CI660 – ADVANCE MOBILE APPLICATION DEVELOPMENT

Vickshan Vicknakumaran (student)
[Email address]

Contents

Introduction

The application created is a music streaming service for android which allows users to browse and listen to songs. With the growing popularity of music streaming services, creating a music streaming application gives me the opportunity to explore the challenges and the complexities involved in developing a functional and user-friendly app within a limited timeframe. During the development process, a lot of aspects were considered such as the mobile user interface design, the advance technical features which may need to be implemented, testing and users' interaction.

This report will cover key elements of the application, the development of the mobile user interface design, advanced technical features, the technical challenges faced, and the strategies applied to overcome them. The report will also discuss the testing and evaluation methodologies, project management approaches and provide a reflection on the overall development process.

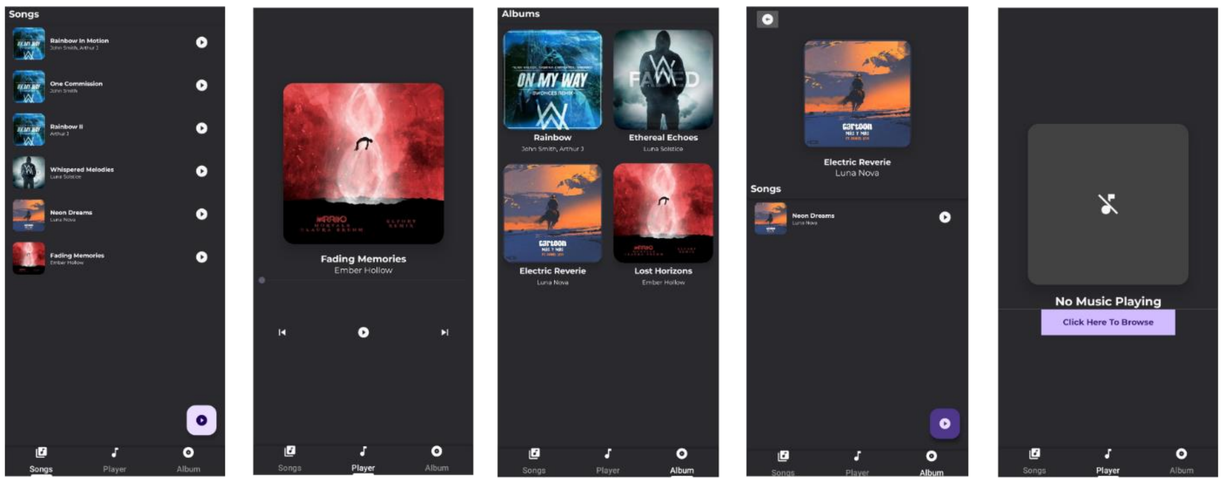


Figure 1 - All Screens of the Application

The application developed meets the criteria for the following two application features:

- **Advanced User Interface:** The user Interface of this application consists of high level of difficulties and technical features. Provides a visually appealing user interface and incorporates complex interactions and functionalities to enhance user engagement.
- **Storage:** The application utilises Firebase Database for data storage enabling the application to request and retrieve music data from a real time database.

Further Research

Software architecture

The application follows the Model-View-Controller (MVC) architectural pattern. The architecture will therefore follow the following:

- **Model:** This will represent the data logic of the application which includes the fetching of albums and songs data from Firebase Database.
- **View:** This will consist of the Fragments (SongsListFragment, MusicPlayerFragment, AlbumListFragment, AlbumViewFragment and AlbumsFragment) which are responsible for displaying the user interface and handling user interface.

- **Controller:** This will handle the logic for navigating between fragments using viewer pager, passing data between fragments, and controlling the playback of songs in the MusicPlayerFragment.

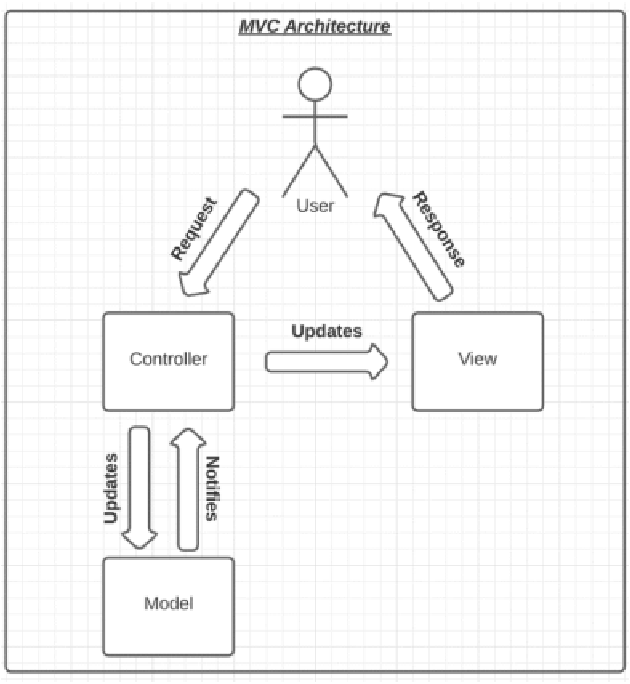


Figure 2 - MVC Architecture

UML Case Scenarios

Table 1 - Use Case Scenario: Playing a Song from the SongsListFragment

Name	Playing a Song from the SongsListFragment
Brief Description	When user selects a song in the Songs Tab
Actor(s)	**User**
Flow of Events	
Basic Scenario	
1. User selects a song from the list. 2. SongsListFragment sends the song key to MusicPlayerFragment. 3. MusicPlayerFragment updates the UI to display the song details. 4. User is switched from Song Tab to Player Tab Automatically 5. MusicPlayerFragment initiates song playback.	
Alternate Scenario	
Title	**Description**
Currently Playing Song	If a song is currently playing, the musicplayerfragment will clear the current playlist and populate with the new playlist based on user's request
Pre-Conditions	
Title	**Description**
Songs List	This Fragment must display the songs list from firebase

