Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Elective Project CJ1

Mykhailo Svyrydovych, Elvin Mammadov, Parisa Ostadzadeh

Raspberry PI compatible Music Voting Server

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Mykhailo Svyrydovych, Elvin Mammadov, Parisa Ostadzadeh

Raspberry PI compatible Music Voting Server

Elective Project CJ1 developed in course: Elective Project

in program Bachelor of Science Information Engineering
Department of Information and Electrical Engineering
Faculty of Engineering and Computer Science
HAW

Supervising professor:: Prof. Dr. Marc Hensel
Second examiner:

Released: 15. March 2021

Mykhailo Svyrydovych, Elvin Mammadov, Parisa Ostadzadeh

Title of the paper

Raspberry PI compatible Music Voting Server

Keywords

Java, Web, Server, JavaScript, Flutter, Mobile, Client-server, Spring

Abstract

This document is describing a group work on a software project. The project is about planning, designing a developing a web application that is compatible with Raspberry PI microcomputer and can be operated in a local network(over WiFi). Software includes backend server part based on Java Spring Boot framework, frontend for browsers based on React JavaScript library and a mobile app written in Dart with Flutter.

Contents

1 Introduction

The topic of this project is development of a Web Application that can be ran on a Raspberry PI[1] microcomputer. The application can be installed on an offline server (without Internet connection) and run entirely inside a local WiFi network. Raspberry PI can run a Linux operating system and particularly it has special designed distributive called Raspbian[2]. It is a free system from Debian family optimized for Raspberry PI hardware. The software can be also run on any computer(e.g. developers machine) which makes the development and testing process easier. The computer acts as a WiFi access point, so clients can connect to data served on its localhost. A user can connect with a mobile app or using web browser. In order to support both Android and iOS the mobile app is written using Flutter SDK[3]. Flutter is quite popular nowadays among mobile cross-platform development tools. It uses Dart language developed by Google. The use case of the project is to provide means for song voting that forms a live playlist. The application also plays music and allows songs uploads. It is useful during parties where participants would like to vote for their favorite songs. Generally, the project adopts Client-server architecture, where a server part hosts a database and a code responsible for voting process, user tracking and songs playing, and a client - is a mobile app or browser. Both communicating with each other using HTTP protocol[4] and REST API[5].

[1]https://www.raspberrypi.org
[2]https://www.raspbian.org/
[3]https://flutter.dev/
[4]https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[5]https://en.wikipedia.org/wiki/Representational_state_transfer

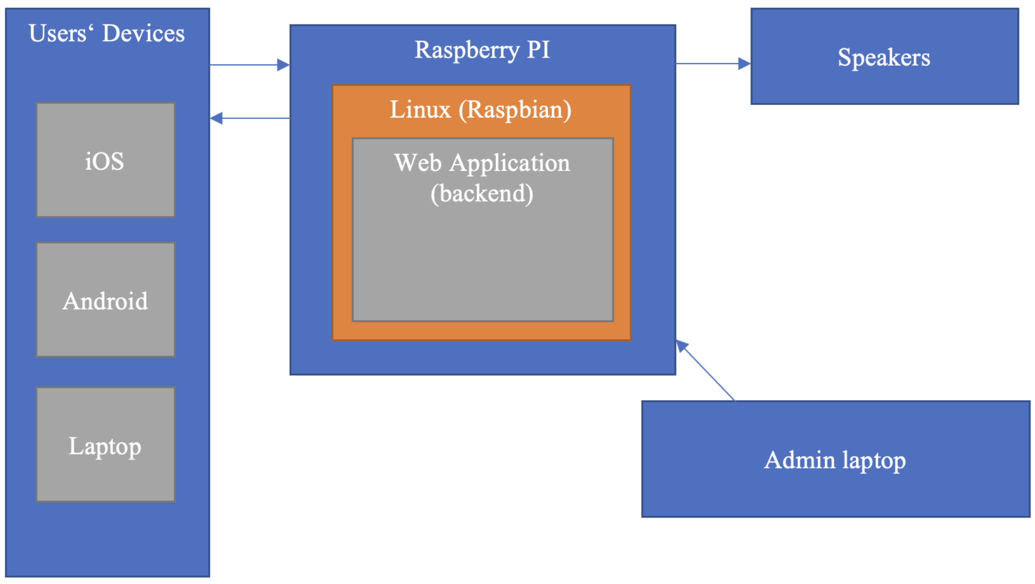


Figure 1.1: General project scheme

2 Motivation and objectives

The main objective was to practice working on a software project (that has more complex architecture than university labs) and coordinating as a team of several developers. We had to plan the project schedule using Gantt Chart. It is quite popular tool in Project Management when there is a need to show an activity over time. In our case it helped to make sure that our tasks and activities are inside the project time-frame. We used it to plan our work. The second tool that we used was YouTrack[1] - browser-based software. It is JIRA-like project management tool that is used for task planning and bug tracking. The second goal was to get knew knowledge and deepen existing in programming languages, frameworks and tools for Web- and mobile development. As these two areas of software development are quite popular nowadays. Finally, our goal was to have a working prototype that can be open for building additional functionality and third-party integration on top. The main features were planned with look into keeping up to a general time frame for a project. But we would like to continue working on it in future as well. The main features included: upload songs, played pre-installed and uploaded songs, show song list to users, accept votes, allow remote administration.

