Listen Up

Design Document

Mohammad Nazibul Kabir Khan

4263308

How is SOLID guaranteed

Single responsibility

Every class has their own special responsibility so that there in not more than one reason for one class to change.

Open/closed principle

Each management classes are designed such a way it will be closed for modification, but it is open for extension for later phase.

Liskov substitution

Some classes are using reference from their base classes. For example, user of the application is divided by two types: Admin and Customer. Same for song: Single and Album Song.

Interface segregation

At the moment this principle is not used in the application.

Dependency inversion

Business classes is connected with their database classes through abstraction. It means business classes can be depended on different database. The presentation layer is connected to the business layer through another abstraction. Thus, it allows presentation layer to use multiple business layer.

Important Design Decision

Why user spring boot

I chose spring boot because it is easy to create stand-alone spring-based application with it. I just have to run it. Dependencies that a project need can also be added to it.

Front end library for building user interfaces

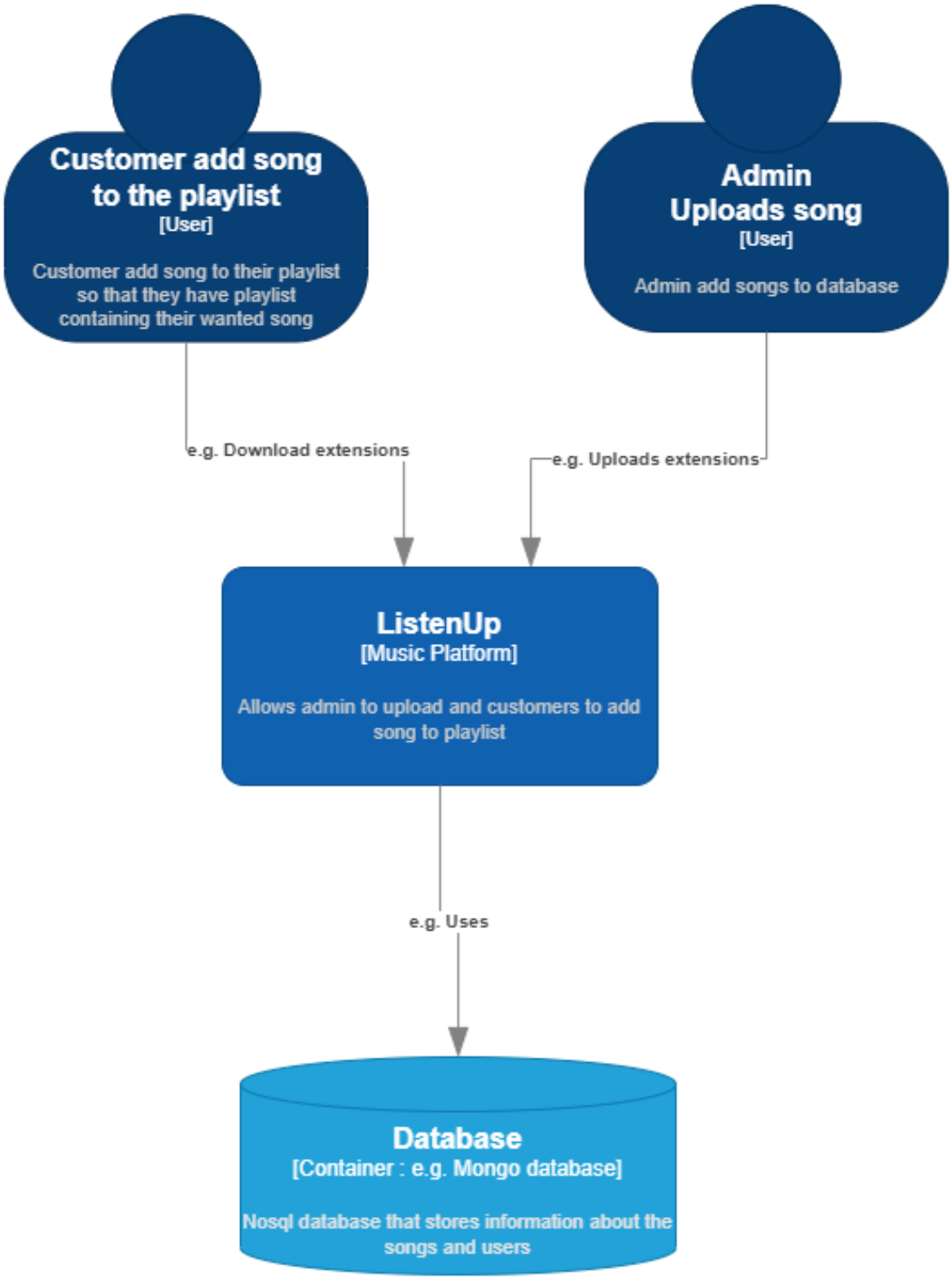
I am using react because I would like to learn more about it. Bigger companies are using react therefore there is no reason to chose otherwise. React also makes coding simpler which will reduce the amount of code I have to type.

