D B M S Project

Online Music Store

Dupally Nikhil
S20170010045
nikhil.d17@iiits.in

Kiran Sankar E J
S20170010073
kiransankar.e17@iiits.in

Neelakanta Sriram
S20170010102
email address

Setty Venkat Vishwanth
S20170010143
venkatvishwanth.s17@iiits.in

Hemanth Vanam
S20170010172
hemanth.v17@iiits.in

Bhavani Vankdoth
S20140020046
bhavani.v14@iiits.in

***Abstract*—This document summarizes the application and usage of the online music app created for submission in the Database Management Systems course. The online music app, 'Blynk' is implemented using the flutter framework, sqlite and firebase database management system.**

I. Problem Statement

The problem statement is to create a functioning, easy-to-use, music application that can use database management systems to remember user preferences and cater to the needs of each individual user. The application needs to be sophisticated enough to handle a wide range of genres, artists, and songs but should be accessible by a wide range of users. The motivation behind this particular project comes from the unavailability of music apps without ads or in-app purchases to remove them. The challenges included the problem of storing and processing a huge number of songs and artists, implementing data scraping to automatically populate new songs from the internet instead of manually adding albums, songs, artists and such, and finding a data set which automatically updates itself on the aforementioned attributes in real-time on the internet.

II. Methodology

A. UML Diagram

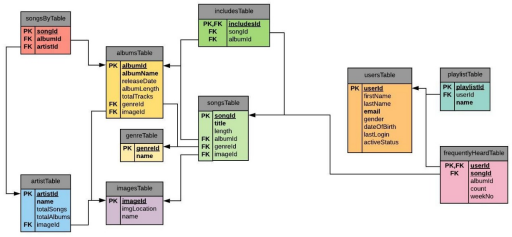


Fig. 1. UML Diagram

Fig. 1 is the UML Diagram. We have 10 tables with varying amounts of inter-connectivity. There is no main focus on one table because of the aforementioned connectivity. The firebase NoSQL is being used to store and retrieve most of the data from the internet. Mapping functions are being used to convert these objects into SQLite format for use in the android or iOS devices.

B. Use Case Diagram

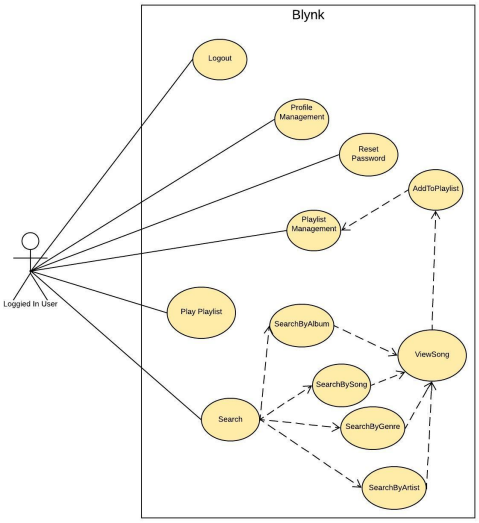


Fig. 2. Use Case Diagram

Fig. 2 is the Use Case Diagram.

C. System Design

From the given image (ref Fig.3), we can see that the user device fills its SQLite database from the data it acquires from firebase NoSQL database from the transactions that occur within the firebase and its NoSQL DBMS systems which are