[1]https://www.jetbrains.com/youtrack/

3 Preparation

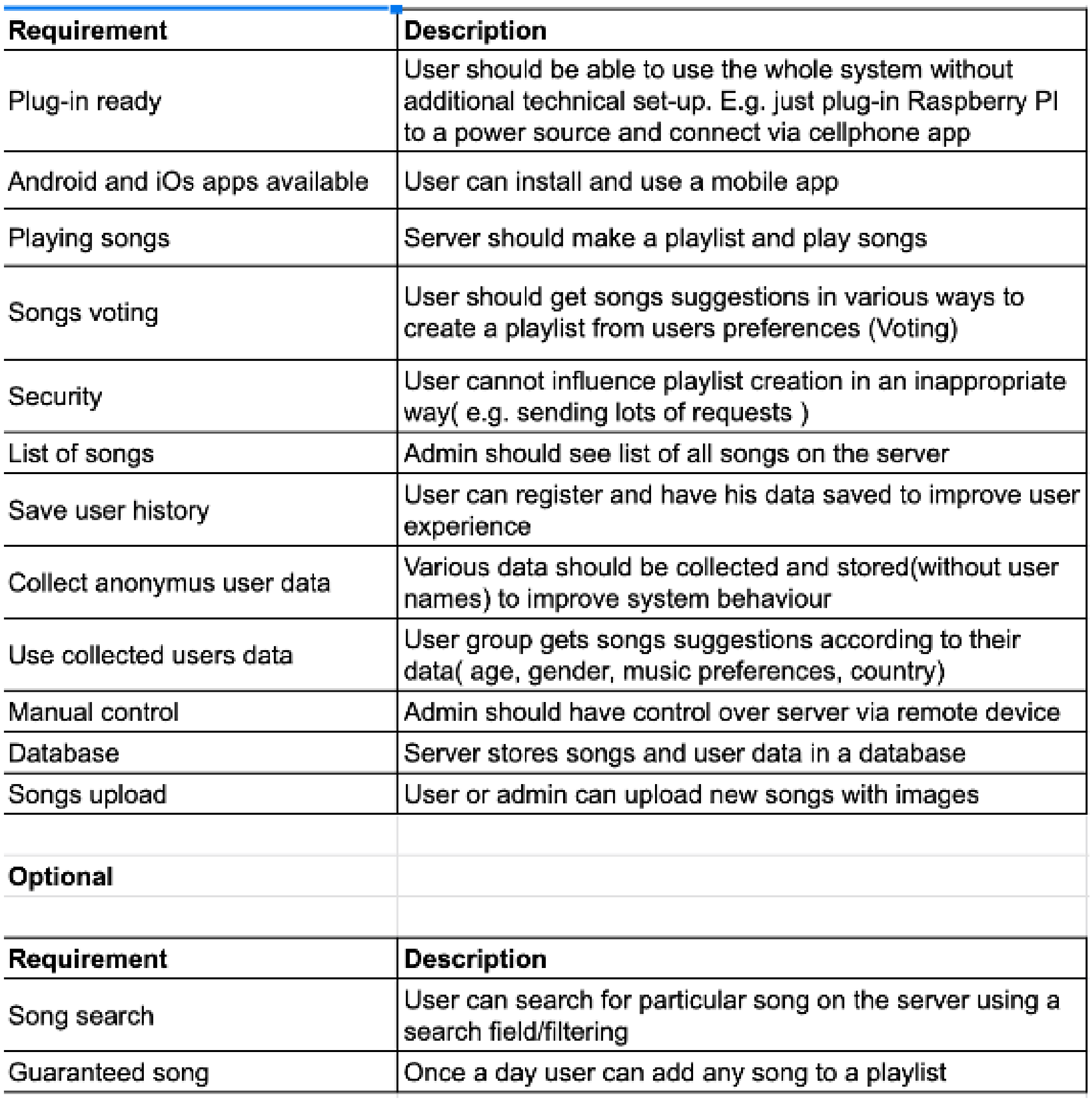
3.1 Gantt Chart

Gantt chart is a bar chart that is used to illustrate a schedule of a project. This kind of a chart includes tasks to be performed (on the vertical axis) and time intervals for the tasks (on the horizontal axis)[1]. For our project we used an Excel Gantt chart template as it is simple and easy to use. We needed the chart to plan the tasks for team members for the whole project according to given time frame. We did not include there every possible single task, but rather main general tasks to complete the project. We understood at that point that some additional tasks will appear as we progress and planned the time accordingly. We split tasks into five phases. Task were assigned to a Task Owner, but it did not mean that the person should do the whole task alone, but to control and be responsible for task completion. Our Gantt chart is included in project folder.

