Software Design Document

Section 1 - Project Description

1.1 Project

MusicPalette

1.2 Description

MusicPalette is modeled from Spotify Wrapped except it aims to allow users to view their music statistics on a daily, weekly, monthly, or yearly basis. It will utilize Spotify's API to retrieve and analyze user data, capturing listening statistics and preferences. MusicPalette helps users analyze their music listening habits while encouraging users to engage with their music and artists more deeply. We aim to create an engaging app for Spotify users that provides comprehensive statistics about users' music listening trends. We also want to make the product shareable and social to help users connect through music.

1.3 Revision History

Date	Comment	Author
10/3/2023	Initial Draft	Brett Denette Jose Lopez Aubry Mouanoutoua Praise Okoli Gurmanjot Singh Padda
10/8/2023	Revisions and formatting changes	Brett Denette Jose Lopez Aubry Mouanoutoua Praise Okoli Gurmanjot Singh Padda
10/17/2023	Updated Section 5.1	Brett Denette Jose Lopez Aubry Mouanoutoua Praise Okoli Gurmanjot Singh Padda
10/29/2023	Revisions and Updates	Brett Denette Jose Lopez Aubry Mouanoutoua Praise Okoli Gurmanjot Singh Padda
11/14/23	Revisions and Updates	Brett Denette Jose Lopez Aubry Mouanoutoua Praise Okoli Gurmanjot Singh Padda

Software Design Document

Contents

Section 2 - Overview

2.1 Purpose

The purpose of this module is to lay the foundation for the development and deployment of the MusicPalette application, a tool aimed at enhancing the experience of Spotify users by offering them detailed insights into their music listening trends on a more frequent basis.

2.2 Scope

This module will cover the key aspects of the MusicPalette project, encompassing the design, architecture, data dictionary, software domain design, user interface, and more, with an aim to provide a comprehensive blueprint for the application's development.

2.3 Requirements

Given the nature of the project, our primary requirements involve ensuring seamless integration with the Spotify API, user-friendly interface designs, efficient data management and processing, and shareability across platforms.

2.3.1 Estimates

#	Description	Hrs. Est.
1	Integration with Spotify AI	60
2	App User Interface Design	40
3	Backend Development & Database Management	80
4	Testing & Quality Assurance	30
	TOTAL:	210 # est tot

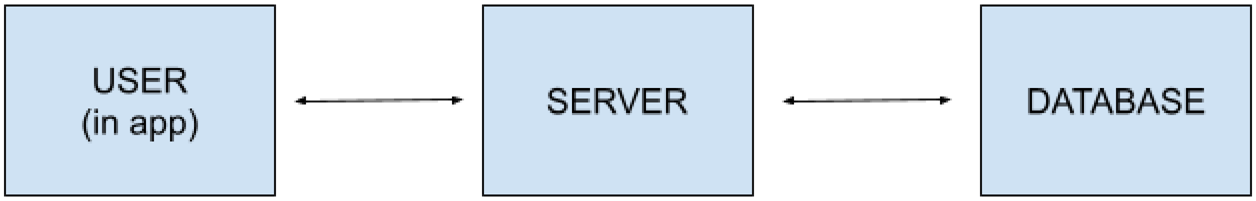
2.3.2 Traceability Matrix

Cross reference this document with your requirements document and link where you satisfy each requirement

SRS Requirement	SDD Module
Integration with Spotify API	5.2.1
App User Interface	7.1, 7.2, 7.3
Backend Development	5.2.1.1
Testing	8.1

Section 3 - System Architecture

The MusicPalette system will be designed as a client-server model, where the mobile app (client) communicates with a server, which interacts with the Spotify API to retrieve user data. The server will handle data processing and send the processed data back to the mobile app for display.