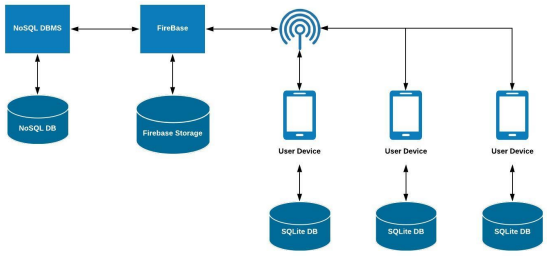
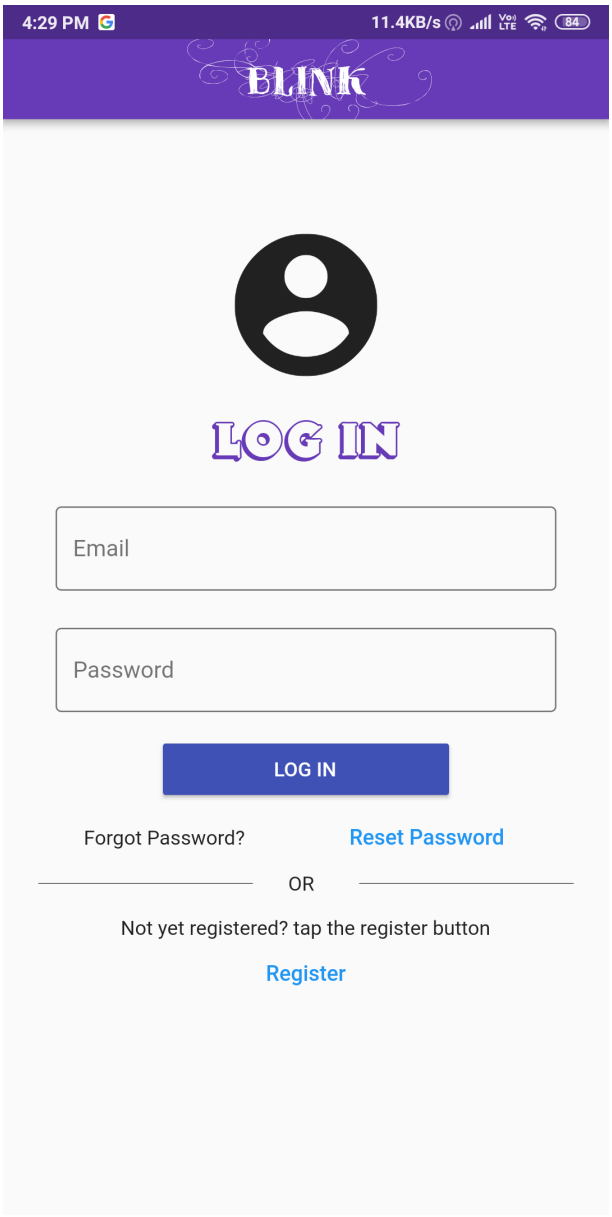