3.2 Requirements

As a part of the planning process, we had a team meeting where we brainstormed the possible projects features and derived requirements.

[1]https://en.wikipedia.org/wiki/Gantt_chart



Requirement	Description
Plug-in ready	User should be able to use the whole system without additional technical set-up. E.g. just plug-in Raspberry PI to a power source and connect via cellphone app
Android and iOs apps available	User can install and use a mobile app
Playing songs	Server should make a playlist and play songs
Songs voting	User should get songs suggestions in various ways to create a playlist from users preferences (Voting)
Security	User cannot influence playlist creation in an inappropriate way(e.g. sending lots of requests)
List of songs	Admin should see list of all songs on the server
Save user history	User can register and have his data saved to improve user experience
Collect anonymus user data	Various data should be collected and stored(without user names) to improve system behaviour
Use collected users data	User group gets songs suggestions according to their data(age, gender, music preferences, country)
Manual control	Admin should have control over server via remote device
Database	Server stores songs and user data in a database
Songs upload	User or admin can upload new songs with images

Optional

Requirement	Description
Song search	User can search for particular song on the server using a search field/filtering
Guaranteed song	Once a day user can add any song to a playlist

Figure 3.1: Requirements

It was a draft list of what the project prototype should be able to do by the end of the planned project schedule.

3.3 Risks

Risk management is a crucial part in any long running project. As we planned to do the project for at least 4 months, we decided to discuss all possible issues that could happen and influence the project. We created an xml table with risks descriptions, likelihood, impact, severity, owner, mitigation actions and contingent actions. The full table can be found in the project documents folder.

3.4 Technologies

It is really important to choose the right technologies at the beginning and not to switch it in the middle of the project. The main points in choosing the technologies for the project were:

1. Fitting to the project requirements (Web-App that can be run on a Raspberry PI micro-computer, mobile apps for the client)
2. Popularity on the job market (Our interest is to learn and work with modern tools that are required by potential employers)
3. Good documentation, tutorials etc. (To be able to learn it fast)

3.5 Frameworks and libraries

Any web application can be generally divided into two parts: back-end (running on the server) and front-end(running on the user device). Nowadays all web-applications are developed using frameworks and libraries. Frameworks usage increases the development speed and makes it possible to invest more time into implementing concrete features rather than inventing a bicycle and do some low-layer programming. For the back-end we decided to choose some framework based on one of the following languages: Java, C++, JavaScript (as team members had some knowledge in these three). The respective frameworks to look at were: Spring Boot, Treefrog, Express.js. Discussing them in a team according to the 3 technology requirements mentioned above, we decided to develop the back-end with Java Spring Boot. Research showed that there were some other successful web-application projects that were run on the Raspberry PI without problems, Java is a very popular language through the Hamburg companies and, finally, there are a really nice official documentation and non-official tutorials. The dominating language for the browser front-end is JavaScript. And we decided to use React framework as it has very useful official documentation with tutorial and it is the most popular front-end framework today. For the mobile clients the choice was to write native apps for Android and iOS in Java and Swift respectively or to choose some multi-platform solution. As we did not have enough manpower, we decided to search for the latter one. The choice here was between Flutter and React native. We decided to use Flutter.

3.5.1 Spring Boot

Spring started as an alternative to Java Enterprise Edition. It offered different Dependency Injection techniques and provided an Aspect-oriented programming paradigm. Initially it was hard in configuration and not novice-friendly. To solve this issue a subset of Spring called Spring Boot has been developed.[2] The main points that came with Spring Boot:

- Automatic configuration - satisfies most of the common Spring apps
- Starter dependencies - Spring Boot ensures that all required libraries are added
- The command line interface - optional feature that can simplify automated builds
- The Actuator - gives a view on what is going on inside the app[3]

Inversion of Control and Dependency Injection

Inversion of Control(IoC) can be described with a phrase: "Don't call me, I'll call you". According to this principle the Framework calls the Application code when it is needed and not vice-versa. Spring approach to IoC for configuration management is called Dependency Injection (DI). It is base on Java language constructs, classes expose their dependencies through methods and constructors, so the framework can call the appropriate values during run-time according to the configuration.[4]

