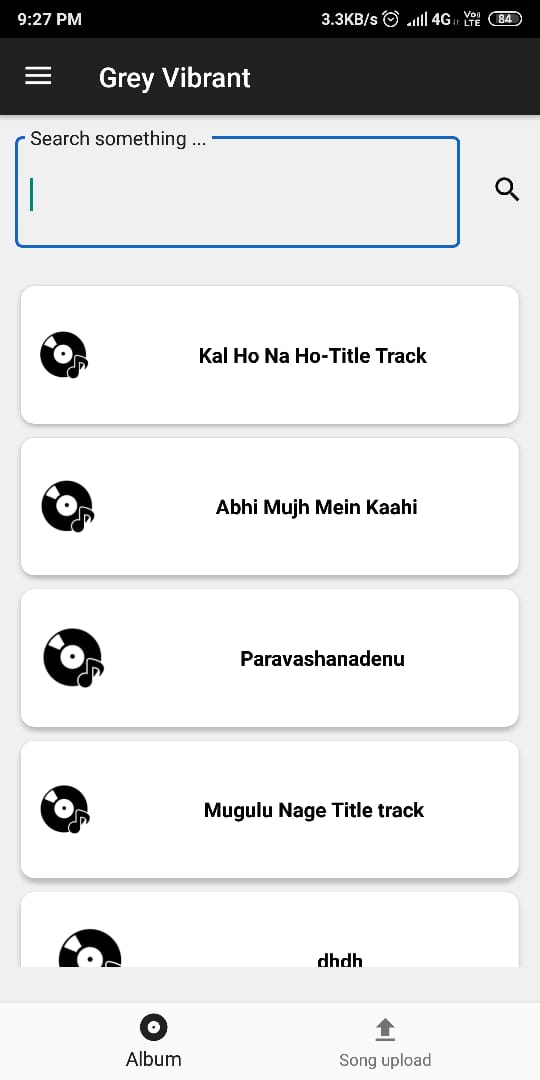


**2.a:** Music player with a queue of songs **2.b**: Providing song details **2.c:** Streaming a song

3. Home page: This page has a list of artists a user follows, unfollows, recommended songs, songs that can be still listened along with a search functionality provided to filter the results. The search works on the basis of artistname, songname, genre, and language. The artists who are followed can be unfollowed and the