**HARPER**

A PROJECT REPORT

BY

TEAM NO. 100

PRANJAL BHARDWAJ (E19CSE432)

ATINDRA SHEKHAR (E19CSE187)

SADHIL CHHABRA (E19CSE299)

A close up of a sign

Description automatically generated

SUBMITTED TO

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

BENNETT UNIVERSITY

GREATER NOIDA, 201310, UTTAR PRADESH, INDIA

NOVEMBER 2020

**DECLARATION**

We hereby declare that the work which is being presented in the report entitled “Harper”, is an authentic record of our own work carried out during the period from August, 2020 to November, 2020 at Department of Computer Science and Engineering, Bennett University Greater Noida.

The matters and the results presented in this report has not been submitted by us for the award of any other degree elsewhere.

Pranjal Bhardwaj

(E19CSE432)

Atindra Shekhar

(E19CSE187)

Sadhil Chhabra

(E19CSE299)

# ACKNOWLEDGEMENT

We would like to take this opportunity to express our deepest gratitude to our mentor, Dr. Manjit Kaur (Assistant Professor)for guiding, supporting, and helping us in every possible way. We were extremely fortunate to have her as our mentor as she provided insightful solutions to problems faced by us thus contributing immensely towards the completion of this capstone project. We would also like to express our deepest gratitude to Raghunath Shevgaonkar (VC), Dr. Debajyoti Mukhopadhyay (Dean), Dr. Deepak Garg (HoD), Dr. Tapas Badal (Assistant Professor), Dr. Gaurav Singal (Assistant Professor) and Dr. Vijaypal Singh Rathor (Assistant Professor),and friends who helped us in successful completion of this capstone project.

**No major bit of code has been taken from GitHub or any other internet sources. All of the codes have been self-created by putting all of our minds together into this project.**

Pranjal Bhardwaj

(E19CSE432)

Atindra Shekhar

(E19CSE187)

Sadhil Chhabra

(E19CSE299)

Contents

ABSTRACT 1

1. INTRODUCTION 1

1.1. Problem Statement 1

2. Background Research 2

2.1. Proposed System 2

2.2. Goals and Objectives 2

3. Project Planning 3

3.1. Project Lifecycle 3

3.2. Project Setup 3

[3.3. Stakeholders 4](#_Toc56211706)

3.4. Project Resources 4

3.5. Assumptions 5

4. Project Tracking 5

4.1. Tracking 5

4.2. Communication Plan 6

4.3. Deliverables 7

5. SYSTEM ANALYSIS AND DESIGN 8

5.1. Overall Description 8

5.2. Users and Roles 8

5.3. Design diagrams/ UML diagrams/ Flow Charts/ E-R diagrams 9

6. User Interface 14

6.1. UI Description 14

6.2. UI Mockup 14

7. Algorithms/Pseudo Code 15

8. Project Closure 16

8.1. Goals / Vision 16

8.2. Delivered Solution 16

8.3. Remaining Work 16

REFERENCES 17

**ABSTRACT**

This project is based on the problems that me and my team faced on a day to day basis in real life. We could have gone with various project ideas that were based on deep learning or IOT but we thought of finding a solution to the problem that not only me, but a community of millions of people were facing, because we thought that this project could solve a real issue.

We thought of a few different ways to fix this problem. And all of them would result in higher efficient use of the user’s resources.

We considered the daily experience of a user on day to day basis about how one user streams same data again and again and the previous data was discarded. We thought about identifying and storing the retrieved data, but that was way above our expertise level.

Therefore, we thought about a simpler and less efficient method which could improve the efficiency to a very high level overall. We thought about directly downloading songs according to the user’s choice and storing them to the persistent data storage of the user, so that there is no need of streaming same data again and again.

The project is divided into three separate modules. First is the user login and registration page. Another is the window of the music player. And the third module is where the download operation happens.

A major amount was taken in making the music player module, because it contains various hidden modules that needed to be debugged and improved to make the best user experience as better as we possibly could. Another module that took a lot of time was when we found out that the API that we were planning to use was no longer available as well as our app could not get certification. Therefore, we researched a lot but had to settle to a method where we could at least show how our application would work if completed.