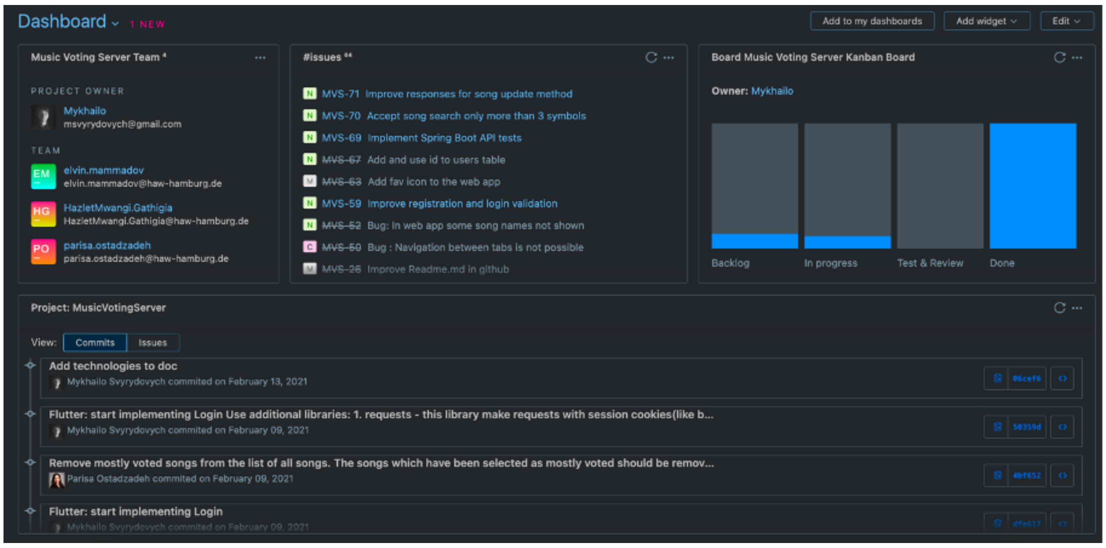
3.5.2 React.js

React is a JavaScript library for building web user interfaces. It provides a way to build user interfaces in a declarative and component-driven way. Components are special units of the application that are the primary building blocks. They have an internal state and properties and render the output accordingly.[5] Reacts uses declarative paradigm instead of imperative. A programmer had to declare how a component should look like and behave under different states. React handles the complexity of managing updates, updating UI and so on by its own. One of the core technologies used in React is a Virtual DOM(Document Object Model). It mimics the real browser DOM and serves as an intermediate layer.[6] This increases the performance as working with the real DOM is usually the slowest part and can be a bottle neck.

[2]Spring Boot in action - Craig Williams, 2016, p1,2
[3]Spring Boot in action - Craig Williams, 2016, p4
[4]Rod Johnson - Professional Java Development with the Spring Framework, 2005, p.26
[5]Thomas, M.T. - React in action, 2018, p.5
[6]Thomas, M.T. - React in action, 2018, p.15,16

3.6 Task tracking

JetBrains YouTrack is a web-based issue tracking and project management platform.[7] It designed for agile teams. It allows to create, assign and keep track of the development tasks. Task template can be easily customized with various options. Next important feature is the Kanban board that represents tasks and their states. The aim of the Kanban board is to make the general project workflow and task progress clear and available to all participants.[8] Finally, YouTrack allows integration with GitHub. This connection helps to track git commits right inside YouTrack(e.g. task key can be mentioned in a commit and it triggers an action at YouTrack).



Music Voting Server Team

PROJECT OWNER

Mykhailo
msvyrydovych@gmail.com

TEAM

elvin.mammadov
elvin.mammadov@haw-hamburg.de

HazletMwangi.Gathigia
HazletMwangi.Gathigia@haw-hamburg.de

parisa.ostadzadeh
parisa.ostadzadeh@haw-hamburg.de

#issues

MVS-71 Improve responses for song update method
MVS-70 Accept song search only more than 3 symbols
MVS-69 Implement Spring Boot API tests
~~MVS-67~~ Add and use id to users table
~~MVS-63~~ Add fav icon to the web app
MVS-59 Improve registration and login validation
~~MVS-52~~ Bug: In web app some song names not shown
~~MVS-50~~ Bug : Navigation between tabs is not possible
~~MVS-26~~ Improve Readme.md in github

Board Music Voting Server Kanban Board

Owner: Mykhailo

Project: MusicVotingServer

Add technologies to doc
Mykhailo Svyrydovych commited on February 13, 2021
06cef6