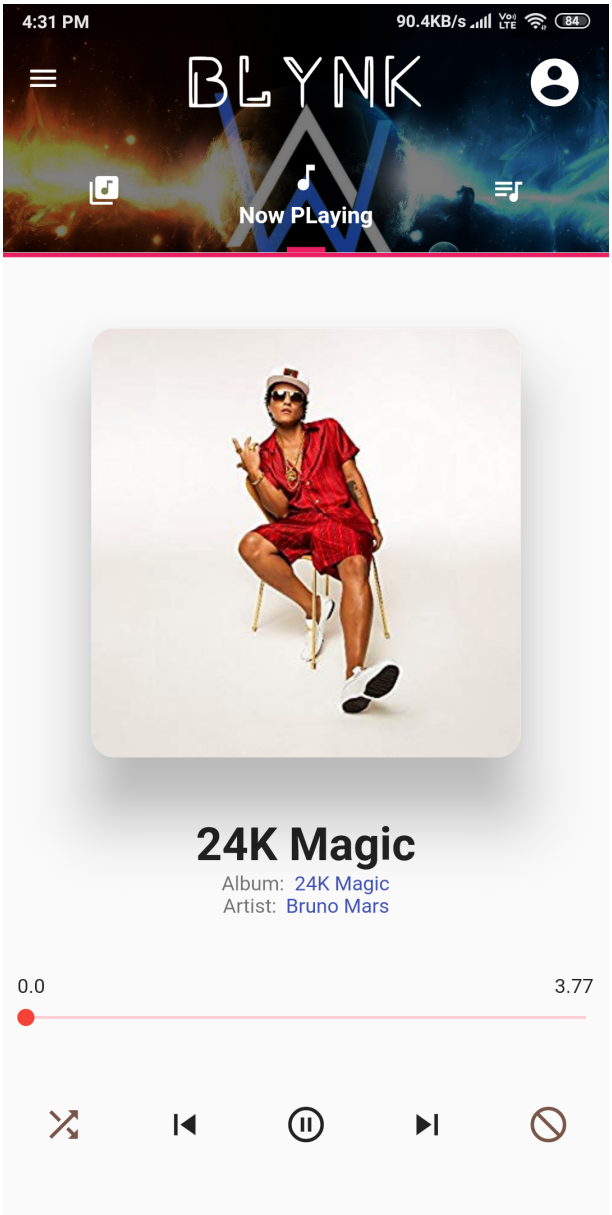
Fig. 3. System Design

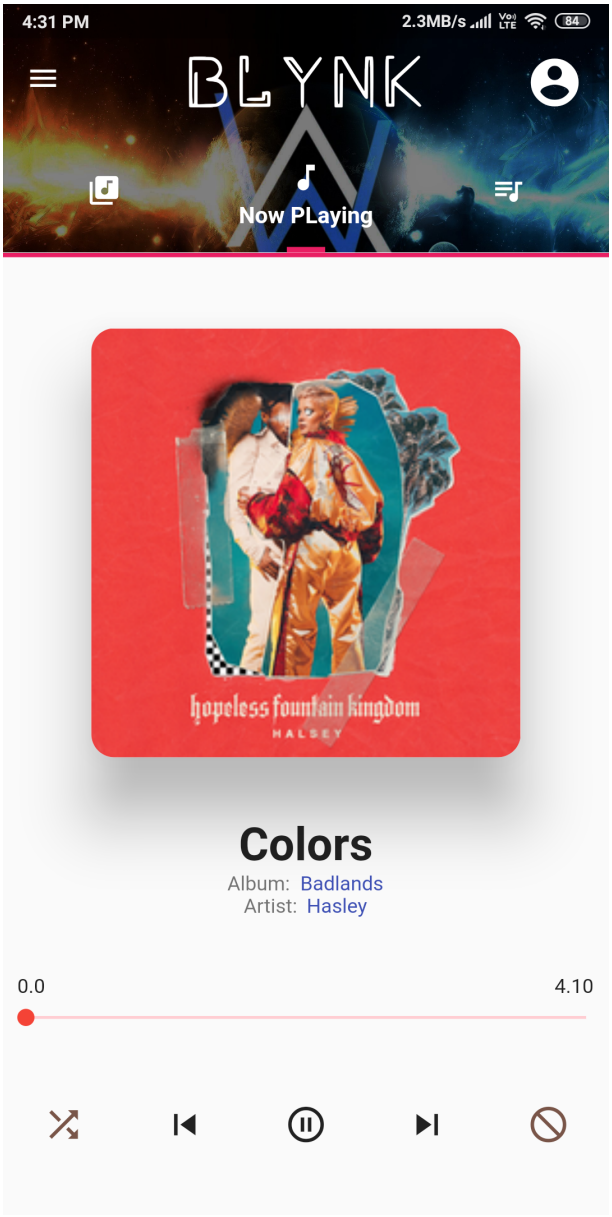
hidden from the user. So all of the relevant data corresponding to a given song or album is downloaded indirectly from the Firebase and NoSQL.