We ended up retrieving the music file from a different API and storing it to the local storage of the user.

This way we could at least represent the way we expected the application to work. Although it was a fun project to work with. And the biggest thing it taught in the computer science field that the applications that might look simple, can be very complex. Me and my team enjoyed every bit of this project and hope you like it as well.

**ABOUT THE TEAM**

My team is a group of very ambitious students, who love coding. Each member of the team got exposure to coding in the first semester only. The team is of three members who have learnt about their interests about coding in the pandemic and a lot of knowledge gained in the pandemic was used to make this project.

**INTRODUCTION OF THE PROJECT**

The project is a Music Player that automatically as per the preference of the user would download the music in the local storage for anytime playing. The major API that would be used would be YouTube. The application would use major APIs – YouTube API and retrieved Machine Learning data. The application will be provided as a complete product, will be ready to use and will be able to run on almost all major mobile platforms and desktops. This would provide the user a variety of music available locally for any time playing without downloading each song individually.

The project is made using two major tools, first being the Xcode and the other being the Android Studio. The two languages used are Swift and Java. The project uses the Firebase Auth to authenticate the user. Only the authentication uses Firebase Cloud Database, and the rest of the project uses the local persistent database.

The application’s design was made using moqup.com. Application is divided into three main modules. First is the user login and registration module, the other is the music player, and the third is the list view which downloads the songs in the background. The download module was supposed to use data retrieved from the YouTube API but since the application could not get registered and verified by Google, we have added data to the code to show how we had expected the application to work. Then we used that data to use networking and download the music files to the persistent storage.

Once we are able to retrieve the YouTube’s user data, we could download the music files and store them in the persistent local storage. This project can be a solution to millions users. This would provide the user a variety of music available locally for any time playing without downloading each song individually. We tried to make the application available to multiple platforms. We had a lot of fun making this project and hope that you like it as well.

**PROBLEM STATEMENT**

There are a few music streaming services out there, but none of them provides the features that would. Firstly, YouTube is a great music streaming service but it streams videos along with songs and does not provide background playback.

For some basic functionalities companies like YouTube charge a pretty big amount. Also if one downloads video to offline mode, every week the downloads have to be refreshed and re-downloaded. This increases the internet data usage a lot and decreased the efficiency way more.

Secondly other streaming services like Spotify, JioSaavn, Gaana and Wynk, do not provide download feature and have a lot of advertisements. These are also streaming services, so one has to be online to listen to music.

Just like YouTube, these companies also charge its users a premium in order to obtain the profits. We found that these companies have a userbase of millions of people and a large population of them are frustrated by the increased amount of advertisements. Therefore we thought of making this application.

We believe that this application is a great solution for those who like to listen music frequently and spend a lot of data just to stream the songs. This project may not sound fancy but may have a real life application and can attract a very big user base.

**BACKGROUND RESEARCH**

There have been multiple events where people have shown their dislikes in the YouTube community and clearly told about their dislikes about the present application. A major problem nowadays is increased number of advertisements on music streaming websites especially YouTube and Spotify. There are some threads specially dedicated to increased ads. Youtube contains a variety of content and a large part of it is music. Therefore there is a large evidences to show that people do not like YouTube advertisements every other moment.

One such thread with around 200 replies on YouTube community is - [**https://support.google.com/youtube/thread/14482410?hl=en**](https://support.google.com/youtube/thread/14482410?hl=en) **.** I would like to share a few reply from this thread that clearly prove how annoying it is to have advertisements in between,

1. "**Am getting two, long ADS, for every song I listen to. YouTube is NO FUN anymore. What’s the point.",** by John Williams 6549.

2. "**Youtube ads are consuming my precious data.**", by User4052.. .