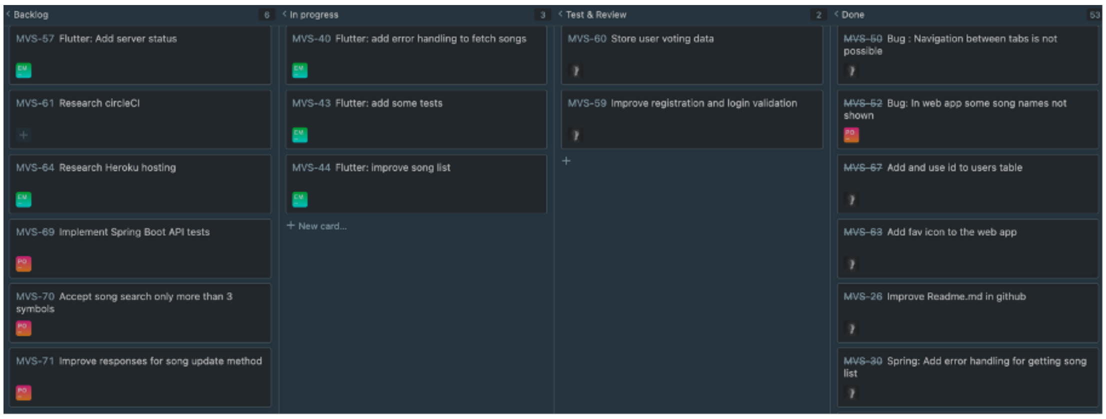
Flutter: start implementing Login Use additional libraries: 1. requests - this library make requests with session cookies(like b...
Mykhailo Svyrydovych commited on February 09, 2021
50359d

Remove mostly voted songs from the list of all songs. The songs which have been selected as mostly voted should be remov...
Parisa Ostadzadeh commited on February 09, 2021
4bf652

Flutter: start implementing Login
Mykhailo Svyrydovych commited on February 09, 2021
dfe617

Figure 3.2: YouTrack Dashboard

[7]https://www.jetbrains.com/help/youtrack/standalone/YouTrack-Documentation.html
[8]https://en.wikipedia.org/wiki/Kanban_(development)#Kanban_boards



Backlog 6	In progress 3	Test & Review 2	Done 53
MVS-57 Flutter: Add server status	MVS-40 Flutter: add error handling to fetch songs	MVS-60 Store user voting data	~~MVS-50~~ Bug : Navigation between tabs is not possible
MVS-61 Research circleCI	MVS-43 Flutter: add some tests	MVS-59 Improve registration and login validation	~~MVS-52~~ Bug: In web app some song names not shown
MVS-64 Research Heroku hosting	MVS-44 Flutter: improve song list		~~MVS-67~~ Add and use id to users table
MVS-69 Implement Spring Boot API tests	+ New card...		~~MVS-63~~ Add fav icon to the web app
MVS-70 Accept song search only more than 3 symbols			MVS-26 Improve Readme.md in github
MVS-71 Improve responses for song update method			~~MVS-30~~ Spring: Add error handling for getting song list

Figure 3.3: YouTrack Kanban Board

3.7 Version Control

Version control or source control is a practice that helps to track and manage changes of a source code. Especially it is useful in software teams, when several developers are working on the same project. It helps them to work faster and smarter. Version control systems keeps track of any change in some kind of a database and allows to revert a change and jump back to a specific point in the history. It protects the code base from human errors and makes bug fixing easier.[9]

3.7.1 Benefits of version control systems

- Getting long-term history of every file.
- Branching and merging. Allows to keep multiple streams of work separated for a while with a possibility to merge.
- Being able to trace each change mode to the program and track it with task tracking tools.[10]

3.7.2 Different VCS types

Version Control Systems can be divided into two groups: distributed and centralized. It describes how the remote architecture is built. A centralized VCS has a single point of failure –

[9] https://www.atlassian.com/git/tutorials/what-is-version-control
[10] https://www.atlassian.com/git/tutorials/what-is-version-control

central storage. When it is lost, the whole project is lost. A distributed VCS provides a full copy of the source code for each instance. Examples of centralized systems: SVN, CVS. Examples of distributed systems: Git, Mercurial.[11] We decided to use Git because it is the most popular distributed VCS. The two most popular Git hosting platforms are GitHub and GitLab. We chose GitHub as it allows to make a repository public in future (GitLab supports only private repositories).