III. Results

A. UI Snapshots







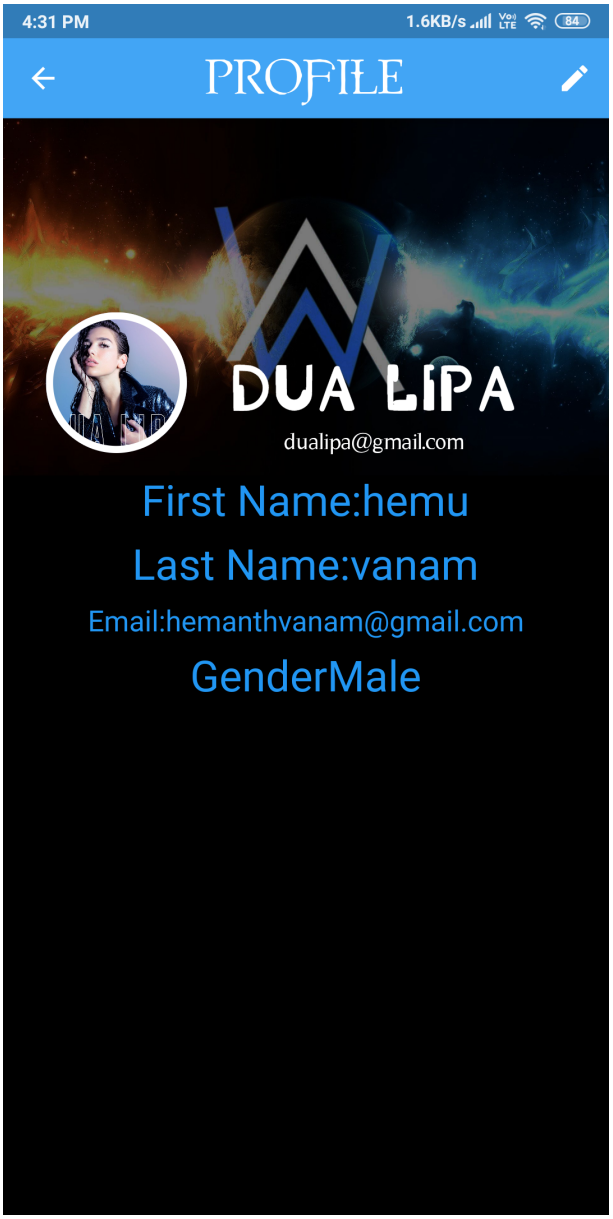