unfollowed artists can be followed just by clicking on their named cards. The songs that are displayed can be held pressed for a longer time to use few more functionalities of the module. They are: More information about the song, add song to playlist, add song to queue for playing.



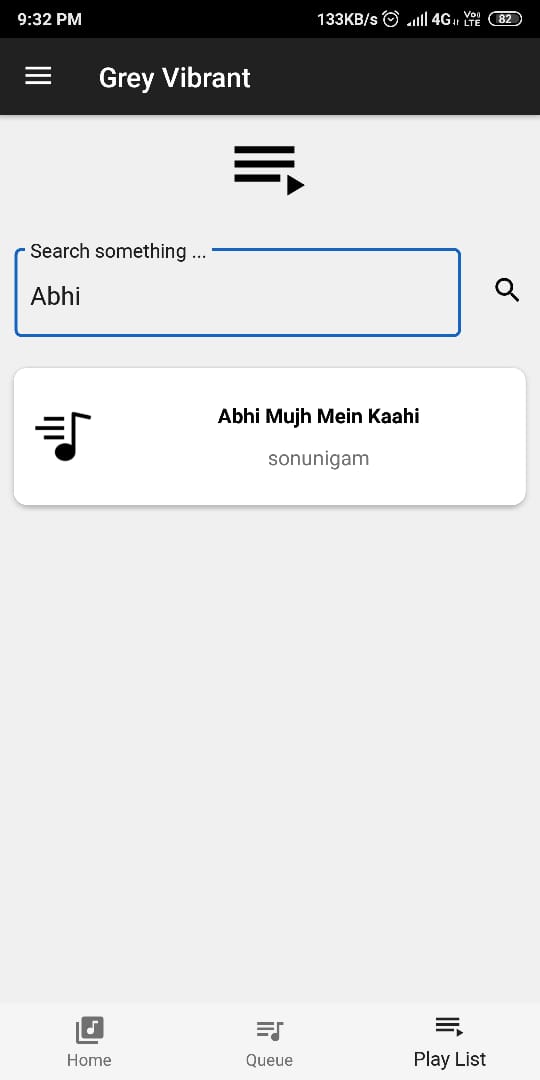
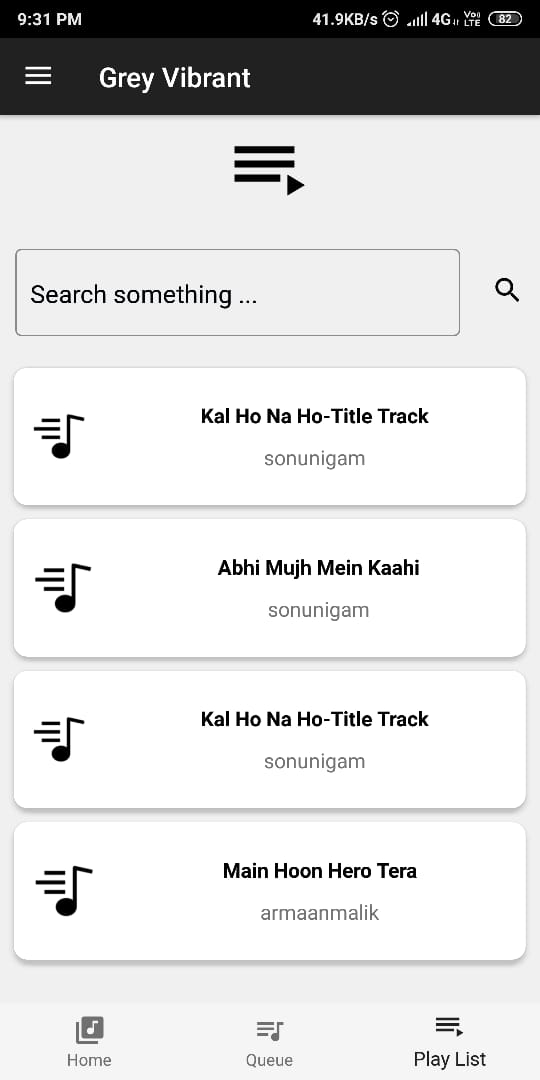
**3.a:** Followed and unfollowed artists **3.b:** More information about a song **3.c:** Hold song for long