Post Conditions	
Title	**Description**
Success	The user is able to listen to the selected song

Table 2 - Use Case Scenario: Playing all Songs from the SongsListFragment

Name	Playing All Songs from the SongsListFragment
Brief Description	When user clicks the play all floating button
Actor(s)	**User**
Flow of Events	
Basic Scenario	
1. User clicks the "Play All" floating button. 2. SongsListFragment retrieves all song keys and bundles them into an array. 3. SongsListFragment sends the array to MusicPlayerFragment. 4. MusicPlayerFragment creates a playlist with the songs in the array. 5. MusicPlayerFragment plays the songs in sequential order.	
Alternate Scenario	
Title	**Description**
Currently Playing Song or list of songs	If a song or list of songs are currently playing, the musicplayerfragment will clear the current playlist and populate with the new playlist based on user's request
Pre-Conditions	
Title	**Description**
Songs List	This Fragment must display the songs list from firebase
Post Conditions	
Title	**Description**
Success	The user can listen to the selected song in sequential order.

Table 3 - Use Case Scenario: View Album Details from Album List

Name	Viewing Album Details from the AlbumListFragment
Brief Description	When user clicks an Album from the Album List
Actor(s)	**User**
Flow of Events	
Basic Scenario	
1. User selects an album from the list. 2. AlbumListFragment sends the album ID to AlbumViewFragment. 3. AlbumViewFragment replaces the Fragment Container and displays the album details through the AlbumsFragment. 4. AlbumFragment loads and displays the associated songs filtered by the album ID using the SongFragment.	
Alternate Scenario	
Title	**Description**

Currently Playing Song or list of songs	If a song or list of songs are currently playing, the musicplayerfragment will clear the current playlist and populate with the new playlist based on user's request
Pre-Conditions	
Title	**Description**
AlbumListFragment	AlbumListFragment is displayed
Post Conditions	
Title	**Description**
Success	The user can view the album details with the album tracks.

Table 4 - Use Case Scenario: Displaying Album Song Tracks

Name	SongsListFragment filtering
Brief Description	Displaying Album Song Tracks
Actor(s)	**System**
Flow of Events	
Basic Scenario	
1. When AlbumsFragment is displayed the albumID should send to the SongsListFragment so it can update 2. Display the filtered list	
Alternate Scenario	
Title	**Description**
N/A	N/A
Pre-Conditions	
Title	**Description**
AlbumId	AlbumID must be sent to SongListFragment for filtering
Post Conditions	
Title	**Description**
Success	The user can view the album details with the album tracks.

Mobile Context and User Experience

Mobile Context

Mobile Context surrounds the concept of specific characteristics and limitations and advantages they offer. It also talks about the unique set of constraints and opportunities that are inherent to mobile devices. It is a must that mobile applications need to be designed with the understanding of context so that optimal performance and user experience can be achieved.

Constraints are one of the important aspects in terms of Mobile context when it is imposed by mobile device such as limited screen size, processing power and battery life. Due to these limitations, developers are required to optimise their application so that resource usage and smooth performance are ensured efficiently on mobile platforms. In (Z. Zhuang, 2010), the paper talks about popularity of location-based application and the active use of these type of applications cause devices to drain battery. The paper goes to talk about an adaptive location sensing framework that significantly improves the energy efficiency of smartphones running location-based applications. The framework came out with positive results with it reducing the usage of the power by up to 98% and improve battery life by up to 75%.

Mobile Context also shows the opportunities for enhancing user experience. Touch input, location services and push notifications are unique features offered by mobile devices which can create engaging and personalised experiences. (Anind K. Dey, 2001) . By using these features effectively, developers can deliver contextual information in real-time and provide seamless integration with the user’s environment.

User Experience

User Experience focuses on the user’s interaction and how they interact with the application. It takes in account many various aspects including usability, visual appeal, accessibility, and responsiveness which all contribute to the satisfaction of users.

In UX design, important consideration is taken in designing the navigation. A Mobile application should have a clear navigation structure that allow users to easily navigate and find features. Elements such as tab layout and side menus help users understand the app’s structure. (Rina Trismininingsih, 2019)

Another crucial aspect of UX is responsive design. To ensure that the content and app’s layout adapt to different screen sizes and orientation. This will also ensure that there is consistency and visually appealing experience across devices. The UX design should offer users with visual cues and feedback to guide users’ action and provide confirmation. Cues could include animations, loading indicators and haptic feedback which enhance the responsiveness of the app and improve user satisfaction.

Usability plays a crucial role in UX design as this directly impacts user satisfaction, engagement, and the overall user experience. A study by (Bargas-Avila, 2011) found that applications which were highly usable had higher user satisfaction and increased task efficiency. When the application developed in mind with usability, it enables users to easily learn and navigate the interface. Due to this, users will likely use this app continuously which will contribute to user retention.

Incorporating Mobile Context and User Experience into the project

The music streaming application has been designed and developed in a way that it incorporates **mobile context and user experience** with consideration of the unique characteristics and constraints mobile devices. The mobile design follows best practices and guidelines which is, so the user is provided with an enjoyable and intuitive experience. This section will explain the key aspects of how mobile context and user experience were integrated into the application.

The use of a **responsive design** was prioritised to ensued that the app would adapt seamlessly to various screen sizes found in mobile devices. By using responsive layout and scalable visual elements, it will aim to deliver an optimal viewing experience for users across different mobile devices.