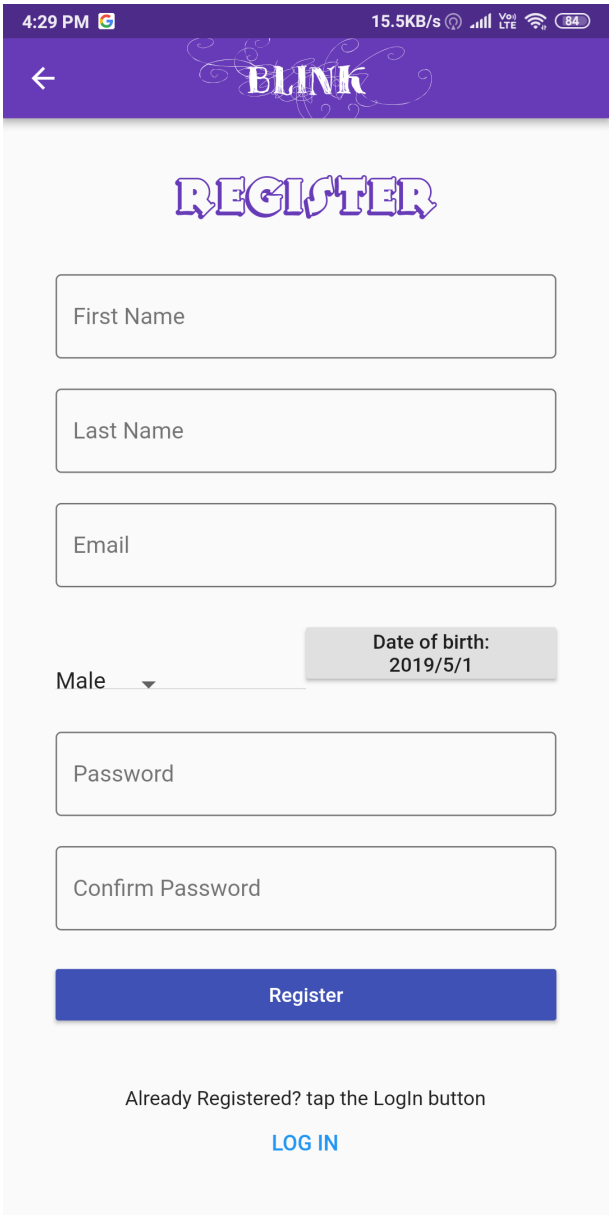
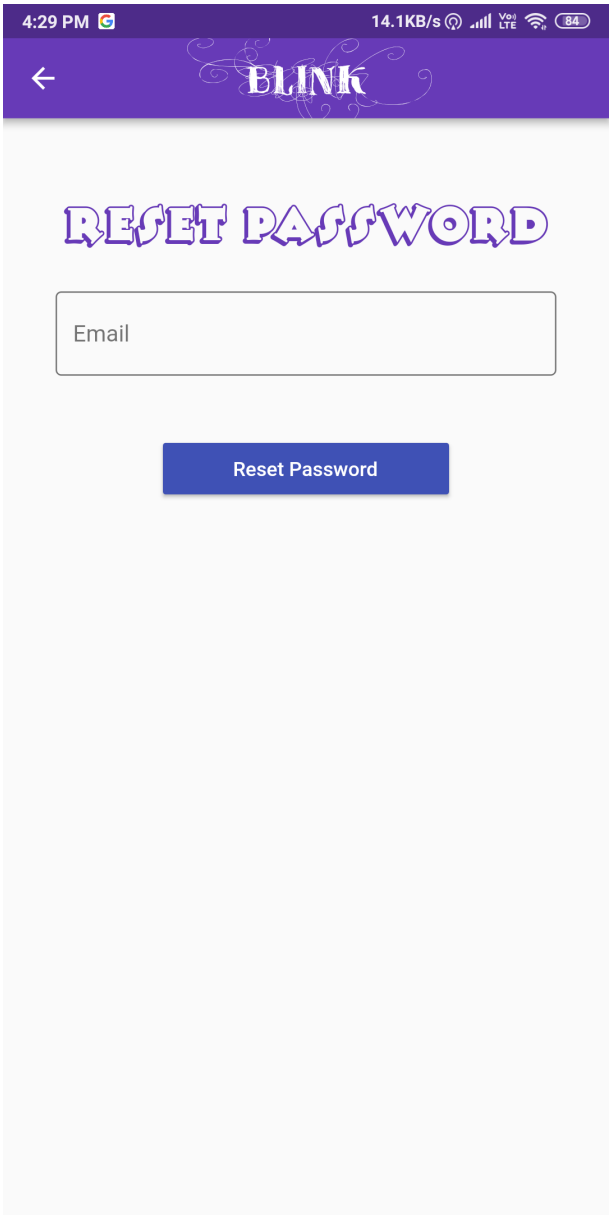