Section 4 - Data Dictionary

Table

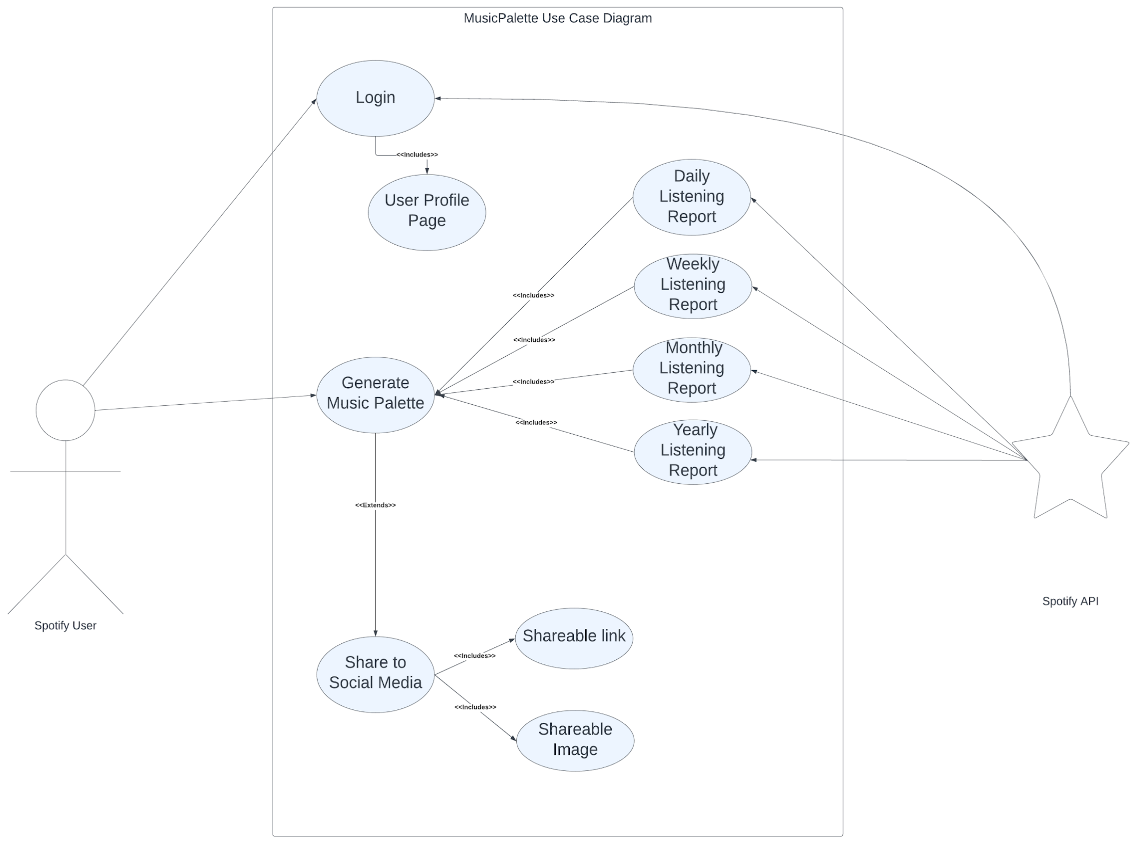
Field	Notes	Type
UserID	Unique Identifier for User	INTEGER
SongID	Unique Song Identifier from Spotify API	STRING
ListenCount	Number of times listened	INTEGER
ListenDate	Date of listening	DATE
SpotifyUser	Unique Identifier for User’s Spotify	STRING
ListeningReport	Music Listening Report Statistics	INTEGER
MusicPalette	Unique listening report and statistics for user	INTEGER
MusicTrack	Name of Music Track	STRING
Artist	Name of Music Artist	STRING
Shareable	Shareable link or image	INTEGER