In terms of navigation, a tab layout, which provided clear labels and visually distinct icons, was used to allow users to easily switch between different pages enabling users to understand the purpose of each tab and seamlessly navigating through.

During the development process, **performance optimisation** was played a key role. The use of real time Firebase database as a data storer was an efficient way to fetch song list and album details. By minimising retrieval times, it provides a smooth and responsive experience for the users. Furthermore, the use of the glide library was used to efficiently fetch and load images. Glide is a popular image loading and caching library which automatically handles image resisting and caching, optimising the loading process and reducing bandwidth usage. (Glide, n.d.)

One of other priorities was to reduce interactions with the app. For example, when a user selects a song from the SongsListFragment, the MusicPlayerFragment immediately displays the song details, switches from the current tab to the player tab and start playback. This seamless transition eliminates unnecessary steps.

The use of Material Design was used for all screens which provided a clean and organised display with each item represented as a separate card. Curved radius and elevation to cards was used to create a sense of depth and make it visually pleasing.

To make the app more interactive, touch-responsive animation were used. When a user taped on a song, album or play button, a ripple effect will appear. This visual feedback helps users understand interaction that they have made and provide a sense of control and responsiveness. (Material Design 3 , 2023)

Development

Main Activity

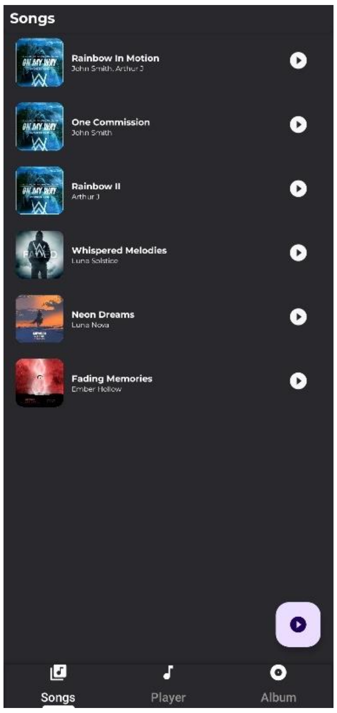


Figure 3 - MainActivity.java

Overview

The MainActivity plays a critical role in the application. The MainActivity provides users with interface with various features. When launching the app, the user is presented with a visually appealing layout that consists of a ViewPager2 and a TabLayout at the bottom which acts as the navigation.

Technical Overview

The ‘MainActivity’ class serves as the entry point for the application. The ‘ViewPager2’ and ‘TabLayout’ objects are then instantiated and assigned to.

The ViewPager2 is used to allow users to swipe horizontally between the main three fragments: SongsListFragment, MusicPlayerFragment and AlbumViewFragment. Each fragment corresponds to the TabLayout. This provides easy access for users to use the different functionalities. To highlight to the user which fragment they are at, the text colour is changed to white.

To manage the fragments in the ViewPager2, a TabAdapter is created which extends to FragmentStateAdapter. The adapter is responsible for the creation and the management of fragments. The TabAdapter initialises a list of fragments according to the navigation bar. To pass bundles across fragment, methods are made under the TabAdapter which will update accordingly. The ‘TabLayoutMediator’ is used to assign and synchronise the ‘TabLayout’ with the ‘ViewPager2’. It binds the tabs text and icons based on the positions.

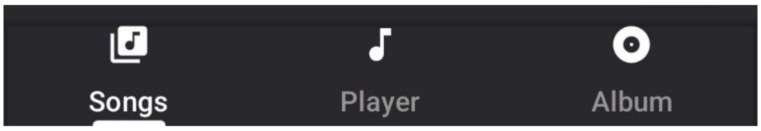
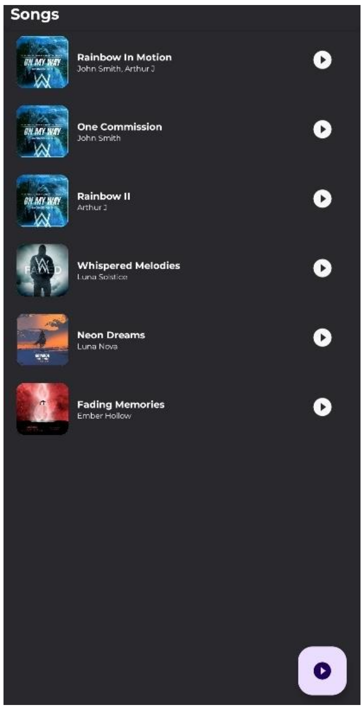


Figure 4 - Tab Layout Navigation Bar

Song List Fragment



Rainbow In Motion
John Smith, Arthur J

One Commission
John Smith

Rainbow II
Arthur J

Whispered Melodies
Luna Solstice

Neon Dreams
Luna Nova

Fading Memories
Ember Hollow

Figure 5 - Song List Fragment

Overview

The SongListFragment provides a visual appealing and interactive interface for browsing and playing songs. It integrates with the Firebase Realtime Database to fetch the songs details. The SongListFragment was developed in mind of using it for multiple purposes the list as follows:

- **Displaying a List of Songs:** The SongListFragment displays a list of songs in a user-friendly manner. Each item in the RecyclerView represents a song showing the songs's name, artist and album art.
- **Filtering Songs:** The Song List can be filtered based on an albumID or containing a particular text. The fragment retrieves the songs that match specific criteria from the Database and updates the list accordingly.
- **Playing All Songs:** When the "Play All Button" is clicked which represented by a floating button, the user can play all the songs in the list. The fragment collects the song keys from the songsList and sends them to the MusicPlayerFragment through a bundle. It then switches to the MusicPlayerFragment tab where the song will be played.
- **Playing a Specific Song:** When a user clicks on an individual song in the list, it allows the user to play that specific song. The fragment retrieves the selected songs songKey and creates bundle which contains the key. It then updates the MusicPlayerFragment with the bundles and switches to MusicPlayerFragment automatically.
- **Seamless Navigation:** The SongListFragment is part of a ViewPager2, allowing the user to navigate between different fragments smoothly. The fragment can be swiped or accessed through tabs to switch between the SongListFragment, MusicPlayerFragment, AlbumListFragment.