4. Album: This module is provided to an artist to view all his uploads in his album. Search functionality is provided to filter the results as per artist’s choice. There is more information provided for each song and the song can be deleted from the album if the artist wants to.

(I): More info about song in album

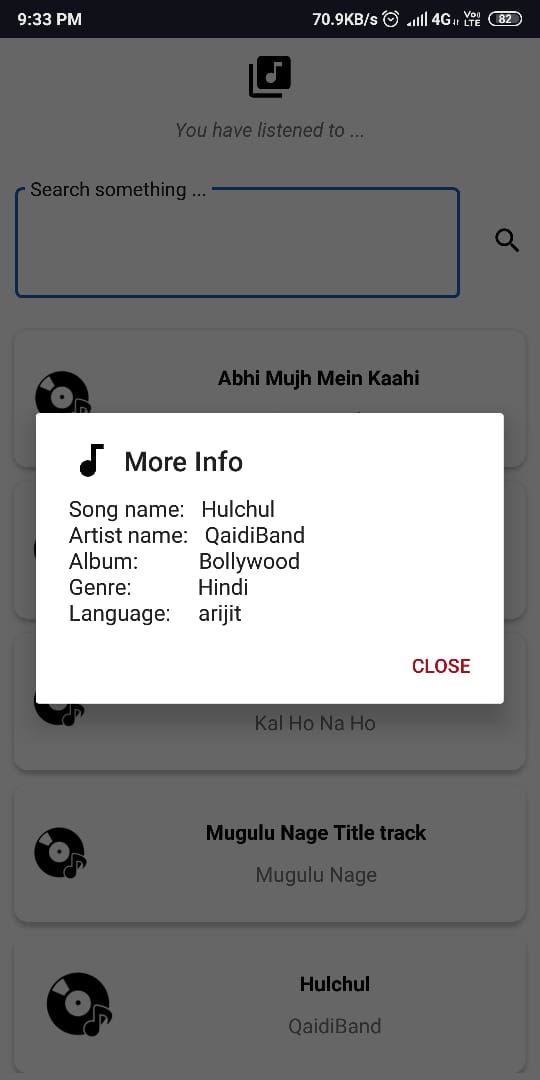
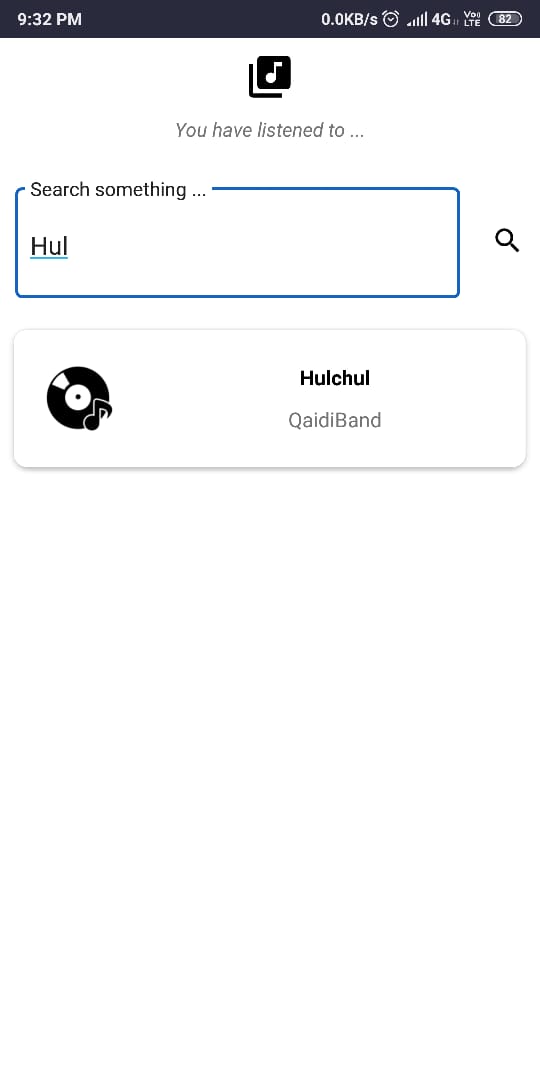
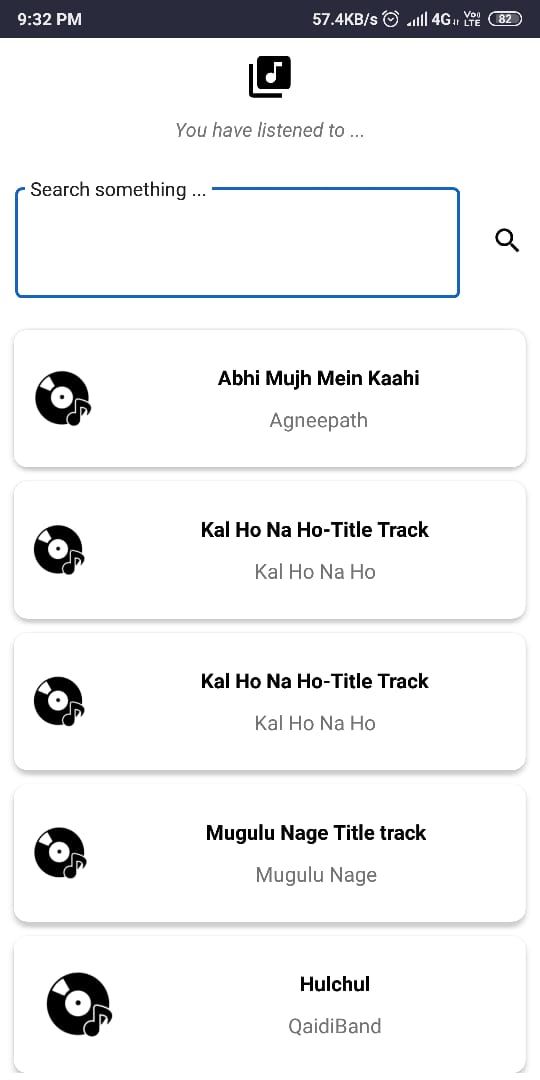
(ii): Album

5. Playlist: A playlist contains a list of songs that have been chosen by the user from a list of all songs available in the application. This playlist contains those songs which the user feels little special about. These songs can all fit in at one place called playlist. The functionalities provided in this module apart from displaying the playlist are: adding a song to the queue, deleting a song from the playlist, more information about the song and search functionality. The songs can be searched on the basis of song name, artist name, genre, and language.