3.7.3 Basic Git concepts

Repository

A Repository is a git database that holds all the information needed to retain the history of an object. The repository retains a complete copy of a project. In addition git has a set of configurations for the repository. I.e. repository user name or email address. These settings are not transferred during cloning and repository sharing. There are two main data structures within the repository - object store and index. All the git data is been stored inside a hidden sub-directory called '.git'. Object store is designed to be efficiently copied between repositories during cloning. On the other side, index stays private to each repository.[12]

Object types

Object store contains file data, log messages, authors, dates and other information. There are four types of objects[13]:

- Blobs. Each version of a file is a blob(Binary Large Object). It contains file data, but does not contain any meta-data.
- A Tree object contains all meta-data, filenames and blob identifiers for objects in one directory and recursively connects to other directories, so it builds a hierarchy.
- A Commit object contains metadata for every change of the repository. These includes author, committer, commit date, and log message. Also it points to a tree object. Most commits have one parent and initial commit has no parent.
- A Tag object assigns a name to an object.

[11]https://bitbucket.org/product/version-control-software
[12]Jon Loeliger - Version control with Git, 2009, p.29,30
[13]Jon Loeliger - Version control with Git, 2009, p.30

Index

Index is a dynamic and temporary binary file that contains the structure of all directories. The index is an intermediate staging area between the repository and a directory. User allowed to add and remove changes from the staging area.[14]

Content-addressable names

Each object in the Git object store has a unique name produced by applying SHA1 algorithm to the file content. Any changes to a file makes SHA1 name to be changed. Sometimes these names can be abbreviated to a smaller prefix.[15]

Git tracks content

Object store is based on the contents of the file and not on the name of the file. It differs git from other VCS. Git does not track file and directory names, but it tracks contents. In example, if two identical files are stored in different directories, git stores single object in the objects store. This blob object is indexed with SHA1 value and both files use this object for content. If one of these changes, git computes new SHA1 for this file and creates additional entry in object store, so from this point each file has its own object in store. Another important point - Git database store every version of every file[16] (the size of a git repository can become really big).

[14]https://www.javatpoint.com/git-index
[15]Jon Loeliger - Version control with Git, 2009, p.31
[16]Jon Loeliger - Version control with Git, 2009, p.31,32

4 Use case diagram

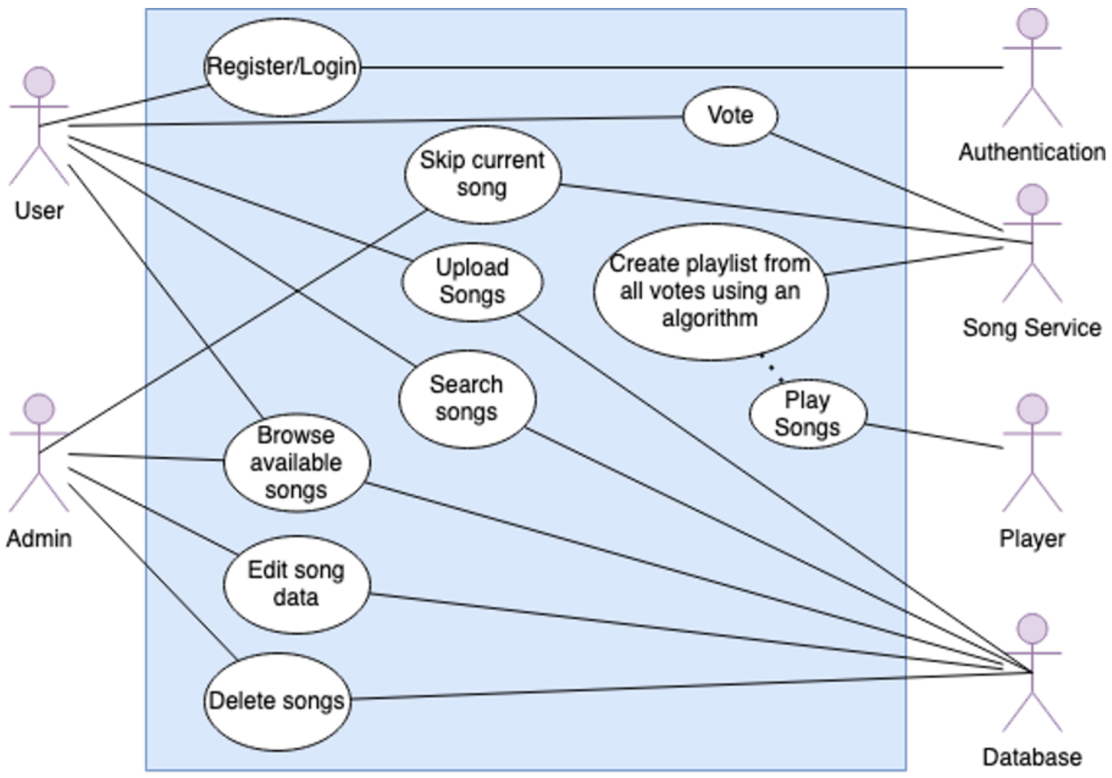


Figure 4.1: Use case diagram

5 Client-server architecture and REST API

A server is a remote computer that accepts requests from clients, processes them and sends back the response data over HTTP/HTTPS protocol. A client is a computer or a device that sends requests to the server and receives a response.[1]