Technical Overview

The SongListFragment is responsible for displaying a list of songs. It populates the RecyclerView with the fetched data. It also provides the functionality to play all songs and handle item click events.

The Key methods and features of the SongListFragment are as follows:

- **'fetchSongsFromFirebase(Bundle bundle)':** This method retrieves songs from the Firebase Realtime Database based and it is responsible for clearing and adding the songsList. Finally, it updates the RecyclerView adapter with the filtered song list if filter is needed else the whole song list shows.

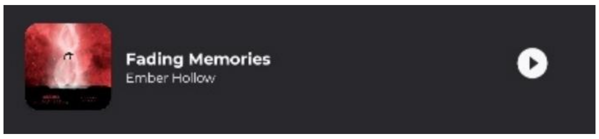


Figure 6 - Song Item

- **'checkSongCriteria(Song song, int albumId, String searchText)':** This methods check if a song meets the specific criteria. It checks whether albumID or SearchText with the field of the song object and returns true if the song matches the criteria.
- **'playAllSongs()':** The method is called when the PlayAll button is clicked. It creats a list of song keys from the songsList and bundles it to be sent to the MusicPlayerFragment through the TabAdapter. It then sets the current item of the ViewPager2 to the MusicPlayerFragment.

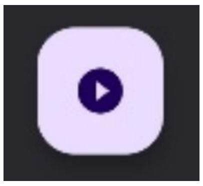


Figure 7 - PlayAll Floating Button

- **‘onItemClick’:** This method is invoked when an item in the RecyclerView is clicked. It retrieves the clicked song's songKey and creates a bundle containing the songKey. It updates the MusicPlayerFragment in the TabAdapter and sets the current item of the ViewPager2 to the MusicPlayerFragment.

Music Player Fragment

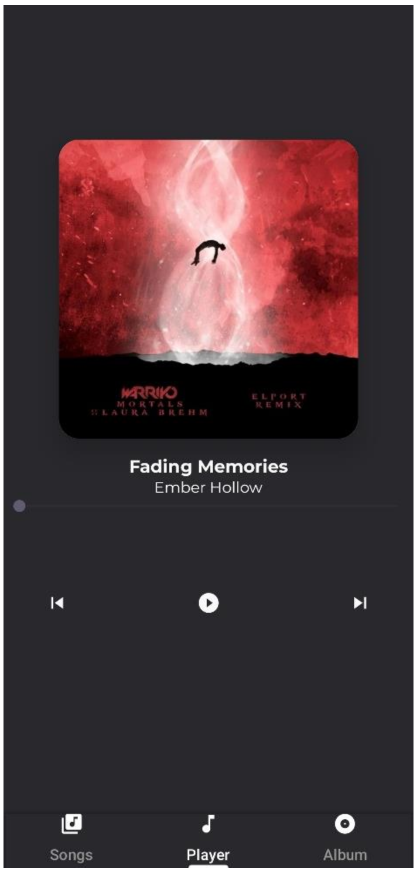


Figure 8 - Music Player Fragment

Overview

The ‘MusicPlayerFragmennt’ is a crucial component of the android application. It provides user-friendly interface for playing and managing music playback. When there no music playing, the layout changes to no music player layout.

At the top of the screen, users can see the song's title, artist's name display. Above that, an album art image is shown in cardview.

To control the music playback, the fragment includes intuitive playback controls. Users can pause or play the currently playing song by tapping the play/pause button. There are also buttons to skip to the previous or next song in the playlist, allowing users to navigate through the playlist.

A seek bar is positioned at the bottom of the fragment, allowing users to control the progress of the current song to their favourite position. Users can easily slide the seek bar to any desired position to jump to a specific part of the song. The seek bar also displays the elapsed and remaining time of the song, giving users a sense of their progress within the track.

Furthermore, the fragment includes a loading progress bar, which indicates the buffering status when loading a song. This visual feedback helps users understand that the application is fetching the audio data and preparing it for playback.

Technical Overview

The fragment provides the functionality for controlling music playback and displaying song details. The key technical features are the following:

- **MediaPlayer:** The class uses the Android MediaPlayer API to handle audio playback. It initialises a MediaPlayer object and implements various listener interfaces to respond to events such as preparation, errors, completion, etc.
- **Firebase Integration:** The code interacts with the Firebase Realtime Database to fetch song details based on the SongKeys which is provided by the bundle. It creates a database reference and listens for changes in the data using a ValueEventListener. It retrieves song information such as album, artist, name, URL, and album art from the database.
- **Glide Library:** The Glide library is used to load and display album art images. It is integrated to load images from the provided URL using the Glide.with() method.
- **Playback Controls:** The class provides functionality for controlling music playback. It handles button clicks for play/pause, previous, and next actions. The playPauseSong() method toggles between play and pause states. The playPreviousSong() and playNextSong() methods handle song navigation within the playlist.
- **SeekBar and Timer**: The SeekBar component is used to display and control the progress of the currently playing song. The class implements the SeekBar.OnSeekBarChangeListener interface to respond to changes in the seek bar position. A Timer is used to update the seek bar progress at regular intervals.
- **UI Updates:** The updateSongDetails() method updates the UI components with the details of the currently playing song, such as the song name, artist, and album art.
- **If the bundle is null:** The MusicPlayer checks if the bundle is null, this means whether a song has been requested. If the MusicPlayer is null then the player will inflate the no music playing layout.