Section 5 - Software Domain Design

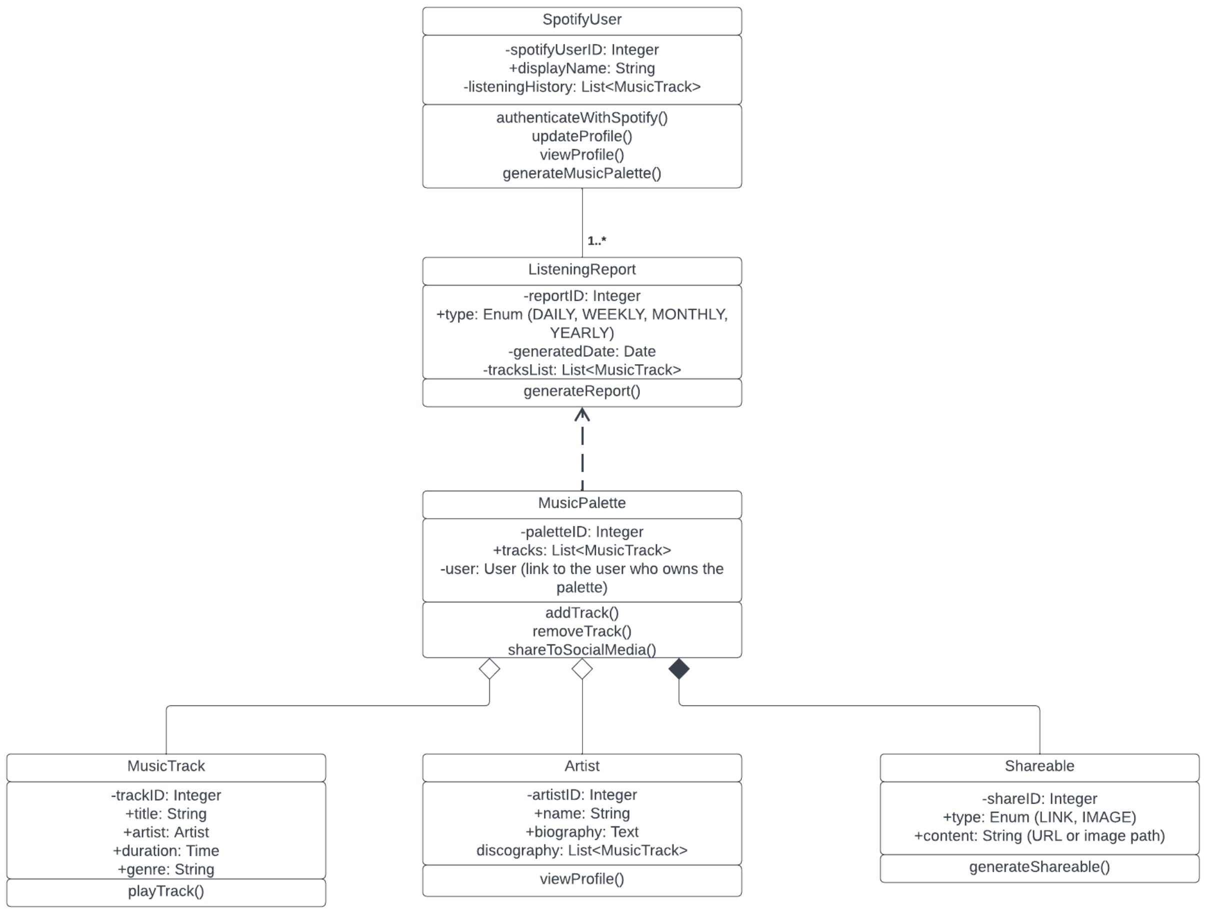
5.1 Software Application Domain Chart

The MusicPalette software application domain that is used is Spotify API.