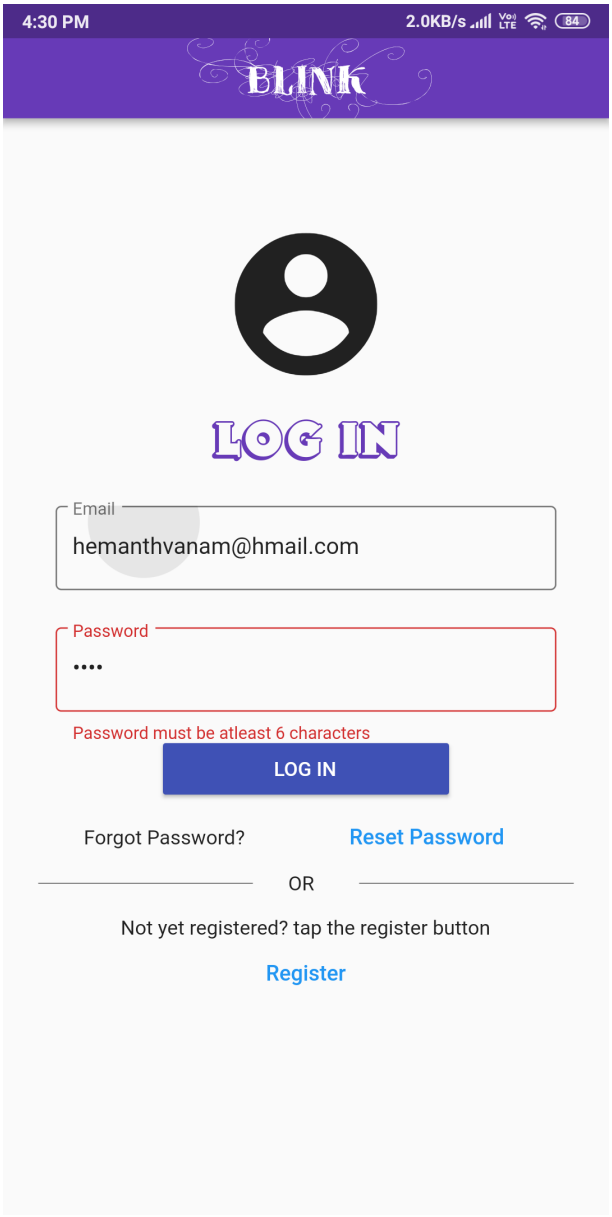
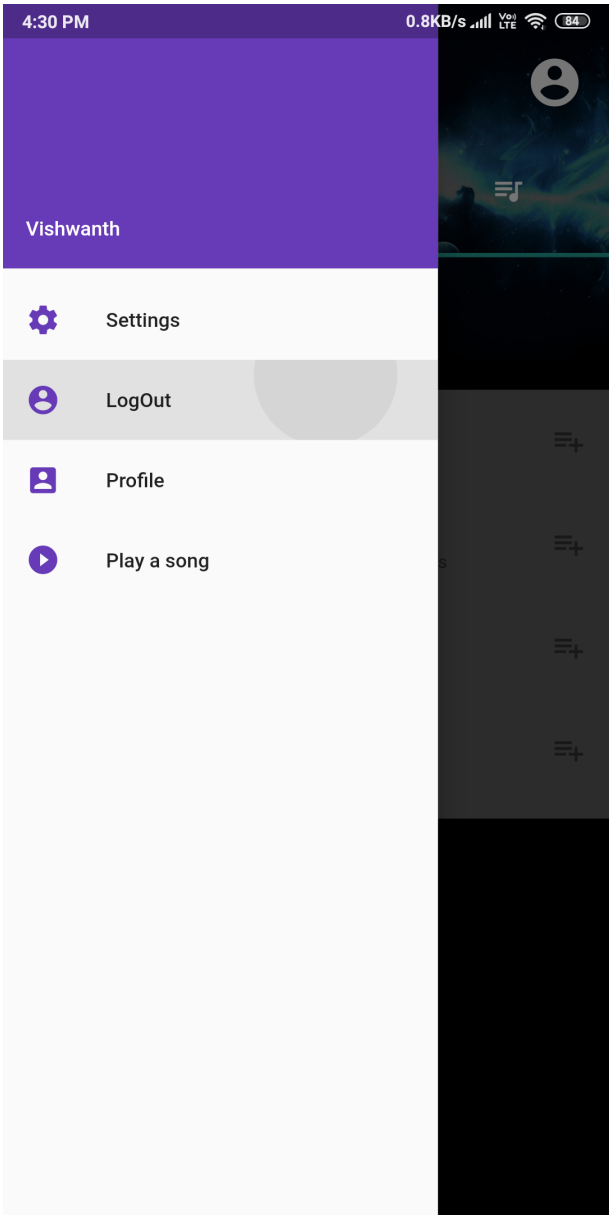