Also New York Times has posted an article regarding the increased advertisements and how the consumers are hating it, [**https://www.nytimes.com/2019/10/28/business/media/advertising-industry-research.html**](https://www.nytimes.com/2019/10/28/business/media/advertising-industry-research.html) **.**

Another thread that I would like to share is the one which shows that using offline mode on YouTube is nothing less than a headache, [**https://support.google.com/youtubemusic/thread/55192632?hl=en**](https://support.google.com/youtubemusic/thread/55192632?hl=en) ,

I would like to quote one reply from the above thread, "**Youtube music offline is terrible. Am I missing something**?" , by James Koehl. Also in Youtube, the videos have to be downloaded more than once.

One such reply in one of the thread is by Chelsey Van Overveld, "**I keep losing all my downloads and having to re download once have WiFi, I keep downloading music and then I goto work to listen to it and used all my data and I have to re down load it when I have WiFi! What’s the point of paying $12.99 if have to keep re downloading music and using my data!**".

Since our product is targeted towards such audience and we know that there are people that are looking for a product like what we are making, and we highly believe that this application would become successful.

**PROPOSED SYSTEM**

First let’s talk about the basic modules of the application that are the login and registration module and the second being the music player.

In order to authenticate the user, we have used Firebase Authentication. This would allow us to store the user’s login data securely and retrieve the authentication result securely.

For the second module that is the music player, a framework has been used that is the Audio Video Foundation, which provides the basic functionality of loading data and playing the audio or video files. Various other complex functions in the music player module have been coded in order to improve user experience.

For the last module of the application we had to retrieve music file and store it to the persistent data storage. In order to obtain the results, we researched and found that there were various methods to reach the solution, but some of them were way above our technical knowledge, like the part where we could not figure out how to store data from cache. Therefore we thought about a different method that could provide us the expected result along with high efficiency and great user experience. We came to the method where we would generate a link and using basic networking, retrieve the data and store it to the persistent local storage.

We thought about using YouTube API and were about to incorporate the API in the project, but we found that the API is not updated to the current requirements as well as in order to obtain the user’s data we had to get the application certified by Google, which we were unable to do. Therefore we added the converted link directly to the project in order to depict how the project would look, if we had the complete resources. Then we used basic networking to retrieve the music files and stored them to the local storage of the device.

**GOALS AND OBJECTIVES**

* The user should be able to login into the application safely.
* If the user is not registered, the user should be able to register new users to the database.
* The user should be able to play music file in the application.
* The music must continue to play in the background.
* Music file must automatically download and store itself to the persistent storage.
* The user must be able to see the updated list after the song downloads.

**PROJECT PLANNING**

**Project Lifecycle :-**

The team will focus first on completing the complex as well as basic operations of the application. The main goal of the team is to make a simple but attractive application that does its job with efficiency. Since there are not much members in the team, the team focuses on the fact that one has to complete his responsibility and contribute to the project. The approach of the team is to get the work done at its best.

**Project Setup :-**

Various decisions were made during making of the project and some of the major decisions are stated below:

* + - * Firebase Authentication VS Local Authentication
      * Best interface design for the best UI/UX experience
      * YouTube API vs Spotify API
      * Database choice Realm VS MySQL
      * Most efficient manner to download data

**Stakeholders :-**

Numerous stakeholders can be identified with this project and are listed below:

* Dr Vijaypal Singh Rathore – Guide
* Pranjal Bhardwaj – Team Member
* Atindra Shekhar – Team Member
* Sadhil Chhabra – Team Member
* YouTube Community – Expected End Users

**Project Resources :-**

There are not many resources that are required for this project after the project is completed, since it is mostly an offline application. But the important resources have been listed below:

* User Authentication Server – 1
* Number of Developers – 4
* Test Users to test the end application – 10
* Mac Workstations – 2
* Smartphones to test the application – 2

**Assumptions :-**

The assumptions upon which the project is based are listed down:

* The developer should be able to retrieve APIs the data to the application.
* Team should have completed first two modules of the project in 2 months.
* The test data would be enough to show how the complete application was expected to work.

**PROJECT TRACKING**

**Tracking :-**

The version control system that is used for this project is GitHub. All the code files as well as the text files that contain the weekly status diary and other information has been pushed to the GitHub.