Album List Fragment

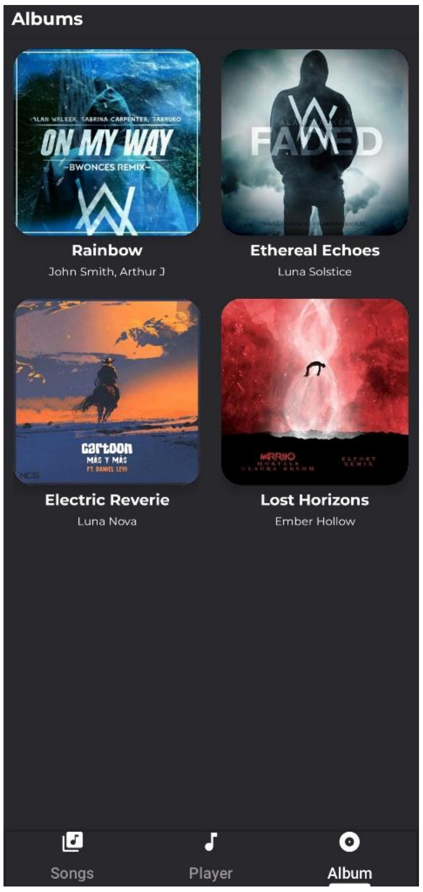


Figure 9 - Album List Fragment

Overview

This fragment is accessed through albums tab, and it displays the albums as a grid layout, showcasing the album name, artist and album art. The user can scroll vertically through the album list and the grid layout ensures that multiple albums are visible for easy browsing. Each album item is interactive. When user selects a specific album, the fragment (AlbumsFrgament) will be replacing the fragment container and it will display the album details.

Technical Overview

One of the significant elements in this fragment is the implementation of 'RecyclerView' as it is used to efficiently display the list of albums in a scrollable grid layout. To retrieve the album data, the Firebase Realtime Database is used and fetches the album using the 'FirebaseDatabase.getInstance().getReference("album")'. When an album item is clicked, the 'onItemClick' method is triggered. A bundle would be created which will consist of the albumID. The 'AlbumsFragment' replaces the fragment container which is in the AlbumViewFragment and passes the bundle.

Albums Fragment

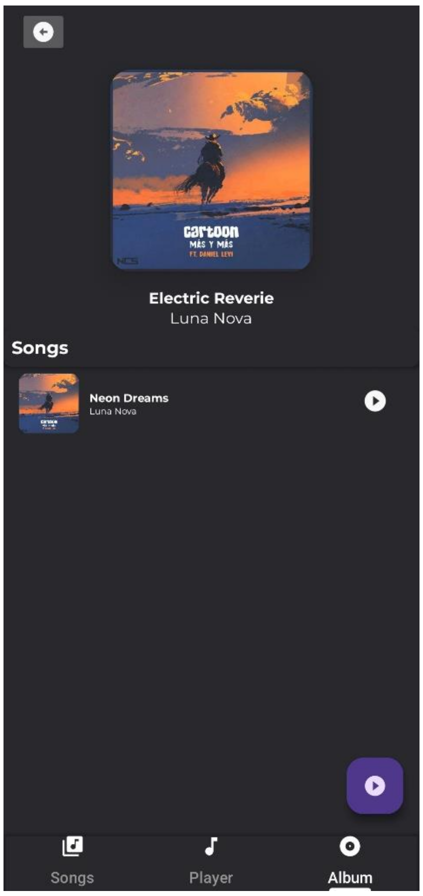


Figure 10 - Albums Fragment

Overview

When the user is shows the 'AlbumsFragment', they are displayed with the details of the specific album which include the album name, artist name and album art. At the top of the fragment, there is a back button represented by an image button. When the user clicks this button, they are taken back to the previous screen, which is most likely the 'AlbumListFragment' that lists all the albums. Below the back button, the album art is displayed in an image view which nested inside a card view which will help create a material design. The fragment retrieves the album details from the database ensuring that the displayed information is accurate. Below these details, there is a fragment container which consists of the songfragment. The bundle will be sent to the container with albumID to the container. This is that it will only show the list of songs which consists in the specific album clicked.

Technical Overview

The 'OnCreate()' method retrieves the argument passed to it from the bundle. It extracts the 'albumId' from the bundle, which is used to indicate the ID of the album to be displayed. The 'retrieveAlbumDetails()' methods is called to fetch the album's details of the album from the database and this is attached to the 'ValueEventListner' to listen for changes in the data. When the data changes, the 'onDataChange()' method is triggered. It retrieves the albumart, album name and album artist from the DataSnapshot. The Picasso library is used for a smooth and efficient loading of the album art (Picasso Library, n.d.). The 'navigateToSongsFragment()' methos is called to replace fragment container in albumsfragment with the bundle argument which consists of the albumID.

This is so that the SongListFragment can show a filtered list which only shows the tracks from the albumID in the container.