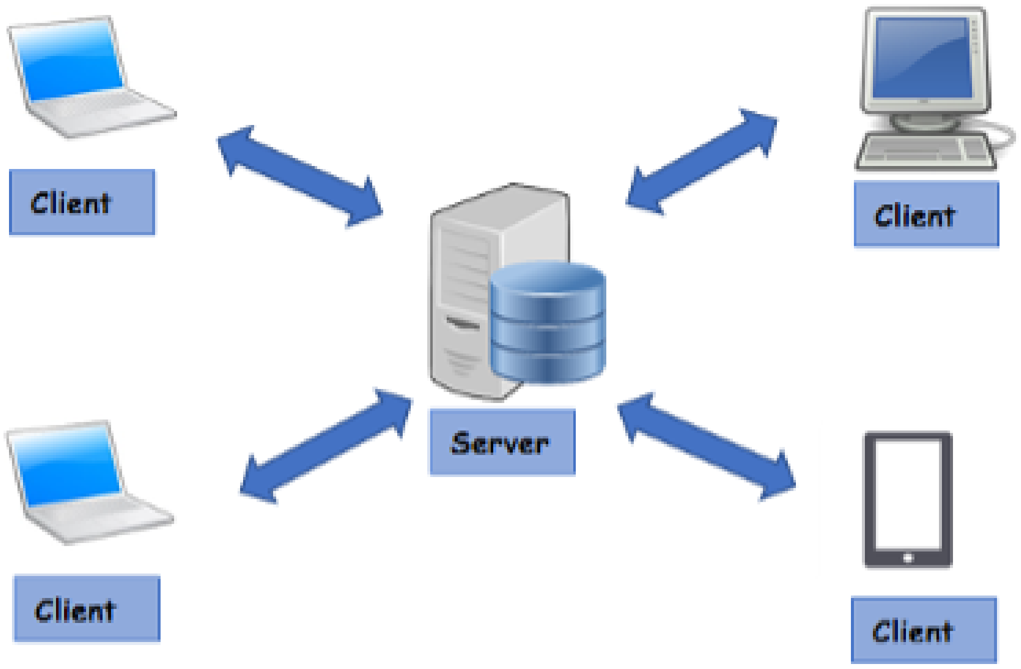


Figure 5.1: Client-server architecture from https://www.omnisci.com/technical-glossary/client-server

REST is a web architectural structure first described in a scientific paper called "Representational State Transfer" (REST). In web architecture clients use Application Programming Interfaces (APIs) to communicate with web server. These APIs are been used to provide a set

[1]https://www.educative.io/courses/learn-rest-soap-api-test-automation-java/JEK9Mv7RLA9

of functions to allow communication between computer programs and make data exchange. Nowadays REST API architecture pattern are widely used.[2] Generally, REST API architecture should conform following principles[3]

- Uniform interface
- Client–server
- Stateless
- Cacheable
- Layered system
- Code on demand (optional)

[2]Marc Masse - REST API design rulebook
[3]https://restfulapi.net/rest-architectural-constraints/

5.1 Project REST API overview

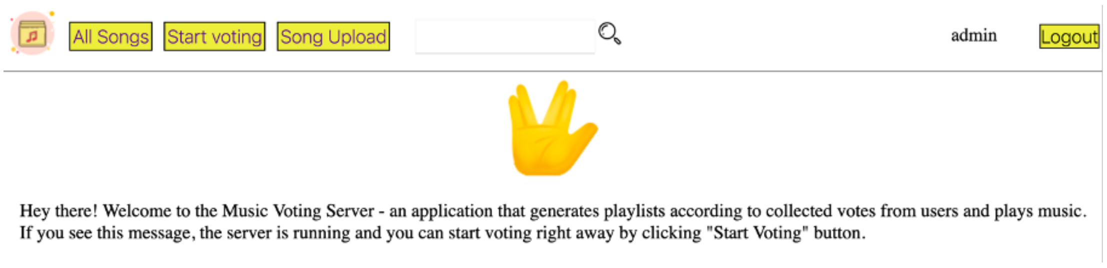
Http method	Endpoint	Description
GET	api/v1/songs	Returns a list of all songs
GET	api/v1/pairs	Returns a list of songs pairs (for voting)
POST	api/v1/vote	Accepts list of songs chosen by a user
POST	api/v1/songs/upload	Accepts mp3 song file
GET	api/v1/songs/current	Returns currently played song
GET	api/v1/songs/mostlyVoted	Returns a list of mostly voted songs
GET	api/v1/songs/current/stop	Stops playing current song
DELETE	api/v1/songs/{id}	Deletes a song
PUT	api/v1/songs/{id}	Changes song info
GET	api/v1/get-voters-number	Returns voters number setting
POST	api/v1/songs/set-voters-number	Changes voters number setting
GET	api/v1/songs/get-playlist-size	Returns playlist size setting
POST	api/v1/songs/set-playlist-size	Changes playlist size setting
GET	api/v1/user	Returns user info
POST	api/v1/register	Registers a new user