Database to use

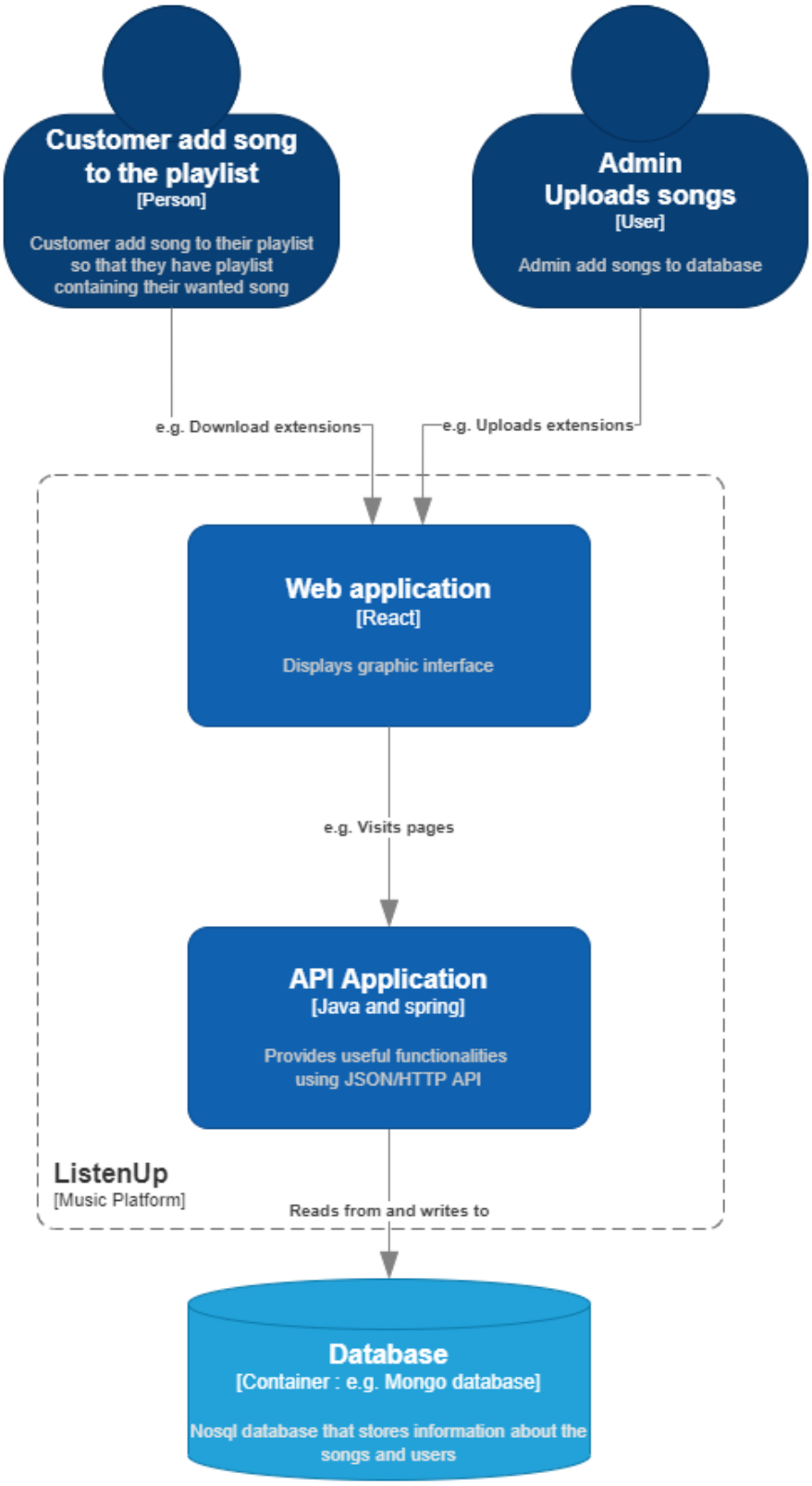
I have not decided which database I am going to use. I think MongoDB are used by bigger companies. But I am more familiar with MySQL. I am going to decide on later phase.