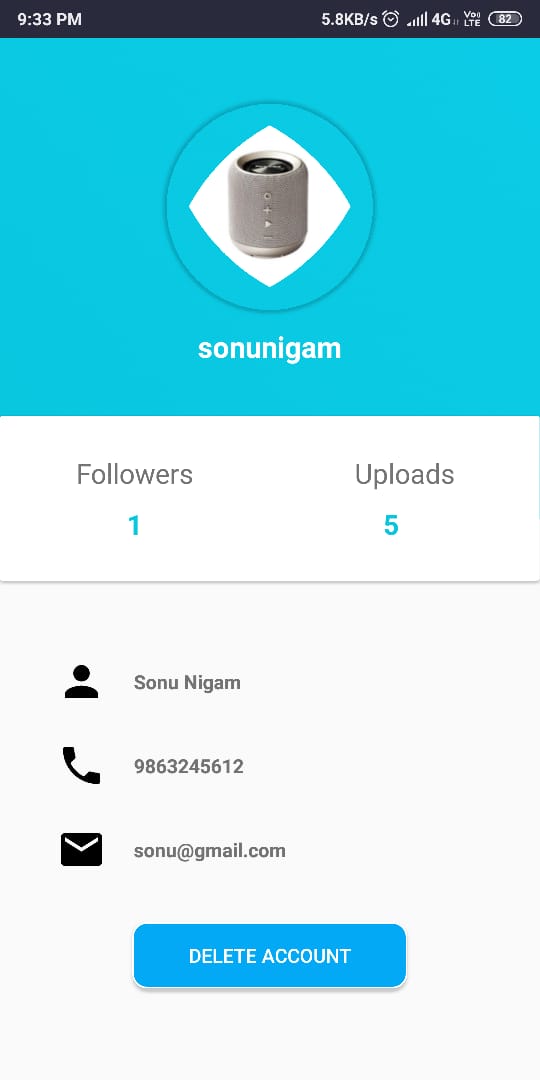
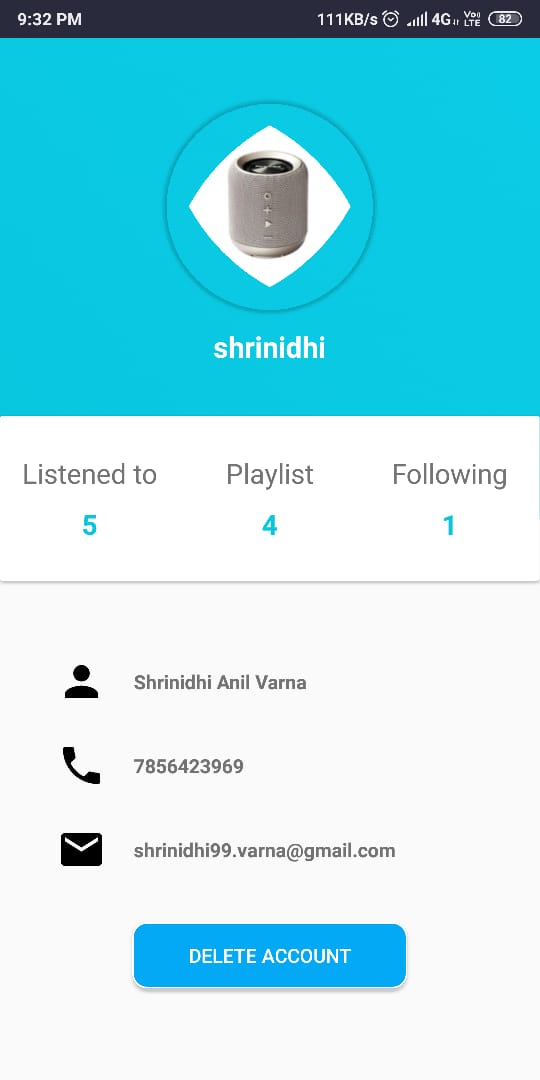
**5.a:** Playlist of a user **5.b:** Searching in a playlist

6. Song history: This module contains all the songs that have been listened by the user. The functionalities provided here are adding a song to the queue, getting more information about a song, deleting a song from the history and searching songs based on songname, artistname, genre or language.