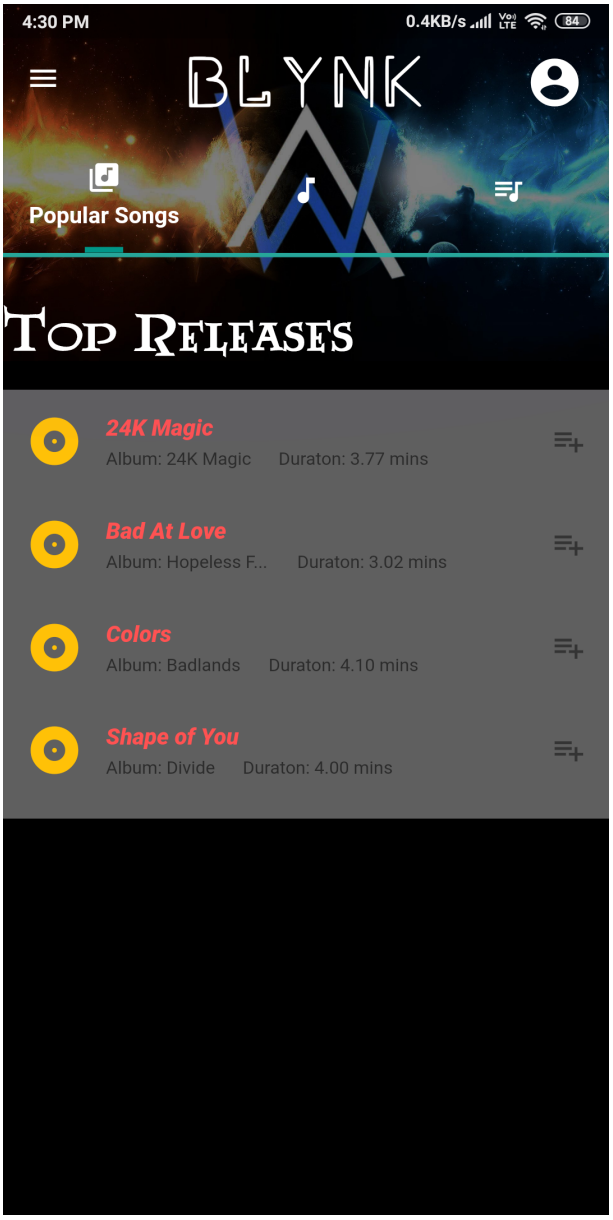
Fig. 4. System Design

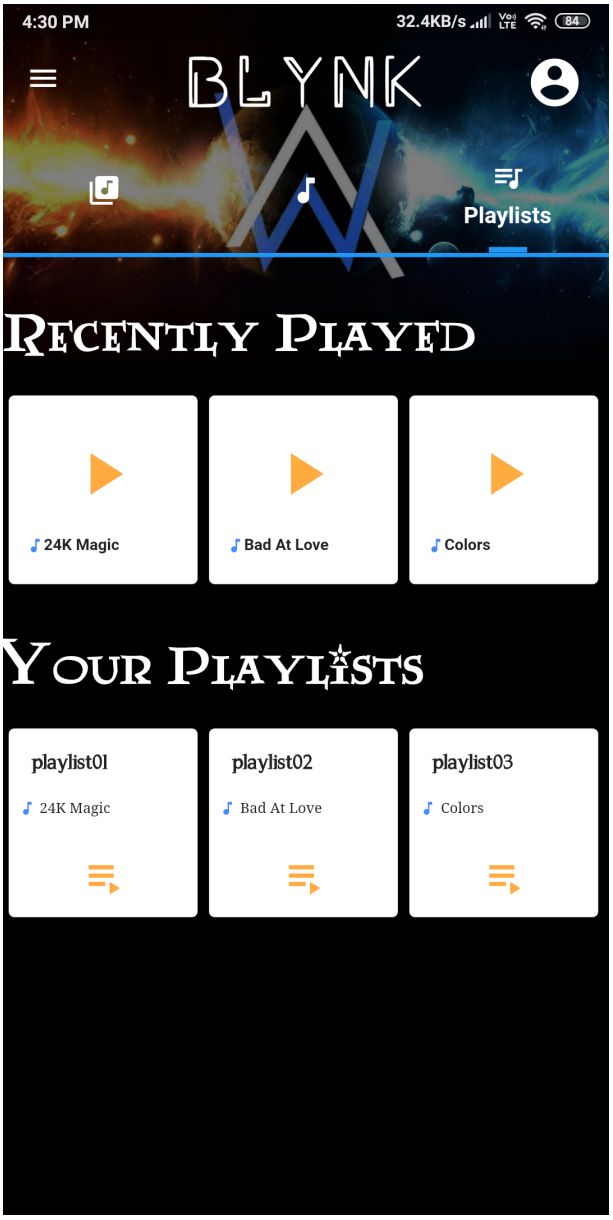

4:29 PM
15.5KB/s
BLINK
REGISTER
First Name
Last Name
Email
Male
Date of birth:
2019/5/1
Password
Confirm Password
Register
Already Registered? tap the LogIn button
LOG IN

