Muzix App Case Study

1	**Name of the Project**	**Muzix App**
2	**Objective/ Vision**	Build an Application to search for music tracks and add them to a playlist/favourite list. Create a Dashboard view with following sections - Favorite Playlist, Recommendations for a playlist. Data can be fetched from https://www.last.fm/api. - This Dashboard is the default view to be shown. - Favorites - Recommended tracks
3	**Users of the System**	All Internet users
4	**Functional Requirements**	**1)** Home Page should consist of Register page through which a user can register himself. Upon registration, the user should be able to login into his account. **2)** User home page should also have options to edit the profile and change his/her password. **3)** User should be able to search tracks based on title/artist **4)** Favorite View - View all Favorite tracks cards under Favorite section **5)** Recommendation View -View all tracks recommendations from 3rd party tracks service provider under recommendations section
5	**Non-functional requirements**	a) App should be accessible from any location with access to the Internet. b) App should be responsive to display consistently across multiple device screens. c) App should have an intuitive UI that can be operated by novice-expert Internet users
6	**Tools and Technologies to be used**	**1.** VCS : Gitlab **2.** Middleware : Spring Boot **3.** Frond end : Angular/React **4.** Data Store : MongoDB / MySQL **5.** Testing : JUnit, Mocha, Chai, Jest, Protractor **6.** Container : Docker **7.** Bug Fix : Sonarlint **8.** CI : Gitlab

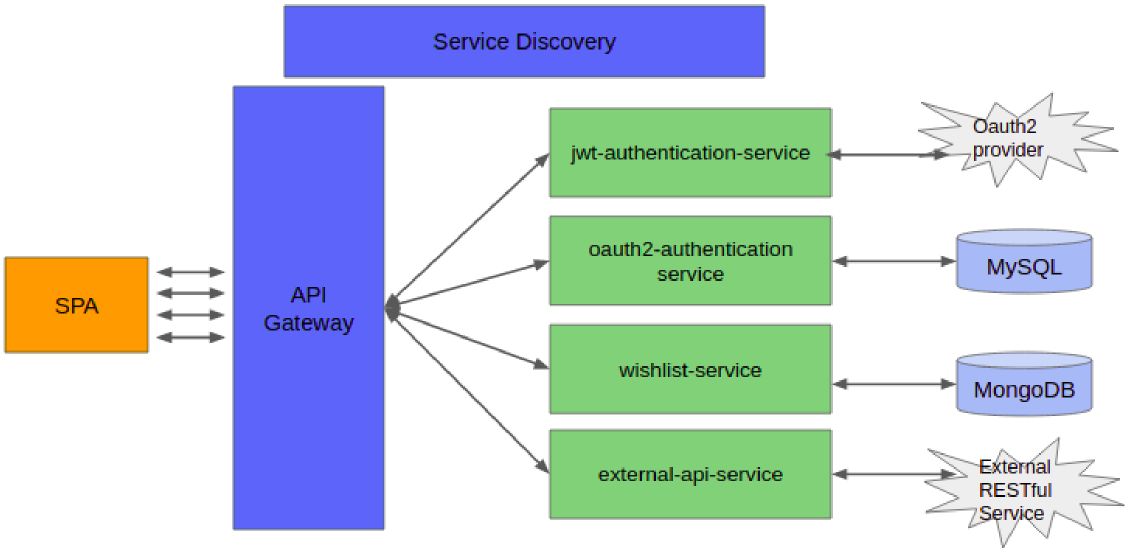
User Stories

1	**As a user I should be able to register with the application so that I can login and use the functionalities of the application.**
2	**As a user, I should be able to login with my user name and password in order to access the functionalities of the application.**
3	**As a user, I should be able to login with my Gmail account in order to access the functionalities of the application.**
4	**As a user I should be able to search resources to view their details**
5	**As a user, I should be able to save resources to a wishlist/favourite so that I can access them later**
6	**As a user, I should be able to access items saved to my wishlist/favourite**

Notes:

- The application should be based on microservices architecture
- API Gateway pattern should be implemented using Spring Cloud Gateway
- Services should register themselves with Eureka Service Discovery server.
- All layers of microservices should be covered with automated unit and integration tests
- All microservice endpoints should have API documentation

High Level Architecture Diagram


Service Discovery
SPA
API Gateway
jwt-authentication-service
oauth2-authentication service
wishlist-service
external-api-service
Oauth2 provider
MySQL
MongoDB
External RESTful Service