Project Management

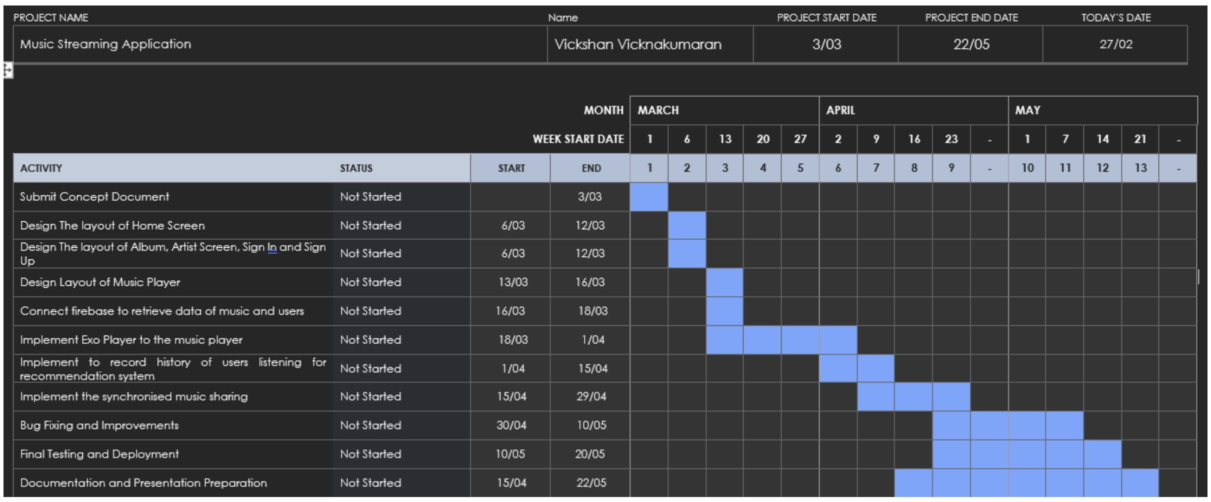
Agile methodology approach was used to successfully complete the project. The approach allowed for a flexible and iterative approach which allowed for change in requirements and continuously improve the application. The iterative approach help ensure that the final product met the expectation.

To ensure that I had a well-structured development process, I first worked on creating the core components of the application. This included setting up the Firebase, implementing core functionality such as Listing of songs, the items which consist in the recyclerview, and layout. By focusing on these elements, it was an ease in creating the fragments and allowed me to crate a solid framework for features implementations.

One of the core components was development of individual fragments such as SongListFragment and AlbumListFragment. This allowed me to focus on specific features and user interaction in more organised manner. I had made sure that each fragment was fully functional before moving on to the next fragment. This approach allowed me to identify any errors and resolve any bugs early into the project which resulted into having a smoother development process.

I wanted to avoid having any feature half done and so I had put importance into the functionality and the completeness of them. I made sure that each function was thoroughly tested and validated each feature before considering it as completed. This approach ensured that the implemented functionality was higher quality which also provides a more satisfying user experience.

To mitigate the risk of code issues or data loss, version control system was implemented and was regularly backed up different versions of the application. Having multiple versions of the app saved provided extra protection and allowed me to revert to previous versions of the application if any issue occurred. The use of version control helps me code new features knowing that I had the ability to restore the project to its working state if any issue occurred. It is also a reliable backup mechanism which will go against any technical failures and/or accidental losses.



PROJECT NAME	Name	PROJECT START DATE	PROJECT END DATE	TODAY'S DATE
Music Streaming Application	Vickshan Vicknakumaran	3/03	22/05	27/02

			MONTH	MARCH					APRIL					MAY				
			WEEK START DATE	1	6	13	20	27	2	9	16	23	-	1	7	14	21	-
ACTIVITY	STATUS	START	END	1	2	3	4	5	6	7	8	9	-	10	11	12	13	-
Submit Concept Document	Not Started		3/03	🟦														
Design The layout of Home Screen	Not Started	6/03	12/03		🟦													
Design The layout of Album, Artist Screen, Sign In and Sign Up	Not Started	6/03	12/03		🟦													
Design Layout of Music Player	Not Started	13/03	16/03			🟦												
Connect firebase to retrieve data of music and users	Not Started	16/03	18/03			🟦												
Implement Exo Player to the music player	Not Started	18/03	1/04			🟦	🟦	🟦	🟦									
Implement to record history of users listening for recommendation system	Not Started	1/04	15/04						🟦	🟦								
Implement the synchronised music sharing	Not Started	15/04	29/04							🟦	🟦	🟦						
Bug Fixing and Improvements	Not Started	30/04	10/05									🟦	🟦	🟦	🟦			
Final Testing and Deployment	Not Started	10/05	20/05									🟦	🟦	🟦	🟦	🟦		
Documentation and Presentation Preparation	Not Started	15/04	22/05								🟦	🟦	🟦	🟦	🟦	🟦	🟦	

Figure 11 - Gantt chart from proposal

In the proposal report, I had presented Gantt chart that outlined the project timeline and its allocated task. However, I encountered difficulties in following with the timeline due to several factors, including the complexity of programming certain features, encountering unexpected bugs and error and challenges faces when implementing some key features. As a result, some tasks took longer than initially estimated which caused me to deviate from the planned schedule.

One key lesson I learnt from this experience is the importance of breaking down tasks into smaller section and with more manageable tasks. While the initial Gantt chart helped provide an overall structure of the project, I found that separating complex features into smaller subtasks would have been more achievable and allowed me to plan the overall project which would have worked in the time constraints.

Furthermore, given the time constraints and the need to complete other projects, I realised that I should be thinking about setting realistic and achievable goals which is very crucial. If the complexity of the features had been considered prior to the development of the project, it would have allowed me to priories time for critical project areas.

Testing

To ensure that the quality and functionality of the application, I conducted through testing using various tools and techniques. The testing phase involves both automated and manual test, focusing on different aspects of the app's behaviour and performance.

Functional Testing

The aim of functional testing is to validate the application's behaviour and functionality requirements. The test focuses on the aspects that are critical to application functionality such as bundle creation and passing, array handling, and error management using Logcat. During testing I faced a lot of issues with the logcat because in certain scenarios the logcat would not display any information which causes a lot of time to be wasted since the logcat is used to find errors.