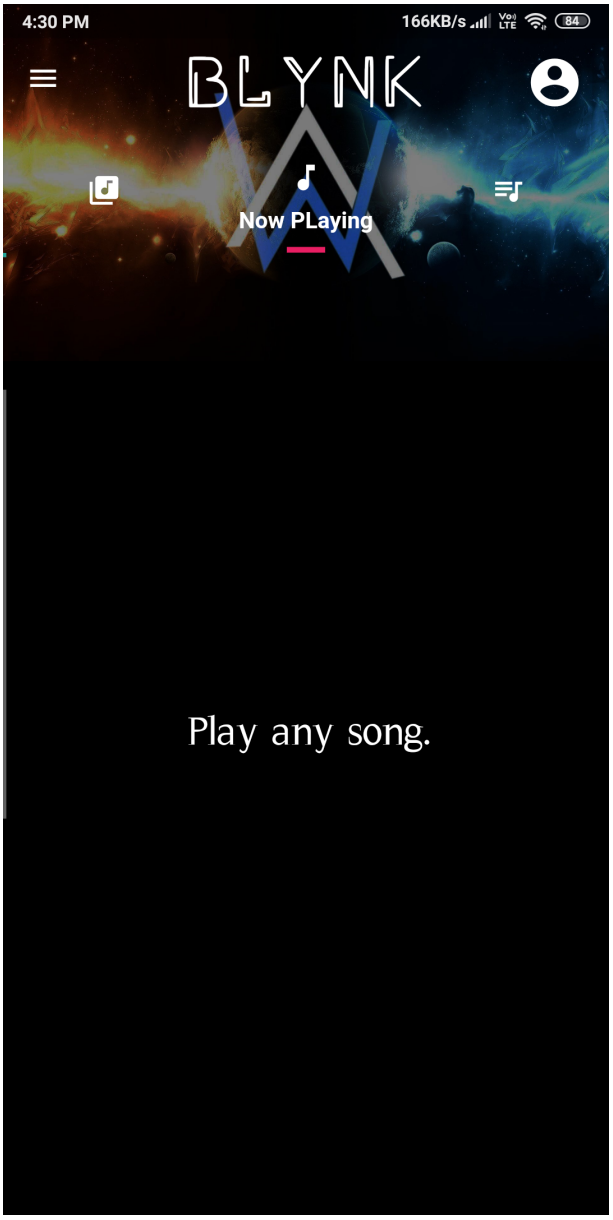

4:29 PM
14.1KB/s
BLINK
RESET PASSWORD
Email
Reset Password

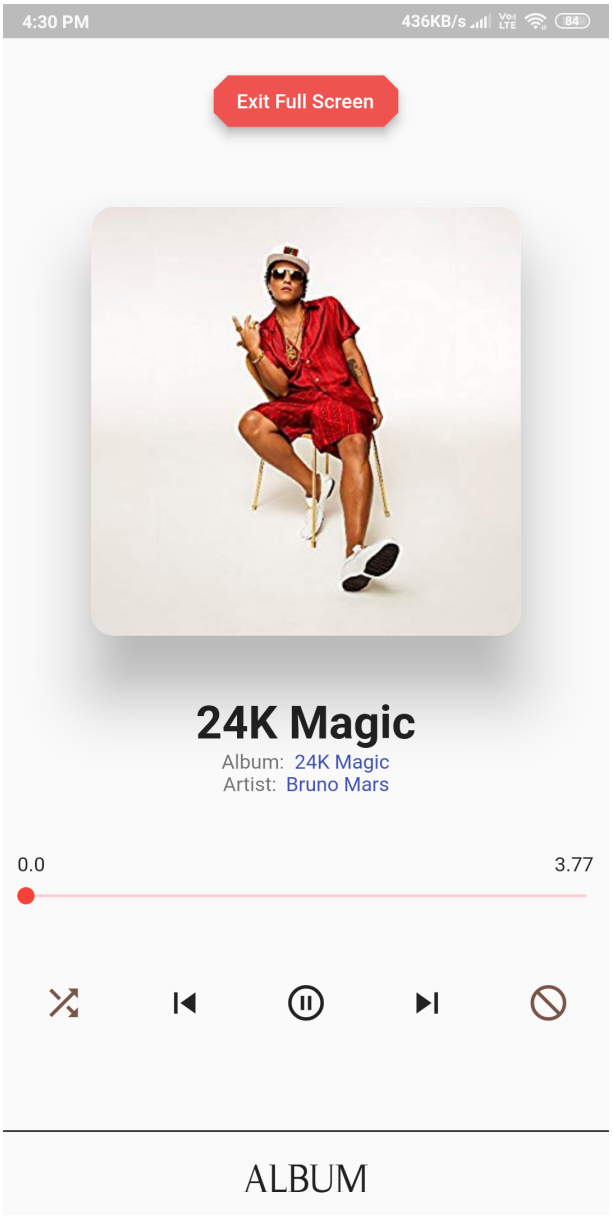

4:30 PM
2.0KB/s
BLINK
LOG IN
Email
hemanthvanam@hmail.com
Password
••••
Password must be atleast 6 characters
LOG IN
Forgot Password?
Reset Password
OR
Not yet registered? tap the register button
Register


4:30 PM
0.8KB/s
Vishwanth
Settings
LogOut
Profile
Play a song


4:30 PM
0.4KB/s
BLYNK
Popular Songs
Top Releases
24K Magic
Album: 24K Magic Duraton: 3.77 mins
Bad At Love
Album: Hopeless F... Duraton: 3.02 mins
Colors
Album: Badlands Duraton: 4.10 mins
Shape of You
Album: Divide Duraton: 4.00 mins


4:30 PM
32.4KB/s
BLYNK
Playlists
Recently Played
24K Magic
Bad At Love
Colors
Your Playlists
playlist01
24K Magic
playlist02
Bad At Love
playlist03
Colors





B. Results and Analysis

The Database section of the project has been more or less a success considering the unavailability of a practical real-time database for us to implement data scraping. The U.I part has been completed and we have succeeded in achieving almost all of the challenges in the problem statement excluding the aforementioned data scraping problem.

C. Robustness Testing

To assure that the application can work with any data, we have fed data to and from all the relevant databases and their systems and conducted debugs to ensure smooth working of the application in real-life conditions with respectable latency and access speeds.