Software Requirements Specification

for

<MUZIX JAVA BOILERPLATE>

Version 1.0

Prepared by <Arsh Aftab>

<Cognizant>

<25-04-2019>

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

*Build a system to list music tracks and bookmark favourite track and it to playlist.* 1

1.2 Intended Audience and Reading Suggestions 1

1.3 Product Scope 1

1.4 References 2

*Reference APIs:* 2

*1. https://www.last.fm/api* 2

*2.* *https://ws.audioscrobbler.com/2.0/method=chart* 2

*3.* *https://ws.audioscrobbler.com/2.0/method=track.search* 2

2. Overall Description 2

3.1 Product Perspective 2

*The application needs to fetch Music tracks by registering with the following link* 2

*and get API key required to call the APIs.* 2

3.2 Product Functions 2

*A frontend web app where the user can register/login to the application, list Music tracks, show track details for selected track and add to playlist your favourite track.* 2

*User can add track into his play list and should be able to view the playlist for user* 2

3.3 Operating Environment 2

3.4 Assumptions and Dependencies 3

4. External Interface Requirements 3

4.1 User Interfaces 3

*1* *List Music tracks and summary of tracks for each track fetched* 3

*2* *View details of a track* 3

*3* *Add a track to playlist* 3

*4* *should be able to see his playlist* 3

*5* *UI should be responsive which can run smoothly on various devices* 3

4.2 Flow of Modules 3

4.3 Building the accountmanager 3

4.4 Building the muzixmanager 4

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

## *Build a system to list music tracks, show details of tracks fetched and user can add a particular track to his playlist.*

## Intended Audience and Reading Suggestions

Developer and StackRoute Team

## Product Scope

An application which can help music lovers to see all their favourite songs and their artists. They can even save a favourite song in their playlist or remove them.

## References

# *Reference APIs:*

# *1. https://www.last.fm/api*

# [*https://ws.audioscrobbler.com/2.0/method=chart*](https://ws.audioscrobbler.com/2.0/method=chart)

# 3 [*https://ws.audioscrobbler.com/2.0/method=track.search*](https://ws.audioscrobbler.com/2.0/method=track.search)

# 2. Overall Description

## Product Perspective

## *The application needs to fetch music tracks data by registering with the following link*

## *and get API key required to call the APIs.*

## Product Functions

## *A frontend web app where the user can register/login to the application, list Music tracks, show music tracks details for selected track and add to playlist ther favourite song.*

## *User can add match into playlist and should be able to view the playlist for user.*

## Operating Environment

Can run in any environment with

Spring Boot

Angular

Docker installed

## Assumptions and Dependencies

The data is dependent on [www.last.fm/api](http://www.last.fm/api)

# External Interface Requirements

## User Interfaces

## *List Music tracks and summary of music for each track.*

## *View details of a tracks.*

## *Add a music track to his playlist.*

## *should be able to see playlisted songs.*

## *Should be able to search for a song .*

## Flow of Modules

* **Building Frontend**
* Building Responsive Views:
* Build a View to show all music

1. Music - Populating from external api

2. Build a view to show created playlist

3. A view to authenticate users

Using Services to populate these data in views

Stitching these views using Routes and Guards

* E2E test cases and unit test cases
* Writing CI configuration file
* Dockerizing the frontend
* **Building the Account Manager**
* Creating a server in Spring Boot to
* facilitate registration and login using JWT token and MySQL
* Writing Swagger Documentation
* Unit Testing
* Write CI Configuration
* Dockerize the application
* **Building the Muzix Manager**
* Creating a server in Spring Boot to
* facilitate CRUD operation over music, playlist and bookmarked resources stored in MySQL
* Writing Swagger Documentation
* Unit Testing
* Write CI Configuration
* Dockerize the application
* **Write docker-compose file to build both frontend and backend application**
* **Demonstrate the entire application**