Table 5- Black Box Testing

Test Scenario	Test Case ID	Test Name	Test Steps	Results
Bundles Creation and Passing	TC001	Bundle Creation	1. Create a bundle object with a specific integer value for the "SongKey" key. 2. Verify if the bundle contains the SongKey value and comparing it to the expected value	FAIL
	TC002	Bundle Creation with code modifications		PASS
	TC003	Bundle Passing	1. Pass the bundle from SongListFragment to MusicPlayerfragment when an album is clicked. 2. In MusicPlayerFragment, retrieve the bundle using getArguments() and check if it contains the "SongKey" key and value.	PASS

			3. Verify that the retrieved "SongKey" matches the expected value.	
Array Testing	TC004	Array Operations	1. Perform various operations on the albumList and songList array, such as adding , removing, and accessing albums and songs by index. 2. Verify that the array behaves as expected by checking the size, content, and order of albums after each operation.	PASS
Error Testing with Logcat	TC005	Logcat Viewing	1. Run the app and monitor Logcat output for any runtime errors or exceptions. 2. Analyze the log messages to identify any errors or exceptions. 3. Resolve any identified errors or exceptions by debugging and fixing the code.	PASS
Emulator Testing	TC006	Emulator Testing	1. Depoly and run the app on both Android emulator and Android physical phone. 2. Verify the app's behaviour, responsiveness, and compatibility on the emulator	

Device Testing

Device testing involves the evaluation of the application's behaviour and performance on both Android Smart Phone and Android emulator.

Android & Emulator Smartphone Testing

The developer tool in android provides many testing tools which help us as developers into ensuring that the application is working as it should. Both Android and Emulator testing was done but I had preferred the android physical device testing as the emulator was very slow and lagged on my computer which is not very good when it comes to testing. However, I used to emulator if my wireless debugging did not work.

The first tool I used is the "Show layout bounds" which show clip bounds. Margins and many more.

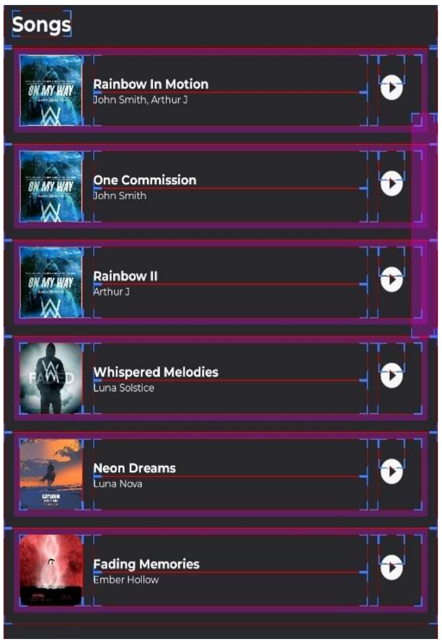


Figure 12 - Layout bound for SongListFragment

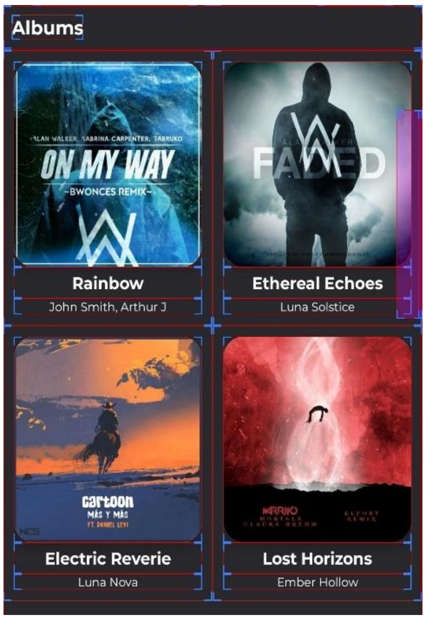


Figure 13 - Layout bound for AlbumListFragment

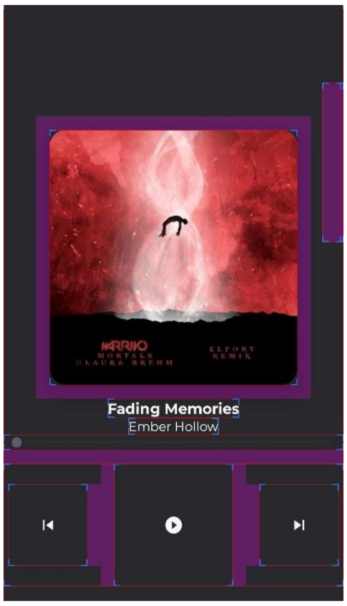


Figure 14 - Layout bound for MusicPlayerFragment

I also used the refresh rate indicator feature which indicates the refresh rate of the app whilst browsing which it stays consistent throughout the app.

