Harmony

Zai Erb, Harrison Reed, Kara Deskin, Ethan Doughty, Jack Pigott

Contents

1 Software Stack and Scope of the Project

1.1 Software Stack

1.1.1 Frontend

For the frontend of this project, we will use Flutter. We chose Flutter for its seamless integration with Firebase, thorough documentation, the abundance of available resources, extensive UI libraries, and features enabled with Firebase such as live database updates and push notifications. Additionally, Flutter allows for general cross-platform development with the option to supplement the Dart codebase with native code for platform-specific features.

1.1.2 Backend

For the backend of this project, we will use Firebase. We chose Firebase for its tight integration with Flutter, built-in quality-of-life features, and widely available documentation. We will use Firestore for our databases, which is natively supported within Firebase and is based on a NoSQL design philosophy. Firebase's built-in features will handle user authentication.

1.2 APIs

Based on our research, we will use APIs for Spotify, SoundCloud, YouTube Music, and Last.fm.

API	Group Member	Documentation	
Spotify	Ethan	Official	
Soundcloud	Ethan	Official	
Apple Music	Kara	Official	
Youtube	Jack	Unofficial	
Bandcamp	Zai	Official	Info
Last FM	Zai	Official	

	APIs							
Features	Spotify	Soundcloud	Apple Music	Youtube	Bandcamp	Last FM	All	
Requires Key	1	1	1	1	1	1		
Acess Requested	1	1	0	1	1	1		API Key Info
Key Obtained	1	0	0	1	0	1	0	
Audio Streaming	0	1	0	1	1	1	0	
Embedded Player	0	1	1	1	1	1	0	
Access to Account Features	1	1	1	1	1	1	1	Essential Feat.
Access to User Collection	1	1	1	1	1	1	1	
Access to Playlists	1	1	1	1	1	1	1	0
Track Info	1	1	1	1	1	1	1	
Track Title	1	1	1	1	1	1	1	
Track Length	1	1	1	1	1	1	1	
Track Album	1	1	1	1	1	1	1	Track Info
Track Length	1	1	1	1	1	1	1	
Track Release Date	1	1	1	1	1	1	1	
Additional Track Info	1	1	1	1	1	1	1	1
Album Info	1	Treated as playli	1	1	1	1	1	
Album Tracklist	1	1	1	1	1	1	1	
Album Artist	1	1	1	1	1	1	1	Album Info
Album Length	1	0	1	1	1	1	0	
Album Release Date	1	0	1	1	1	1	0	
Additional Album Info	1	0	1	1	1	1	0	0
Artist Info	1	1	1	1	1	1	1	
Artist Releases	1	1	1	1	1	1	1	Artist Info 1
Genre Info	1	1	1	0	1	1	0	
Tag Info	0	1	0	0	0	1	0	Additional Info
Image Acess	1	1	1	1	1	0	0	
Video Acess	0	0	1	0	0	0	0	0
INCLUDE IN PROJECT	1	1	0	1	0	1	4	Total APIs we will include

Figure 1: API Comparison

The scripts to call the given APIs are the first step in the data pipeline for our application and the foundation of our backend's data retrieval. Each API interaction is different, so each retrieval script will require an accompanying script to parse, arrange, and return the collected data in a standardized format. These API scripts will only be used to add data to the database, while the frontend will query the database with more specificity.

1.2.1 Standardized Format for Returning Data

Tracks GetTrack should return two objects: an image (cover art for the track) and a list formatted as follows:

- Title
- Release Date
- Length
- Album
- Artist

Albums GetAlbum should return three objects: an array of track objects (tracklist), an image (cover art), and a list formatted as follows:

- Title
- Release Date
- Length
- Artist

Artists GetArtist should return two objects: an image (artist picture) and a list formatted as follows:

- Name
- Number of Releases
- Genres (if available)

1.3 Included Data, Pages, and Queries

1.3.1 Data

Essential Features:

- Audio Streaming
- Platform Account Features
- Platform User Collections
- Platform Playlists
- Track Data:
  - Title
  - Length
  - Album
  - Release Date
  - Additional Track Info (TBD)
- Album Data:
  - Tracklists
  - Artist
- Artist Data:
  - Releases
  - Additional Artist Info (TBD)
- Additional Data:
  - Genre Data
  - Image Access

1.3.2 Pages

- Home Page - This page contains the main flow of the application, with routes to all other pages not contained in settings.
- Playlists - Contains nothing right now. (Will contain all of the current user's playlists connected across accounts.)
- Artists - Contains nothing right now. (Will contain all of the current user's artists connected across accounts.)
- Albums - Contains nothing right now. (Will contain all of the current user's albums connected across accounts.)
- Songs - Contains nothing right now. (Will contain all of the current user's songs connected across accounts.)
- Downloads - Contains nothing right now. (Will contain all of the current user's downloads connected across accounts.)
- Currently Playing - Contains nothing right now. (Will contain UI elements showing the currently playing song, with media controls.)
- Settings - Contains routes to My Account, Preferences, Connected Apps, and a button to sign out.
- My Account - Contains nothing right now. (Will contain the current user's account information.)
- Preferences - Contains nothing right now. (Will contain options for the current user to change their preferences.)
- Connected Apps - Contains nothing right now. (Will contain routes to login to various streaming services.)
- Future Pages - Need to include a login page that prompts users to login when not logged in.

1.3.3 Queries

We want to support queries that return objects containing all associated data while also allowing for queries to retrieve specific individual entries.

User Queries:

- Retrieve user as an object with all related data:
  - Username
  - Linked Accounts
- Retrieve user collection as an object:
  - Playlists
  - Saves
  - Favorites

Music Queries:

- Begin track audio playback
- Retrieve track as an object:
  - Source (service it was added from)
  - Title
  - Cover Art
  - Length
  - Genre(s)

  - Album
  - Release Date (nullable)
  - Additional Track Info (TBD)
- Retrieve album as an object:
  - Tracklists
  - Cover Art
  - Artist(s)
  - Genre(s)
- Retrieve artist as an object:
  - Releases
  - Picture
  - Genre(s)
  - Additional Artist Info (TBD)

2 Project Structure and General Responsibilities

2.1 GitHub Project Structure

- Main: Main branch, only merged with fully integration-tested features
- Full Stack Testing: Tests the integration of frontend and backend
- Frontend Testing: Reads from backend
- Backend Testing: Writes to backend
- API Testing: Isolated testing for API scripts
- Individual Testing Branches:
  - Zai
  - Harrison
  - Jack
  - Ethan
  - Kara

2.2 Group Responsibilities

Sprint 1: Due February 16th				
Task	**Group Member(s)**	**Priority**	**Est Time**	**Actual Time**
Determine Optimal Software Stack	All	1	2 hours	2 hours
Create doc to compare each apis capability	Zai	1	1 hour	1.5 hours
Build the inital application layout	Harrison	1	3 hours	4 hours
Research Spotify API and add to comparison doc	Ethan	1	1 hour	1 hour
Research Soundcloud API and add to comparison	Ethan	1	1 hour	1 hour
Research Youtube API and add to comparison	Jack	1	1 hour	1 hour
Research apple music API and add to comparison	Kara	1	1 hour	1 hour
Research bandcamp API and add to comparison	Zai	1	1 hour	1 hour
Research Last FM API and add to comparison	Zai	1	1 hour	1 hour
Determine which apis are feasible	All (based on comparison Doc)	1	3 hours	2 hours
Create branch structure in github project	Zai	1	20 minutes	20 minutes
Initialize firebase project and database framework	Harrison	2	1 hour	1 hour
Determine what pages are included for UI	Zai/All	1	30 minutes	1 hour
Make barebones UI for frontend	Harrison	2	6 hours	5 hours
Sprint 2 Requirements Stack	Zai	1	30 min	20 min
Sprint 2 Requirements Artifacts	Zai	1	1 hour	1 hour

Figure 2: Group Contribution

3 Barebones Application Layout

The code for the layout is written in Dart. You can find the code within the `src` directory.