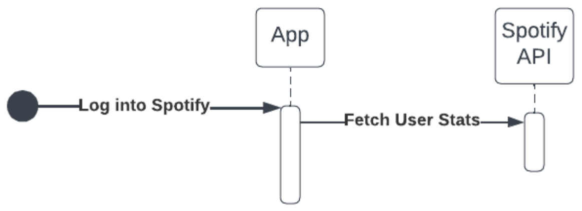
Use Case Diagram

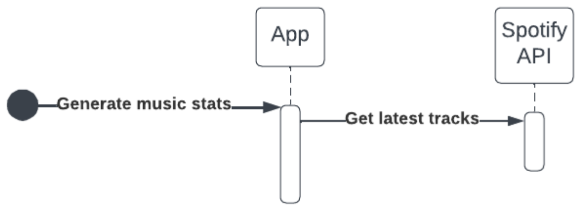


Class Diagram

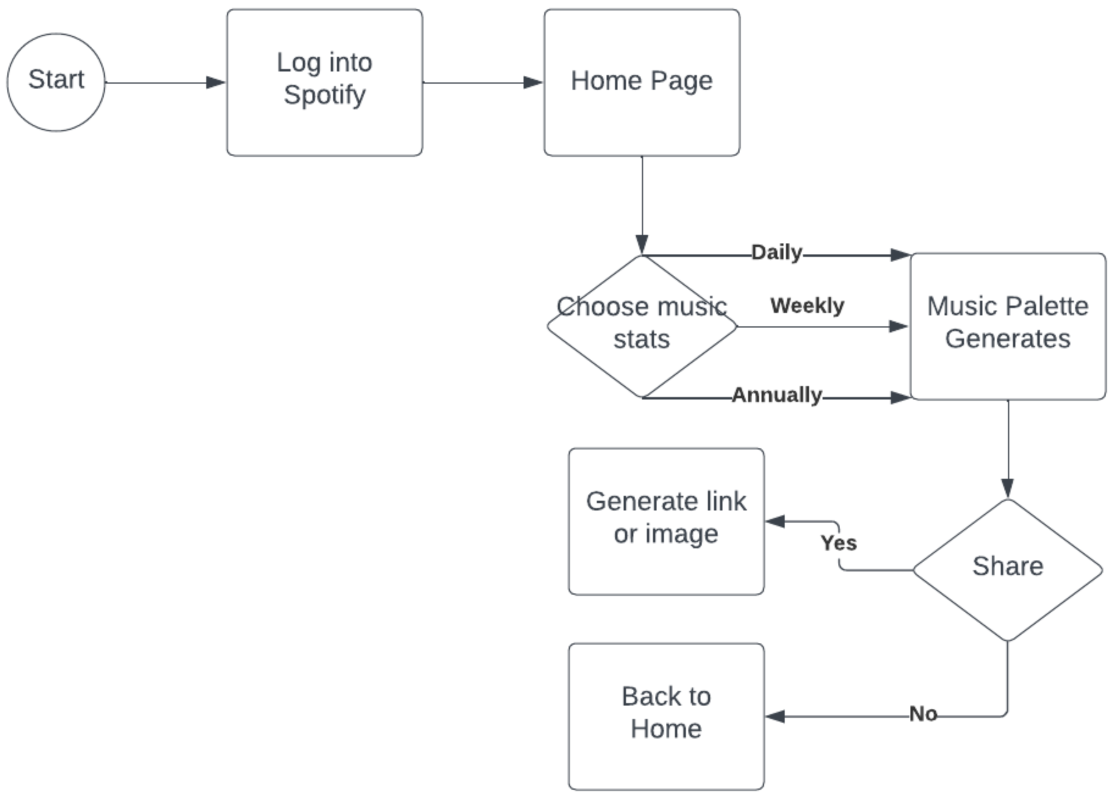


Sequence Diagram





Stateflow Diagram



5.2 Software Application Domain

The MusicPalette application revolves around key domains, each catering to different functionalities and interactions, which include User Interaction, Data Retrieval from Spotify, Data Storage, Data Analysis, and Data Sharing.

5.2.1 Domain X

This domain is crucial in interfacing with the Spotify API. It deals with authenticating the user, fetching listening data, and interfacing with Spotify's rich database to collect insights. Components within this domain include:

- Authentication Module: Handles user login via Spotify.
- API Call Module: Sends requests to Spotify's API endpoints to fetch user listening history.
- Data Fetch Scheduler: Schedules periodic fetching of data based on user settings (daily, weekly, etc.).

5.2.1.1 Component Y of Domain X

This component is responsible for ensuring that the user can securely log in via their Spotify credentials, allowing MusicPalette to access their listening data.

Data flow/control:

1) User initiates login.
2) Redirect to Spotify OAuth page.
3) User grants permission.
4) Receive authentication token.
5) Store token securely for subsequent API calls.

5.2.1.1.1 Task Z of Component Y1 of Domain X

The OAuth Redirect task manages the redirection of a user to Spotify's authentication page and subsequently handles the callback with an authentication token.
Data flow/control:

1) User clicks 'Login with Spotify'.
2) System redirects user to the Spotify OAuth page.
3) After permission, Spotify redirects back with a token.
4) Capture and securely store the token.

Section 6 – Data Design

The MusicPalette application primarily interacts with user-specific data fetched from the Spotify API. This data comprises the listening history, preferences, and user-specific metadata. This section delves into the intricacies of data design for the application.

6.1 Persistent/Static Data

6.1.1 Dataset

- User Profile: Contains information related to the Spotify user such as User ID, Name, Premium Status, Country, etc. This establishes the identity of the user in the app.
- Listening History: This dataset consists of user-specific listening records, capturing data points like Song ID, Date & Time of listening, Duration, etc.
- User Preferences: Categorizes user's genre preferences, favorite artists, most listened tracks, etc.