C4 Architecture

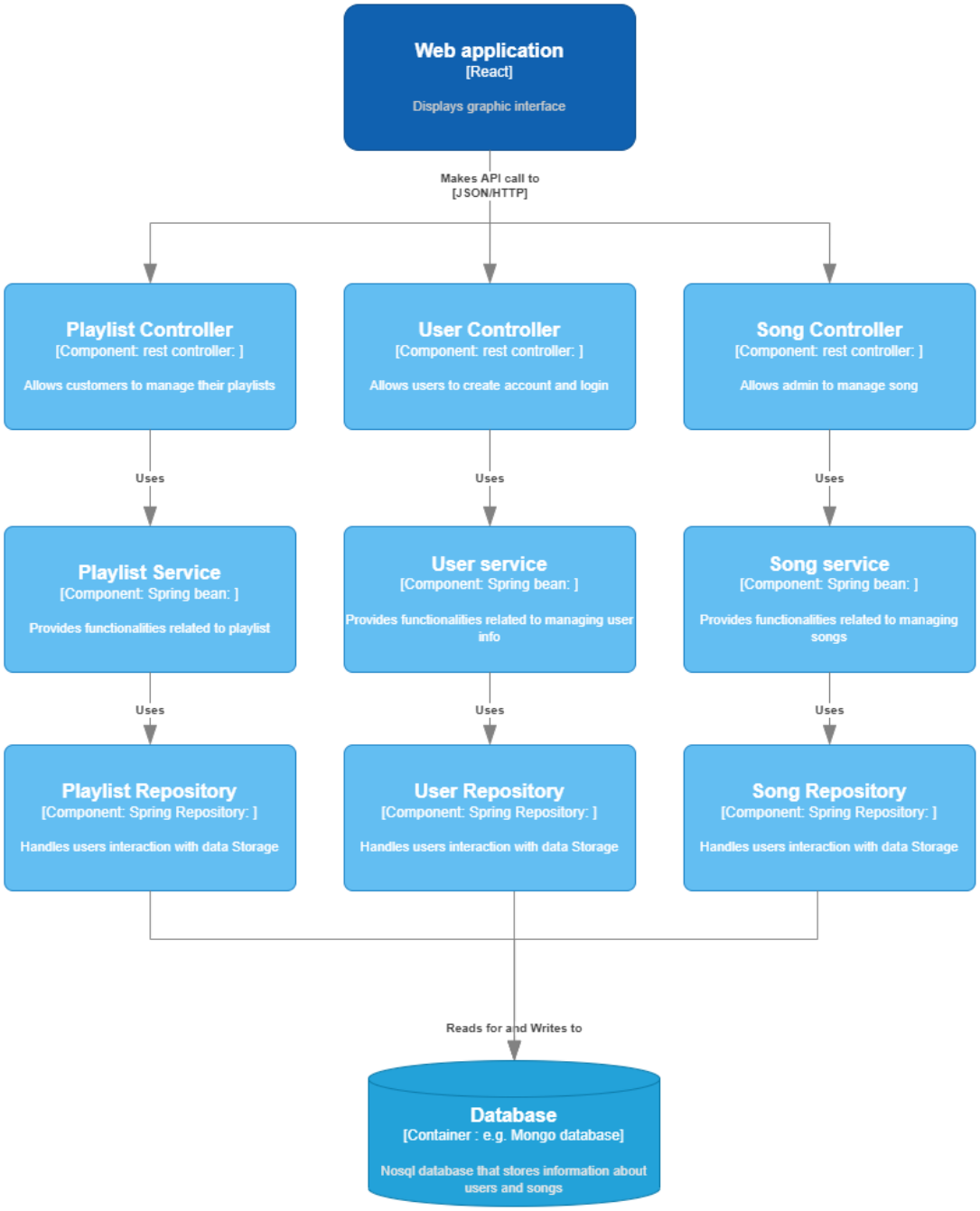
C1



C2