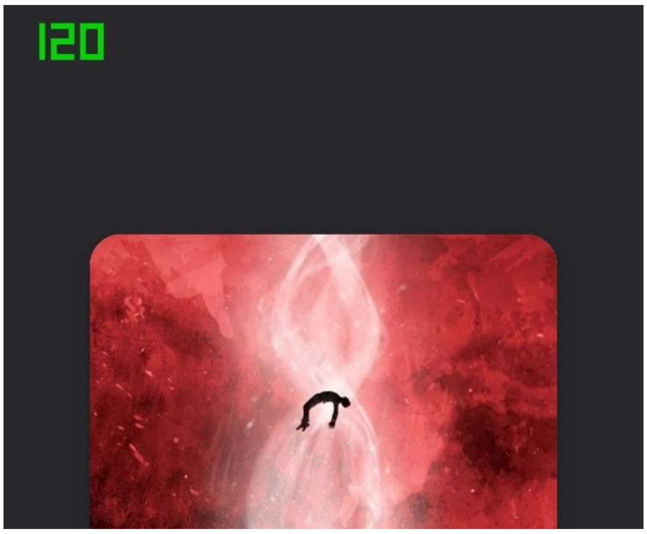


Figure 15 - Refresh Rate counter on top right

Evaluation

Project Outcomes

The project is evaluated based on the achieved functionality, user experience and project requirements. The following aspects were considered:

- **Functionality:** The core functionalities were implemented successfully for the application such as browsing albums, viewing album details, viewing song list, and playing music. The implemented features provided users with the must needed functionality and met the project requirements.

- **User Experience:** The application focused on delivering a user-friendly and accessible user interface.
- **Requirements:** The application mostly follows the specific requirements outlined at the beginning of the project. The features implemented align with the initial project scope. This ensures that the intended functionalities were met.

Achievement of Objectives

Evaluation of the project objectives determine the extent to which they were achieved. The following objectives were set:

- **Objective 1 – Develop a Music Application:** The application successfully achieved this objective by providing users to browse songs and albums. Application also achieves the goal with the creation of the Music Player which plays one or multiple songs.
- **Objective 2 – User Interface:** The user interface aims to be intuitive enabling users to easily navigate through each fragment with the user of a navigation bar.
- **Objective 3 – Ensures Functionality**: The application was thoroughly tested to ensure that the functionality was working correctly. Bugs and errors were addressed and resolved accordingly.

Strengths and Areas for Improvement

During the evaluation, the following strength and improvement were identified:

Strengths:

- The application provides a clear and advance user interface enhancing the overall user experience.
- The app can be updated real-time due to the use of firebase database.
- Successful implementation of core functionalities which provides a function music streaming application where they can browse and listen to music.

Areas for Improvement:

- The development timeline was not followed due to many external factors and difficulties in implementing certain features.
- Further user testing and feedback would allow me to make modifications based on user's feedback.

Lessons Learned:

Throughout the project, I learned many valuable lessons:

Development

During the development process I had learned many new advanced user interface functionalities. For example, the use of fragments was a first time for me so there were many ways which I implemented fragment such as using a fragment container viewer and using ViewPager2.

Effective Time Management

The project highlighted the importance of setting realistic task and manging time efficiently. Due to setting task which require a lot of time and complexity. I was unable to achieve every feature which I had in mind. However, I had completed the core functionalities. If I had separated the tasks into smaller sub section, I would managed time more efficiently

Firebase

This is my first time I am using my own database especially using a real time firebase database. It has been challenging as the queries are limited and does not work in the same way as a custom API. Even though the firebase provides a great way to access Real-time database, I would prefer to use my custom API which will have more flexibility and control in the content I want.

Future Developments

- **Machine Learning-based Song Recommendations:** Implementing machine learning algorithms to analyse user listening patterns and preferences can enhance the app's recommendation system. By leveraging data on users' past interactions and music preferences, the app can provide personalized song recommendations, improving user engagement and satisfaction.
- **Integration of Custom APIs:** Integrating custom APIs can enhance the application's functionality and provide access to a wider range of music-related data. For example, integrating an API that provides lyrics, artist information, or concert schedules can enrich the user experience and offer additional features to explore.
- **Integration of Custom APIs**: Integrating custom APIs can enhance the application's functionality and provide access to a wider range of music-related data. For example, integrating an API that provides lyrics, artist information, or concert schedules can enrich the user experience and offer additional features to explore.
- **Integration of Custom APIs**: Integrating custom APIs can enhance the application's functionality and provide access to a wider range of music-related data. For example, integrating an API that provides lyrics, artist information, or concert schedules can enrich the user experience and offer additional features to explore.
- **Integration of Custom APIs**: Integrating custom APIs can enhance the application's functionality and provide access to a wider range of music-related data. For example, integrating an API that provides lyrics, artist information, or concert schedules can enrich the user experience and offer additional features to explore.
- **Integration of Custom APIs**: Integrating custom APIs can enhance the application's functionality and provide access to a wider range of music-related data. For example, integrating an API that provides lyrics, artist information, or concert schedules can enrich the user experience and offer additional features to explore.
- **Integration of Custom APIs**: Integrating custom APIs can enhance the application's functionality and provide access to a wider range of music-related data. For example, integrating an API that provides lyrics, artist information, or concert schedules can enrich the user experience and offer additional features to explore.

References

Anind K. Dey, G. A. (2001). *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications.* Retrieved from https://www.researchgate.net/publication/2870717_A_Conceptual_Framework_and_a_Toolkit_for_Supporting_the_Rapid_Prototyping_of_Context-Aware_Applications

Bargas-Avila, J. A. (2011). *Old wine in new bottles or novel challenges: A critical analysis of empirical studies of user experience. In Proceedings of the SIGCHI conference on human factors in computing systems.*

Glide. (n.d.). Retrieved from Github: https://github.com/bumptech/glide

Material Design 3 . (2023). Retrieved from Google: https://m3.material.io/

Picasso Library. (n.d.). Retrieved from Picasso Library: https://square.github.io/picasso/

Rina Trisminingsih, D. N. (2019). *User Experience Design of Task-Management Application for Plantation Supervisor Using Lean UX.* Retrieved from https://ieeexplore.ieee.org/abstract/document/9166579

Z. Zhuang, K. K. (2010). *Improving energy efficiency of location sensing on smartphones.* Retrieved from https://dl.acm.org/doi/10.1145/1814433.1814464

Tables Of Figures