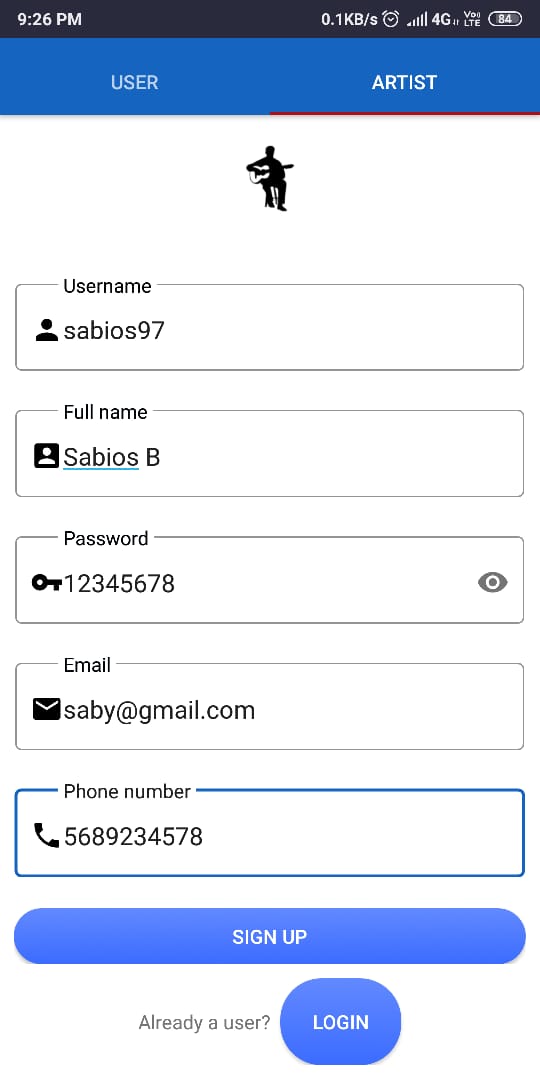
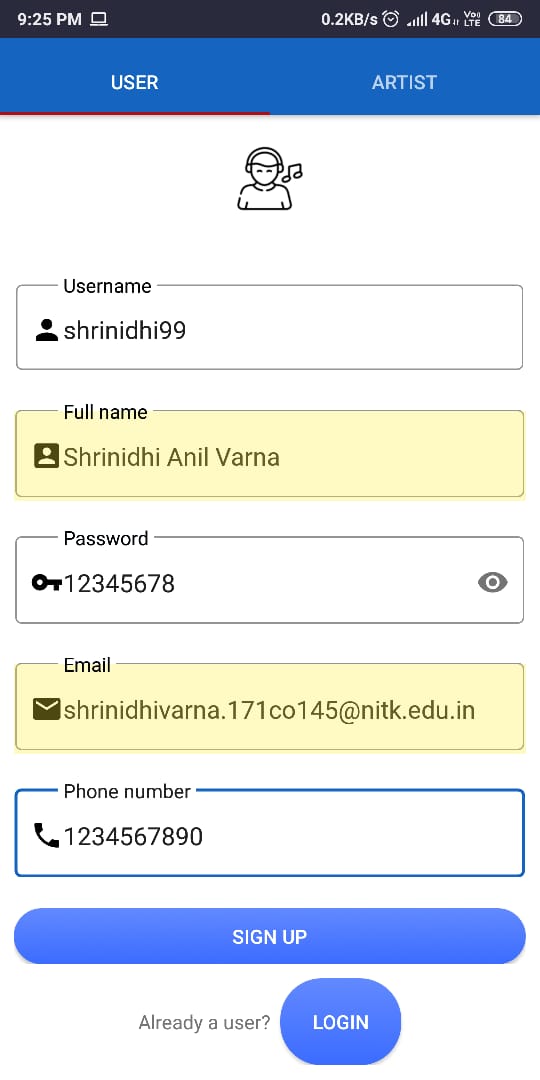
**6.a:** Song history **6.b:** Searching in song history **6.c:** Song information

7. Profile page: This module is provided for both user and artist. The user and artist can see some personal information provided to the application at the time of registration. The user can additionally see the number of songs present in his/her playlist, history and number of artists followed. The artist on the other hand can additionally see the number of users who follow him/her, and number of uploads in the album.