To go to the repository please go to this link –> <https://github.com/DecimatorMind/Harper>

The various commits of the project when a specific module was completed or modified were pushed to GitHub and can be found in the project commits.

To go to the weekly status diary go to –> <https://github.com/DecimatorMind/Harper/blob/master/WeeklyStatusDiary.md>

Each time any code was referenced from any website or video, the website link was added to the project References.md file.

To go the references file go to –> <https://github.com/DecimatorMind/Harper/blob/master/References.md>

At the beginning of the project, the UI mockups were captured and stored to the repository.

To access all the designs of the mockups go to –> <https://github.com/DecimatorMind/Harper/tree/master/UI-UX%20Mockup>

All the project is available on GitHub as a public repository, that anyone can fork and contribute to.

**Communication Plans :-**

The team members were in regular contact with each other. The meeting regarding the agendas of the project was done twice every week.

On Monday and Wednesday, we would create a personal meeting on Google Meet, and join and share the progress of the application. We would help each other if anyone is facing any problem. We had a separate group on WhatsApp in order to regularly get updated and share any ideas.

Twice in a week, we had a meeting with our mentor, where she would suggest the change in the existing project, if any.

**Deliverables :-**

* Code
* Final Report
* GitHub Repository
* Weekly Report
* Test and results

**SYSTEM ANALYSIS AND DESIGN**

**Overall Description :-**

The project is divided into three modules. First is the user login and registration page. We have used Firebase Authentication in order to store the user login credentials safely on the cloud database.

The first module segues to the second module that is the music player. The music player uses the AV Foundation to play the music files and load them into the application. AV Foundation is an inbuilt framework in swift that lets the application to play any audio or video file. The music player consists of various complex functions. One of them is the seek bar. It took a lot longer to program the function that controls the seek bar, because it had to be updates every passing second, therefore we set a timer that would call an increment function every second. The AV Foundation framework allows the program to play the file by locating it and creating a URL of the file. That URL is used as a function argument that loads the music or video file. And when the play function is called the music file starts playing. But playing the file is not the only task. There are numerous buttons and functions that were made in

order to receive basic functionalities. This module segues to the last module of the application.

This song list view module allows the user to see the info about the songs that are locally stored in the user’s device. And the user can see an updated list when the song finishes downloading the song. The last module is a list view with downloading code. The song list is updated every time a new download happens. The module uses a basic networking code, that retrieves the data from the URL and stores it to the local storage. The code uses data from the Realm database that contains the data of the song info and some test data. This module is a dynamic module that appears on the top of the previous module and can be dismissed easily.