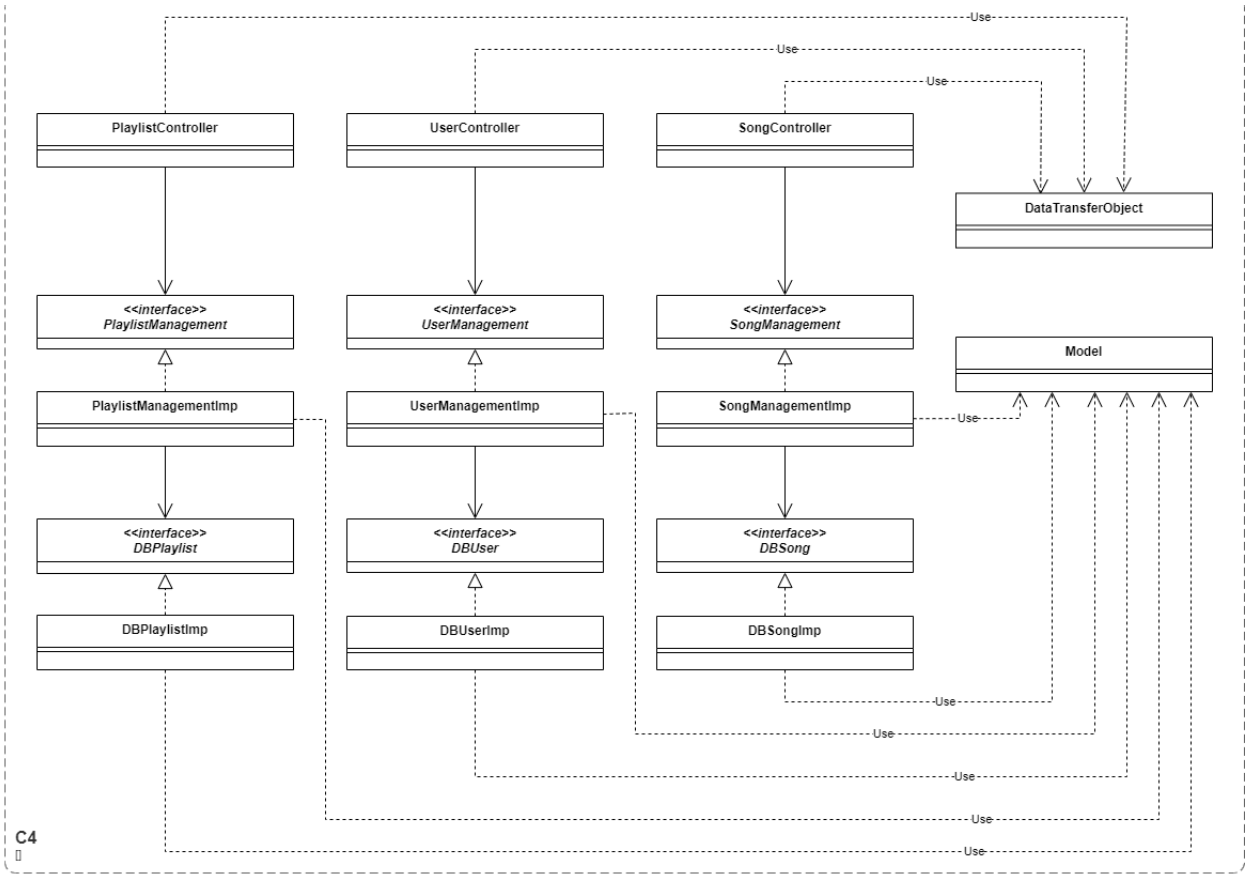
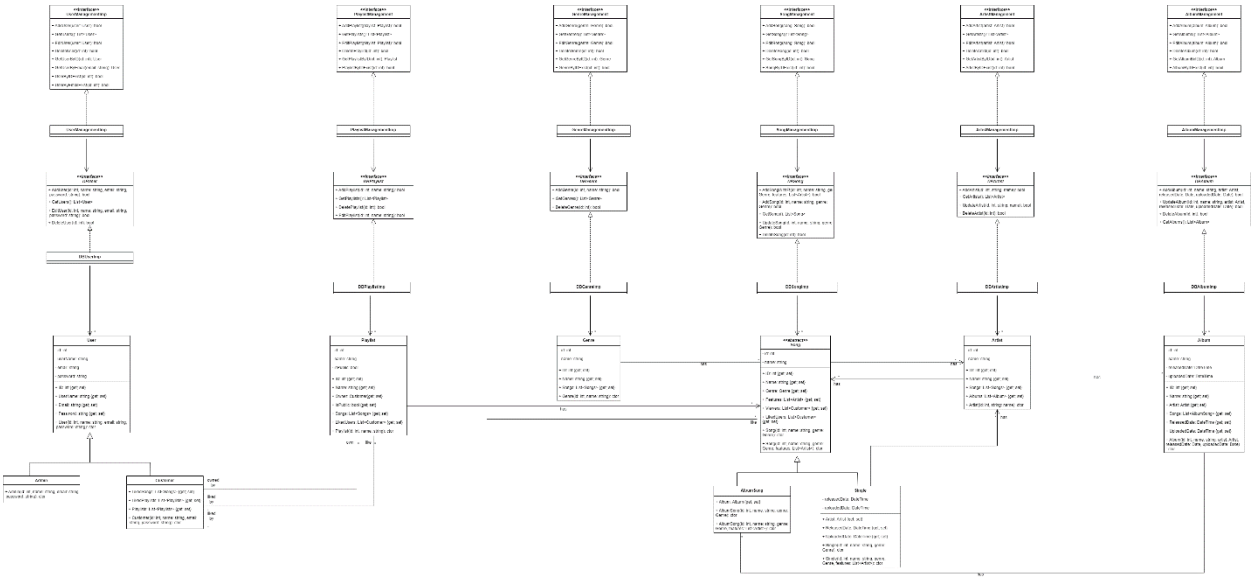