**7.a:** User profile **7.b:** Artist profile

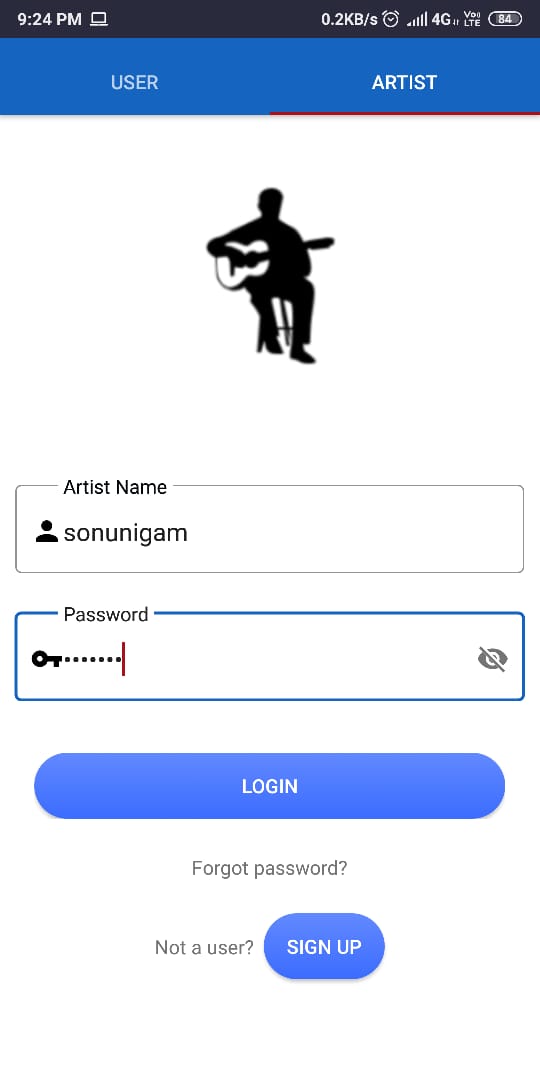
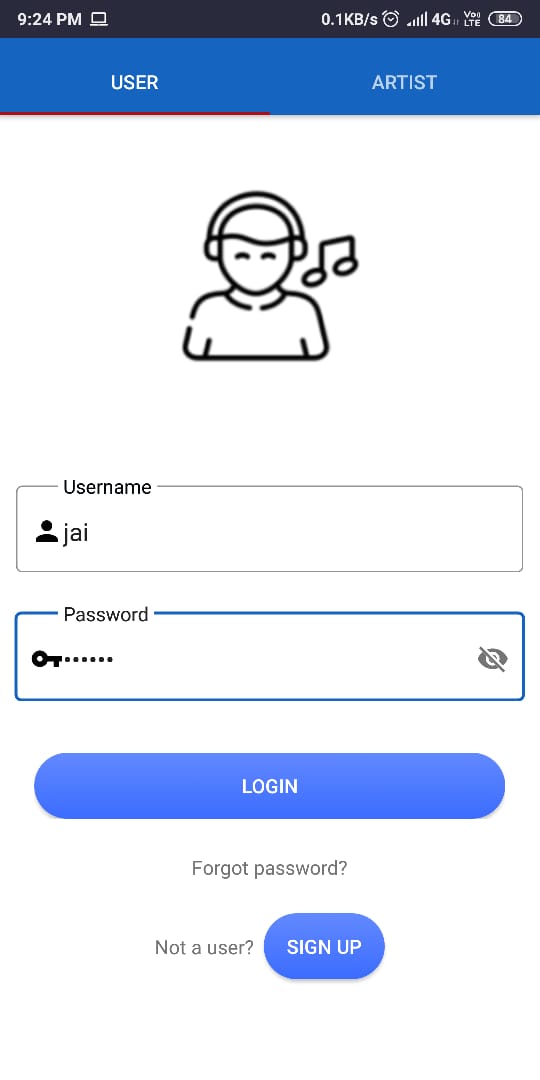
8. Registration page: There are two ways a user can register. One as an artist and another as a user for listening to songs. The registration happens in a similar manner taking the same set of details from both the user and the artist with some of the details taken keeping in mind about some factors. One such factor is to authenticate user who has forgotten the password.



(i) User registration

(ii) Artist registration

9. Login page: There are two ways a user can login based on what role he/she plays in the application. If the user is an artist then the artist login needs to be done otherwise the user login is the way to go ahead. The login is done by taking username and password where username is unique for every user in the application. Once the user/artist is authenticated, the various functionalities of the application can be made use of and there is a logout functionality that can let the user/artist log out of the application. In case the user/artist is currently logged in, then we have taken care of automatic log in once the user opens the application again after closing it.



**9.a:** User login **9.b:** Artist login

10. Shared preferences: This module is used to store user/artist credentials information in the memory till the user/artist session is on. The stored credentials authenticates the user/artist when he/she opens the app for the second time onward and bypasses the repetitive login authentication process. The stored data about the user/artist is also passed on as parameter to the PHP files for database operations. Once the user logs out the shared preference memory is cleared.

**5. TOOLS**

**FRONT END TOOLS**

Describe the front end tools and framework.

The front-end tool used for this project was Android and the framework used was android. The project was developed using Java language to make most part of the front-end and XML was used to design the user interface throughout the application. The XML is used for making the user-interface and in our application, all the activities and fragments that were made were all coded in XML. The user input, intents, displaying output in a proper manner, making requests to APIs were all done in Java. The button clicks, search query input and shifting from one page of the application to another was all part of java coding. The library used for connecting to MySQL server was Volley. Volley is a Java library used to make GET, PUT, POST, DELETE requests and in our case we used POST method of this library to make requests to the MySQL server. The android framework has ***app/src/*** folder where all the code can be found. The **build.gradle** file has all the dependencies used in the project. The ***app/src/main/res*** has all the XML files that were used for designing the user interface. The ***app/src/main/java*** has all the files in which java code was written.

The GitHub repository that contains the code for our project is [this](https://github.com/shrinidhi99/GreyVibrant.git).

**BACKEND TOOLS**

Describe about MySQL

The MySQL software delivers a very fast, multi threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. MySQL is a trademark of Oracle Corporation and/or its affiliates, and shall not be used by Customer without Oracle's express written authorization. Other names may be trademarks of their respective owners.