**Users and Roles :-**

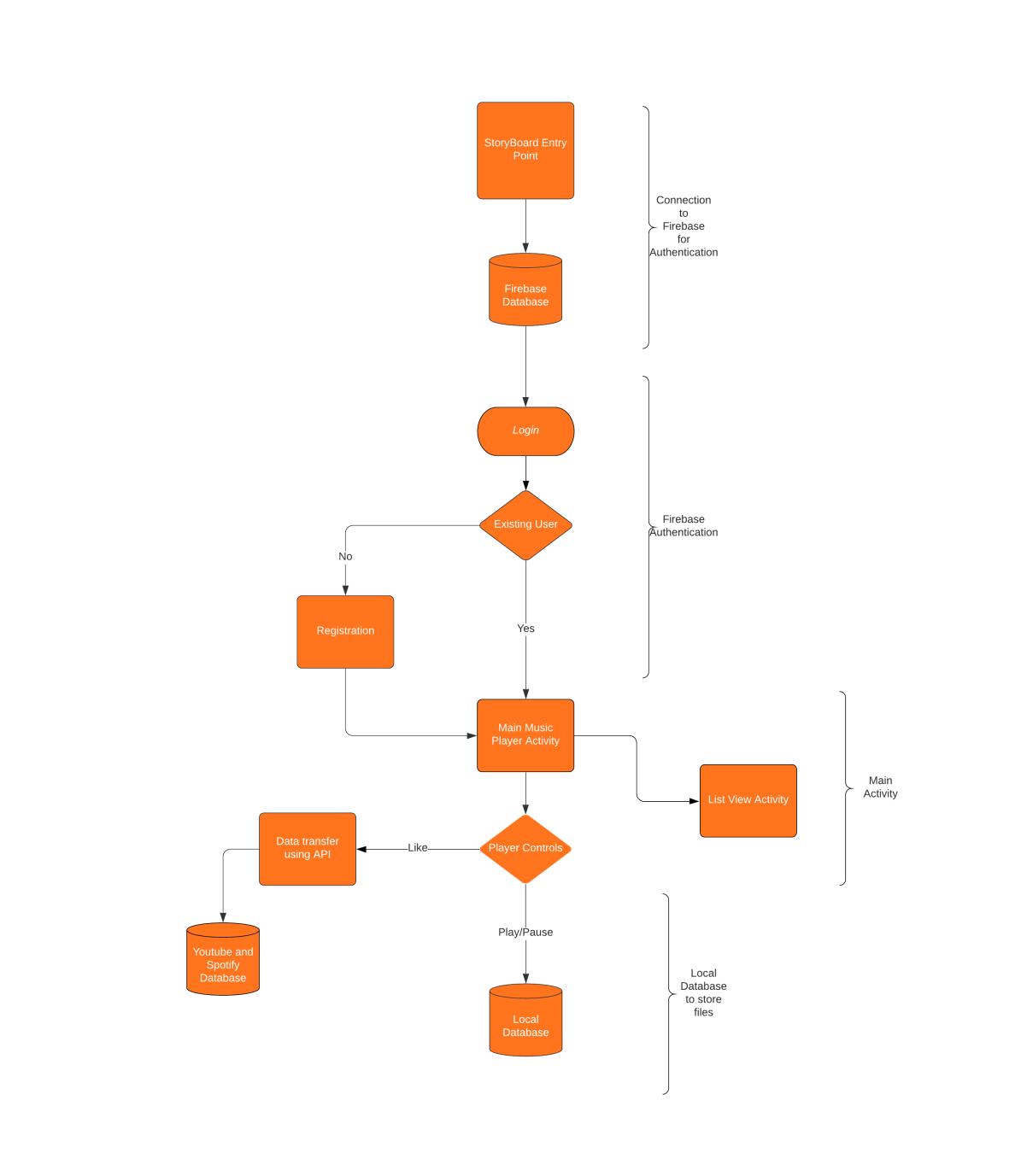
This project requires only developers to build the application, because this application works majorly in the offline mode. The developers need to make a multiplatform application that does the expected task.

