Trifonov,Dobri D.V.

Architecture Document

# Project Description

## Goal of the Project

This project is dedicated to making a Web Application with the goal of synchronized music storage. With a simple design the whole application is easy to navigate through requiring zero to no user configuration. The app saves your music online in a cloud storage so you will never need to worry about transferring your music again. All of your songs will be stored into your account and synchronized between all of your devices so you can enjoy your music on your computer or on the go. You can find new music within the app. All music available on Youtube and Spotify is available within the app through the use of API. The app will be usable from any devices’ browser with a minimalist design and all responsive frontend.

## Scope

|  |  |
| --- | --- |
| **Inside Scope** | **Outside Scope** |
| Users can register |  |
| Users can add music to their account | The app does not provide licensed music |

# Architecture diagrams

## C1 architecture diagram

Diagram

Description automatically generated

## C2 Architecture diagram

Diagram

Description automatically generated

## C3 Architecture diagram

Diagram, schematic

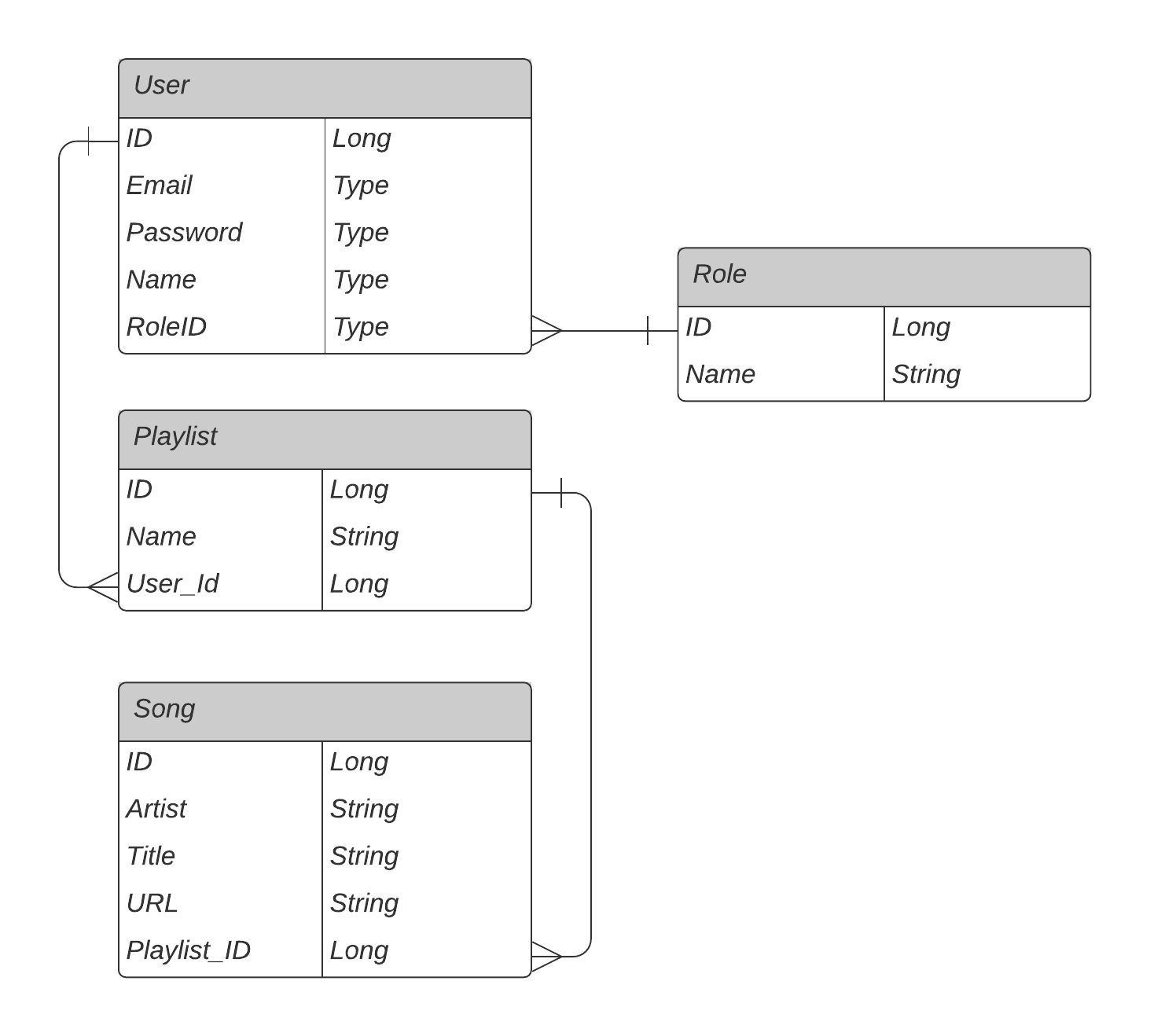
Description automatically generated

## UML diagram

Diagram

Description automatically generated

## ERD diagram



# Solid Principles Guaranteed

## Single Responsibility

Classes are responsible for only one thing. This allows to have fewer unit tests and have less dependencies throughout my project. All classes are single-purpose, small and easy to understand.

## Open/Closed

I will ensure this by adding more to the base application instead of changing the base working code.

## Liskov Substitution

I will ensure this by making every class that implements an interface be able to substitute any reference throughout the code that implements that same interface.

## Interface Segregation

I will ensure this by splitting any interface that becomes too large within the project

## Dependency Inversion

I will ensure this by having implementations that would be easily changeable without altering any complex code.

# Software used

## Backend

For the backend I will use Spring Boot because it lets me create a standalone application that runs on it's own without relying on an external web server such as Tomcat into my app.

## Frontend

For the frontend I will use React because it allows me to create reusable UI components. React is also fast, scalable, and simple.

## Database Layer

The database layer will be a point of research and testing within my project duo to the uncertainty of reading, writing and storing big files, such as audio files, into a database.