The database used for our project was MySQL. The queries were written in MySQL and the back-end server side scripting language is **PHP**.

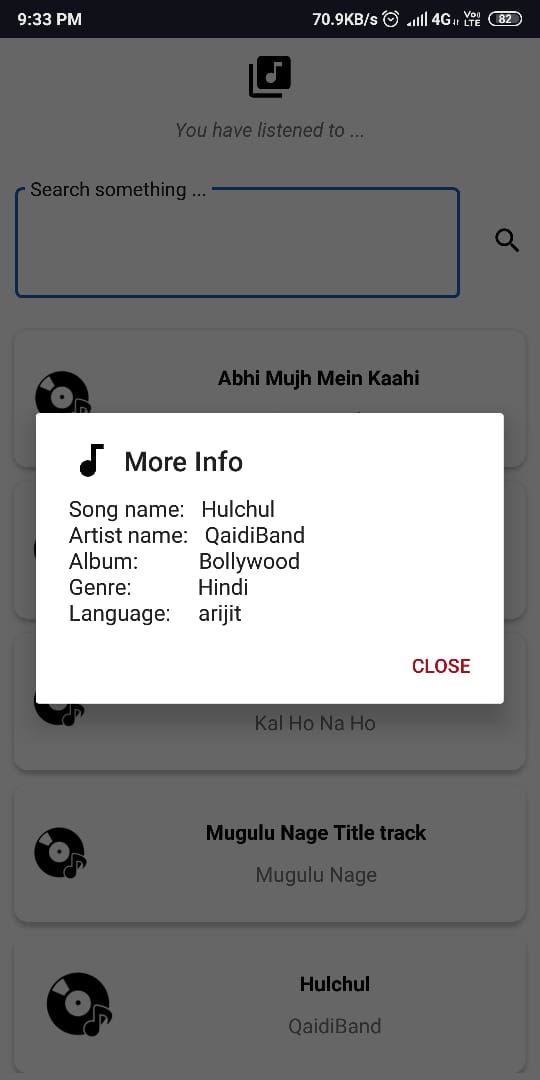
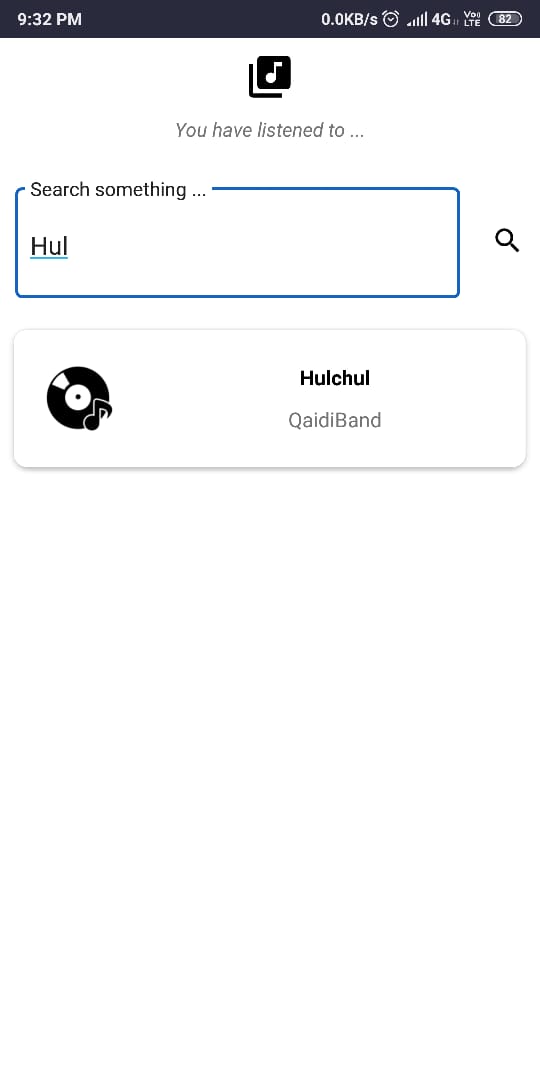
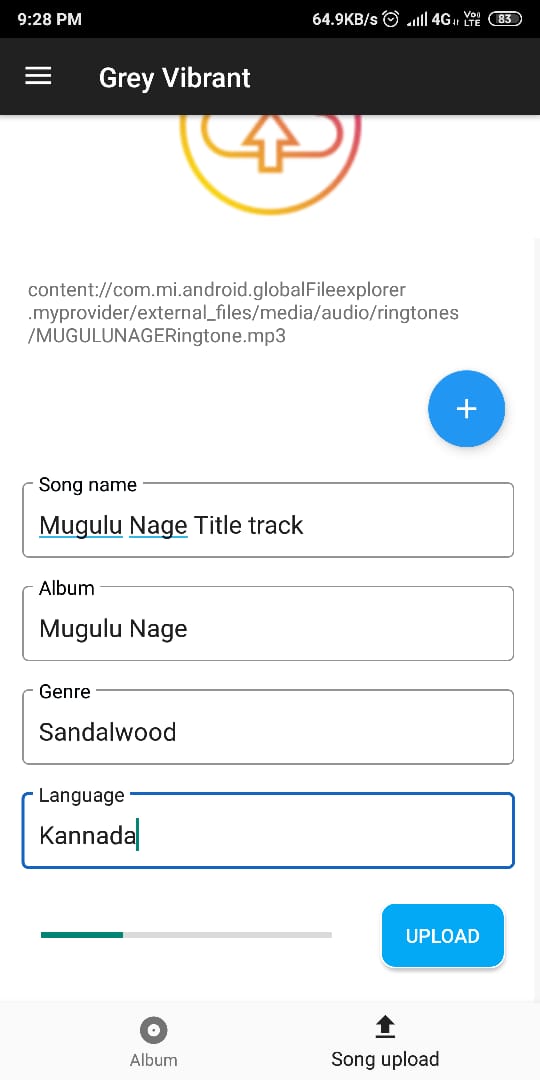
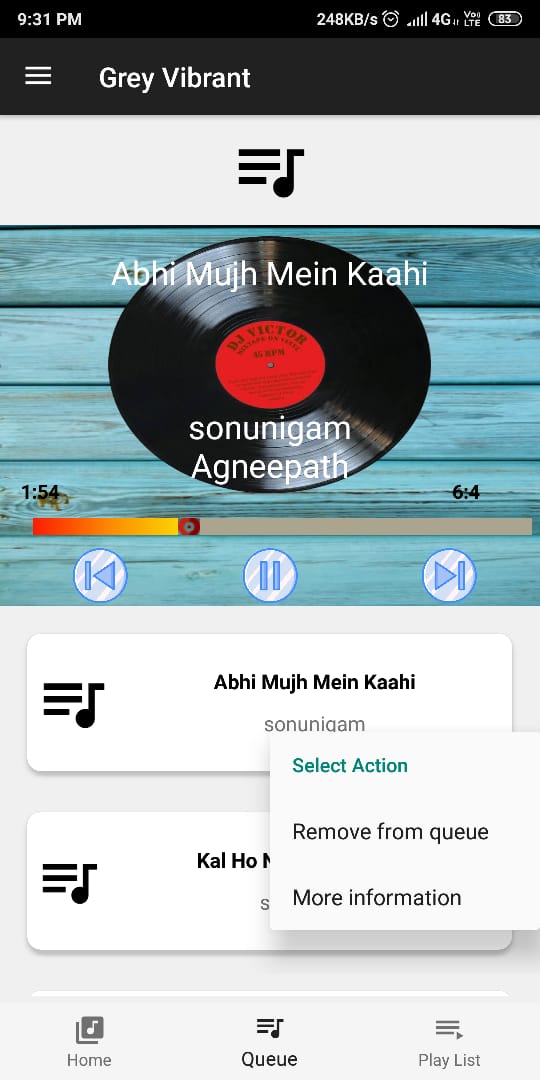
**PhpMyAdmin** is a software that hosts MySQL server. PHP file establishes a connection with the database and then performs queries that are needed. The queries are present in the PHP files which can be found in ***app/main/phpfiles*** folder of our GitHub repository. The Volley library makes app connection to one of these PHP files based on the query that needs to be performed. The PHP files upon getting POST request connects to the MySQL server running at PhpMyAdmin and sends the MySQL query as input. The output (tuples) received by the PHP file are sent back to the android application using the volley library. The tuples are encoded in JSON format before sending them to the android application for easier parsing in the application back-end. The back-end code can be partly found in the Java code which is purely for the android application. The PHP files can serve as a back-end to any application be it android, IOS or Web application.

**6. RESULTS**

Give a brief conclusion.

Paste screenshots of all the results you achieved.

The whole purpose of the application was achieved by the end of the development phase. The application can now upload songs, has two prototypes, can provide an option to create and modify a personal playlist, album and can play a queue of songs. The songs are streamed from the URL present in the database. The bugs that crept during the development phase were fixed after a thorough app manual testing phase. The music player comes with play, pause, play next, play previous functionalities. The additional information for any song can be fetched by the user or artist. The profile page shows some personal details which comes in as a non-functional requirement of the app.



(iii) Music upload (iv) Getting more information about a song

(i) Music streaming

(ii) Searching a song