**Design Diagrams/ UML Diagrams/ Flow Chart/ ER Diagrams :-**

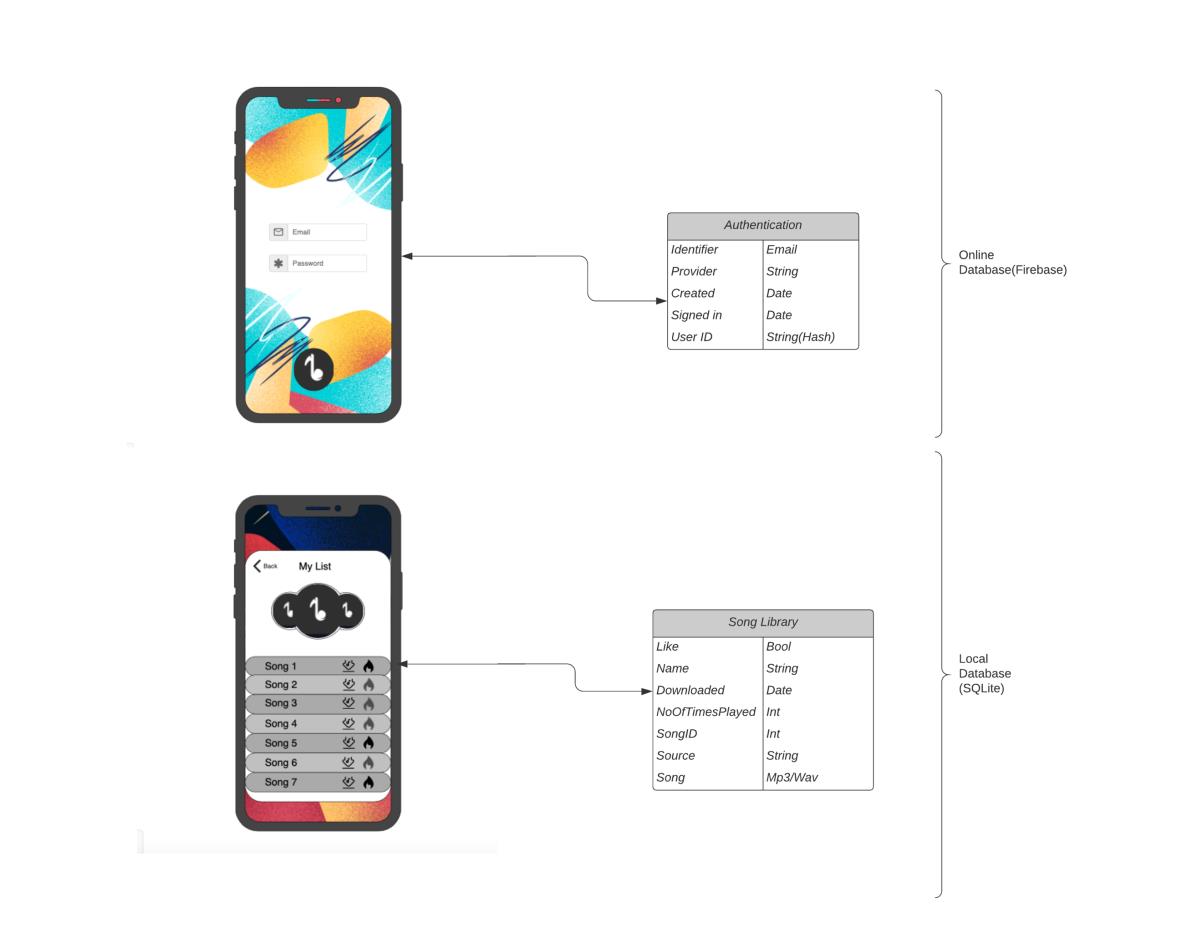
For the Design Diagrams, following are the mockups that were made during the designing process of the application. A lot of time was spent on the application design as it is the foundation of any application.

These were the various snapshots of the designs that were made during the application interface designing process.