Relationships:

- User to Songs: One-to-Many, as one user can listen to multiple songs, but each song listening instance belongs to one user.
- Songs to Artists: Many-to-One, as a song is associated with one artist (or a band), but an artist can have multiple songs

6.1.2 Static Data

- Genres List: A predefined list of music genres fetched initially from Spotify, which helps classify songs and determine user preferences.
- App Configuration Data: Data related to the app’s configuration, settings, display

preferences, etc.

6.1.3 Persisted data

- Shared Links: Unique links or QR codes generated by users for sharing their music statistics. These links, once generated, are stored for future retrieval and access.
- User Statistics Summary: Post-analysis data, such as yearly or monthly summaries, top songs, or artists, might be persisted for quicker access without the need for reanalysis.

6.2 Transient/Dynamic Data

- Session Data: Information about the current user session, including tokens, temporary preferences, current playlist being viewed, etc.
- Current Playback: Data about the song currently being played or paused, its position, and related real-time information.

6.3 External Interface Data

- Spotify API Data: All the raw data fetched from Spotify, including but not limited to user playlists, daily listens, saved tracks, etc. This data is fetched real-time or in batches, processed, and then stored in the app's database or displayed to the user.

6.4 Transformation of Data

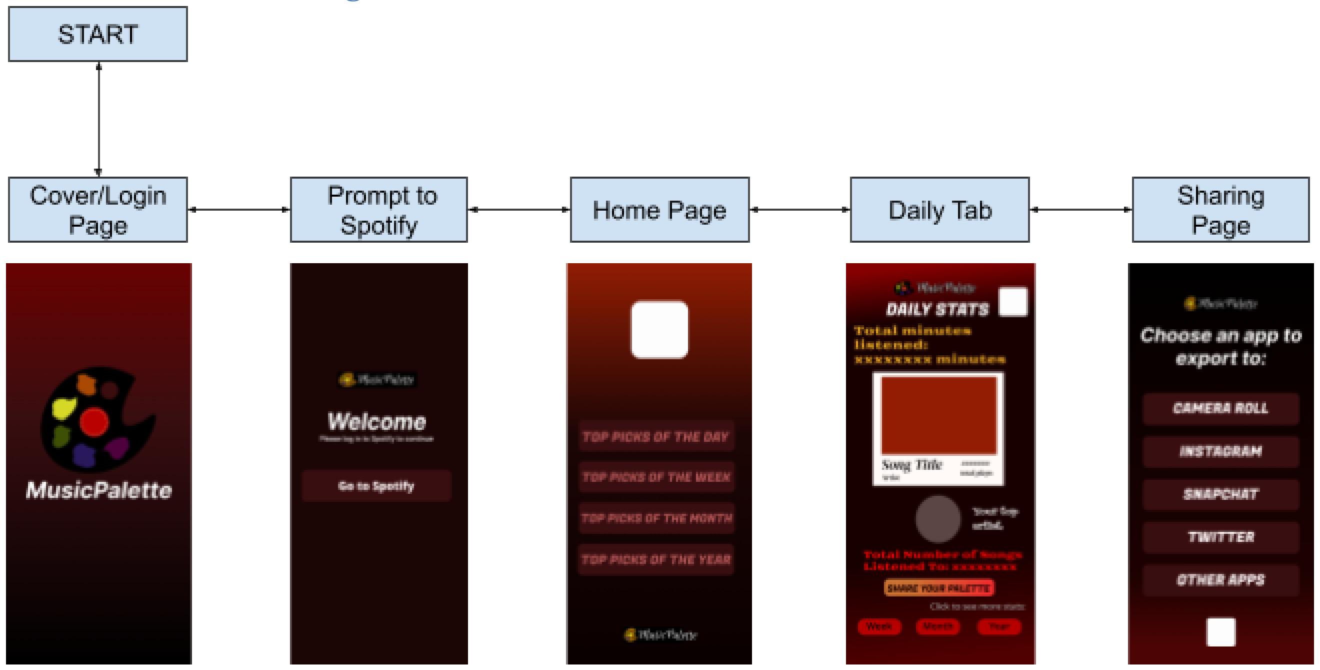
- Genre Classification: Songs fetched from Spotify are classified into predefined genres based on Spotify's data, aiding in user preference analysis.
- Listening Trend Analysis: Raw listening data is processed to derive trends such as the frequency of listening, preferred times of the day, etc.
- Summary Creation: After data analysis, the raw and processed data are transformed into visually appealing summaries, charts, or graphs for user consumption.