Customer add song to the playlist
[Person]
Customer add song to their playlist so that they have playlist containing their wanted song
Admin Uploads songs
[User]
Admin add songs to database
e.g. Download extensions
e.g. Uploads extensions
Web application
[React]
Displays graphic interface
e.g. Visits pages
API Application
[Java and spring]
Provides useful functionalities using JSON/HTTP API
ListenUp
[Music Platform]
Reads from and writes to
Database
[Container : e.g. Mongo database]
Nosql database that stores information about the songs and users

C3


Web application
[React]
Displays graphic interface
Makes API call to
[JSON/HTTP]
Playlist Controller
[Component: rest controller:]
Allows customers to manage their playlists
User Controller
[Component: rest controller:]
Allows users to create account and login
Song Controller
[Component: rest controller:]
Allows admin to manage song
Uses
Uses
Uses
Playlist Service
[Component: Spring bean:]
Provides functionalities related to playlist
User service
[Component: Spring bean:]
Provides functionalities related to managing user info
Song service
[Component: Spring bean:]
Provides functionalities related to managing songs
Uses
Uses
Uses
Playlist Repository
[Component: Spring Repository:]
Handles users interaction with data Storage
User Repository
[Component: Spring Repository:]
Handles users interaction with data Storage
Song Repository
[Component: Spring Repository:]
Handles users interaction with data Storage
Reads for and Writes to
Database
[Container : e.g. Mongo database]
Nosql database that stores information about users and songs

:3

C4


Use
Use
Use
PlaylistController
UserController
SongController
DataTransferObject
<<interface>>
PlaylistManagement
<<interface>>
UserManagement
<<interface>>
SongManagement
Model
PlaylistManagementImp
UserManagementImp
SongManagementImp
Use
<<interface>>
DBPlaylist
<<interface>>
DBUser
<<interface>>
DBSong
DBPlaylistImp
DBUserImp
DBSongImp
Use
Use
Use
Use
Use
C4

UML Class Diagram