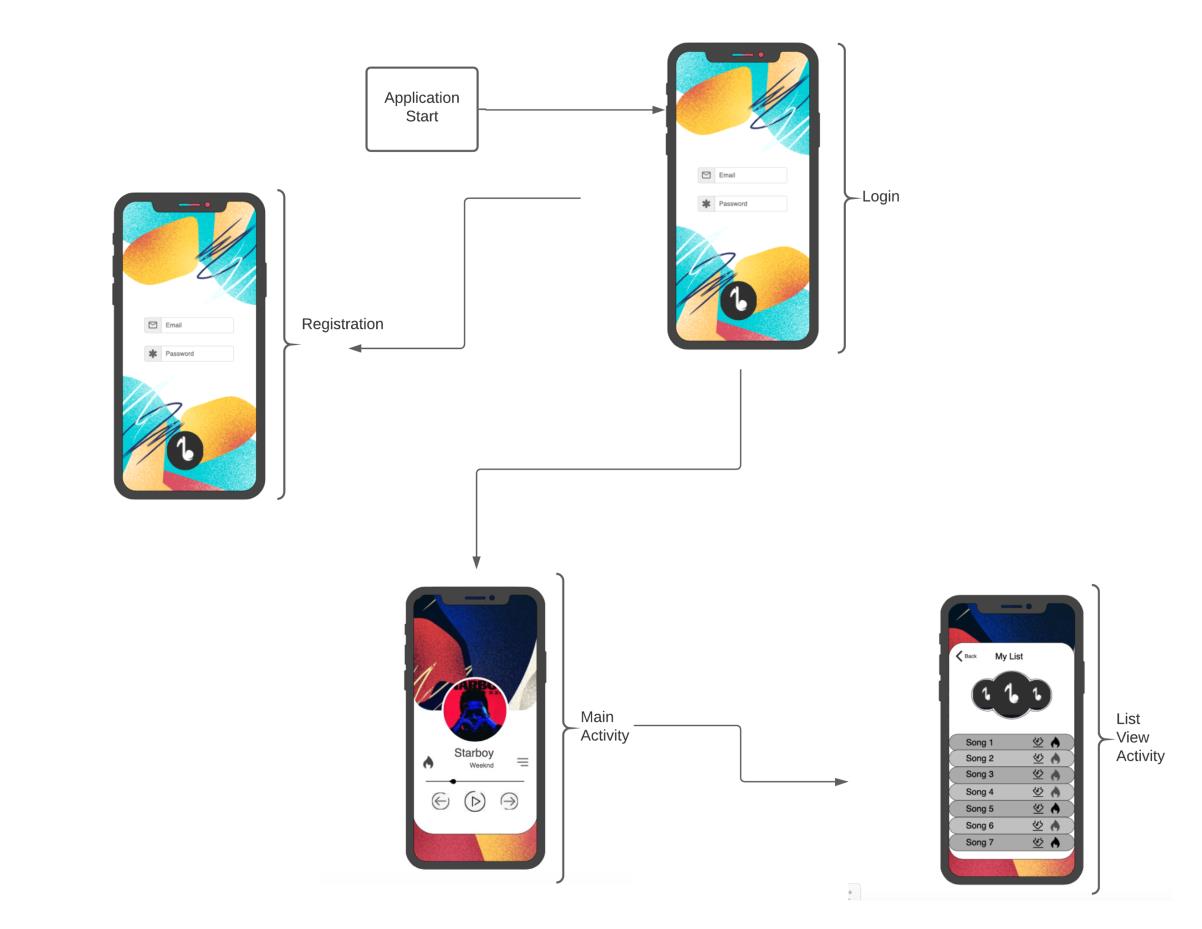
Following is the diagram representing the flow of the application.



Following is the design database architecture of the different modules.



Following is the user experience flow chart that depicts how the application is going to work at the end.



**USER INTERFACE**

**UI Description :-**

The program consists of three major UI modules, all of the three have been very well explained above along with their pictures. The project uses the Swift Storyboard to design the XML files of the application, although the XML design files are not directly accessible by the developer.

**UI Mockups :-**

1. Login and Registration Page

Graphical user interface, application

Description automatically generated

1. Music Player Module
   1. First Finalized Mockup

Graphical user interface, application

Description automatically generated

* 1. Second Finalized Mockup

Graphical user interface, application

Description automatically generated

* 1. Third Finalized Mockup (Final Design)

Graphical user interface, application

Description automatically generated

1. List View Module

Graphical user interface, application

Description automatically generated

**PSUEDO CODE**

import UIKit

import Firebase

import AVFoundation

import Realm

import RealmSwift

class MainViewController: UIViewController{

var SongPlayer = AVAudioPlayer()

var timer = Timer()

var lastPlayedIndex = 0

override func viewDidLoad() {

super.viewDidLoad()

ReadyPlayer(temp: SongName[lastPlayedIndex])

UpdateData()

Slider.maximumValue = Float(TimeInterval(SongPlayer.duration))

}

@IBAction func Play(\_ sender: UIButton) {

if(Song is playing){

Pause the song

Update the icon image

Pause the timer

} else {

Play the song

Update the icon image

Start the timer

}

}

func ReadyPlayer(temp: String){

let path = Load the path of the music file

let url = url generated by the conversion of the url file

do{

SongPlayer = try contents of the loaded music file but not play yet

} catch{

Print if any error occurs

}

Prepare the player to play.

let AudioSession = Create a new audio session that allows for background

playback.

do{

Allow background playback

} catch {

Print error if any

}

Slider.maximumValue = Maximum value of the song length

}

@IBAction func Next(\_ sender: UIButton? = nil) {

Ready the player with the next song.

if(check if it is not the last song){

Increment the last played index

Update the UI data

Play the song

Update the button icon

Start the timer.