Section 7 - User Interface Design

7.1 User Interface Design Overview

The screen layouts will be compatible with all IOS devices. The app will also be easy to navigate for the user. The user will be able to see several options for different stats from their homepage, specifically Top Daily, Weekly, Monthly, and Yearly stats. The other stats can be accessed from the daily page, and the sharing page can be accessed from any of the stat pages. The app’s color scheme will be red, with different variations to highlight select features.

7.2 User Interface Navigation Flow



7.3 Use Cases / User Function Description

The user will open the app and log into the app using their Spotify account. This will then connect to the Spotify API and fetch their data so they can log into the MusicPalette app. After that, the homepage will display options to view different datasets from daily, weekly, monthly, and yearly collections. Each button can navigate to different tabs that will display each group of music listening statistics that are categorized differently.

Section 8 - Other Interfaces

8.1 Interface X

This project is intended to be supported on IOS devices as it is a mobile app. For the network, the user would need to connect to wifi or use cellular data to access it. For the software interface, we will be using Swift for IOS. For the backend, we will probably be hosting on AWS or Heroku. For the libraries, we will use the Spotify API. There will be notification updates through the application in order to let users know when a new music palette is generated. There will probably be an email as well to let the user know if the generated music listening stats are ready to be displayed or shared.

Section 9 - Extra Design Features / Outstanding Issues

The app will adhere to GDPR and other relevant privacy regulations. Secure HTTPS protocols will be used for data transmission, and user data will be encrypted. Currently, because we are using Spotify's API

Integration we will only need a bit of storage. For the future if we wanted to expand and add more features to accommodate then we would need more storage. The minimum hardware requirements for the software include any compatible iPhone version that works as well as software update version. End-users should not have much difficulty navigating through the app. We will aim to minimize the amount of navigation to certain screens. Guides will also be included as well as help bubbles. Other non-functional requirements include the loading time for everything to be kept at a maximum of 2 seconds and the default language to be English.

At the moment we have 1 outstanding issue where the app can get overloaded with requests and lock up. Then after the lockup ends it attempts all requests that happened during the lockup and can crash the app.

Section 10 – References

Figma UI Design Tutorial: Get Started in Just 24 Minutes
https://www.youtube.com/watch?v=FTFaQWZBqQ8&ab_channel=AJ%26Smart

Let's build Spotify with React Native! (Spotify API, Expo App Auth)
https://www.youtube.com/watch?v=mVd8XQ9Pl-0 https://www.youtube.com/watch?v=THnLOIl4wsU

Generate and Download QR code using React | Node JS
https://developer.spotify.com/documentation/web-api/tutorials/getting-started

Section 11 – Glossary

Terms

User Interface (UI): The space where interactions between humans and machines occur, aiming to be effective, efficient, and enjoyable.
Backend: The server-side operations of an application, which handles logic, database interactions, and server configuration.
Frontend: The client-side part of an application that interacts with the user.
API (Application Programming Interface): A set of rules and protocols for building and interacting with software applications.
HTTPS (HyperText Transfer Protocol Secure): An extension of HTTP that is used for secure communication over a computer network.
GDPR (General Data Protection Regulation): A regulation in EU law on data protection and privacy.
Uptime: The time during which a machine, especially a computer, is in operation.
CI/CD (Continuous Integration/Continuous Deployment): A DevOps practice where code changes are automatically built, tested, and deployed.
SDK (Software Development Kit): A collection of software tools and libraries used for developing applications for specific devices or operating systems.
Latency: A measure of time delay experienced in a system, the precise definition of which depends on the system and the time being measured.

Acronyms

UI:	User Interface
API:	Application Programming Interface
HTTPS:	HyperText Transfer Protocol Secure
GDPR:	General Data Protection Regulation
CI/CD:	Continuous Integration/Continuous Deployment
SDK:	Software Development Kit