}

}

@IBAction func Previous(\_ sender: UIButton) {

Ready the player with the previous song.

if(check if it is not the first song){

Decrement the song index

Update the data function is called

Play the loaded file

Update the icons

Start the timer.

}

}

@IBAction func TimeSlider(\_ sender: UISlider){

Set the current time of the song to the value of the slider.

Play the song.

}

@IBAction func Like(\_ sender: UIButton? = nil) {

if(If the song is not liked){

Set the status of the current song to 1

Update the data and the icon

} else {

Set the status of the current song to 0

Update the data and the icon

}

}

@IBAction func ToMenu(\_ sender: UIButton) {

Open the menu window

}

func UpdateData(){

if(song is not liked){

Set icon image to not liked

} else {

Set icon image to liked

}

Set label text to current song

Set artist text to current song

Reset the slider value = 0

Calculate the duration of the song

Set the text of the end label

}

func StartTimer(){

Call the function update timer after every second.

Start the timer

}

func PauseTimer(){

Pause the timer

}

@objc func update(\_ timer: Timer){

Set the slider value to current time

Calculate the time from the current time

Change the label text

if (If song is ending){

Play the next song

}

}

func calculateTimeFromNSTimeInterval(\_ duration:TimeInterval) ->(minute:String, second:String){

let minute = calculate minute from the current time

let second = calculate second from the current time

return (String(minute),second)

}

func updatePlayView(){

if(song is playing){

change image to pause

} else {

Change image to play

}

}

}

**PROJECT CLOSURE**

**Goals / Vision :-**

Our main goal was to retrieve the data from APIs and using that data, download the converted songs to the user’s local device. But this was not possible due to the use of the Third-Party APIs. Therefore, we envision to use a custom server that could directly convert the data and send to the application. Also, we think that adding an ML Model would help in the improving suggestions of the user.

**Delivered Solution :-**

We have tried our best to develop the application as much as possible but there have been some barriers that cannot be crossed now. Like the possibility of Google verifying our application to access user’s data. Therefore, we have tried to depict the best possible way this application should work with some test data. The user can use the application as a music player. The user can see the song being downloaded to the device and can be accessed in the offline mode. We have made an application that depicts how the application would work.

**Remaining Work :-**

We are looking forward to retrieve user’s data from major APIs that would help us in generating the mp3 file from a custom server. This application could come to a great success if we could use a specific server that could return the converted files to the devices after requests. Therefore, we would like to do these two things first.

**REFERENCES**

1. UI/UX Design References
2. https://www.vectorstock.com/royalty-free-vector/liquid-background-best-music-player-ui-ux-gui-vector-25958868
3. https://dribbble.com/shots/9977109-CCA-App-Splash-Screen-Video-Player
4. https://dribbble.com/shots/6719410-Music-Player-App-Concept
5. https://app.moqups.com/o6BDLvhuem/edit/page/a88be41aa
6. Icons
7. https://www.vectorstock.com/royalty-free-vector/liquid-background-best-music-player-ui-ux-gui-vector-25958868
8. https://www.flaticon.com

3. Apple Developers Reference

1. https://developer.apple.com/av-foundation/
2. https://developer.apple.com/library/archive/samplecode/AVFoundationSimplePlayer-iOS/Introduction/Intro.html
3. https://developer.apple.com/documentation/avfoundation

4. GitHub Code Reference

1. https://github.com/bpolat/Music-Player/blob/master/Music%20Player/PlayerViewController.swift

5. API References

1. https://mobikul.com/play-audio-file-save-document-directory-ios-swift/
2. https://github.com/devMadrit/MKYoutubeMp3Downloader