6 Browser App GUI and functionality overview

The browser app is a Single Page Application(SPA) based on React.js library. React introduces reactive programming. (TODO add some info about reactive programming). SPA is a browser program that interacts with user by dynamically reloading new data to single page. Besides it uses router library that provides pseudo pages, so each page can relate to an URL endpoint(i.e. “/allSongs” for all songs page and “/” for main page). But under the hood it still SPA, so pages are changed seamlessly and fast. Browser App GUI does not use any ready components (like navbars etc.) and all UI elements are designed and programmed from scratch.

6.1 Main page

Main page consists of a welcome message and a toolbar. Toolbar contains navigation, search and login buttons.

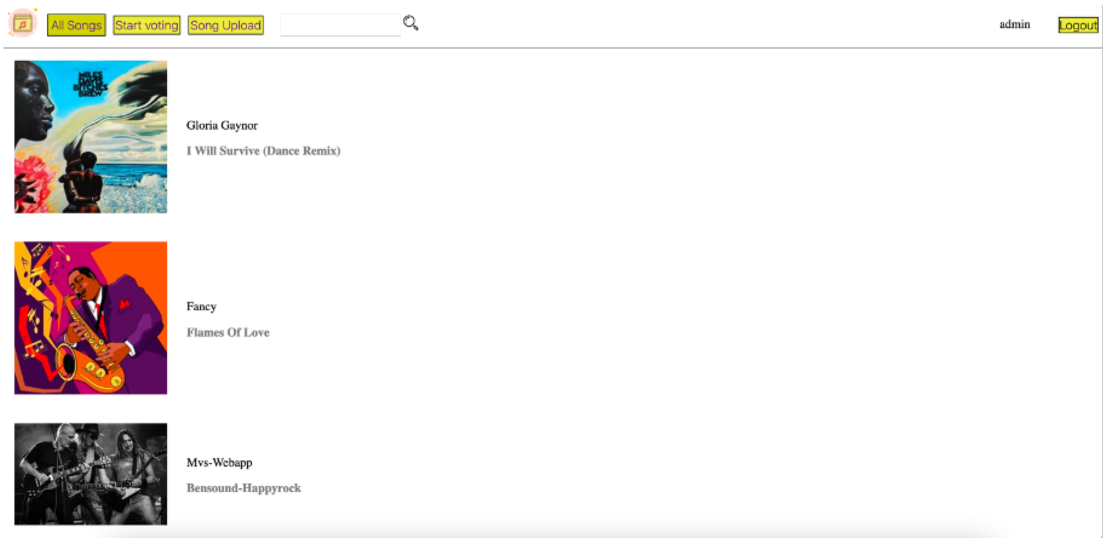


Hey there! Welcome to the Music Voting Server - an application that generates playlists according to collected votes from users and plays music.
If you see this message, the server is running and you can start voting right away by clicking "Start Voting" button.

Figure 6.1: Main page

6.2 All songs

This page retrieves all available song data from the database and displays them as a list. User can click on a song to get more details.



Gloria Gaynor
I Will Survive (Dance Remix)

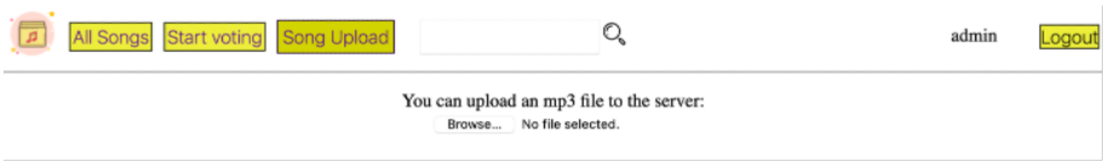
Fancy
Flames Of Love

Mvs-Webapp
Bensound-Happyrock

Figure 6.2: All songs

6.3 Song upload

Upload page allows user to transfer mp3 file to the server, so this song can be stored and played in future.

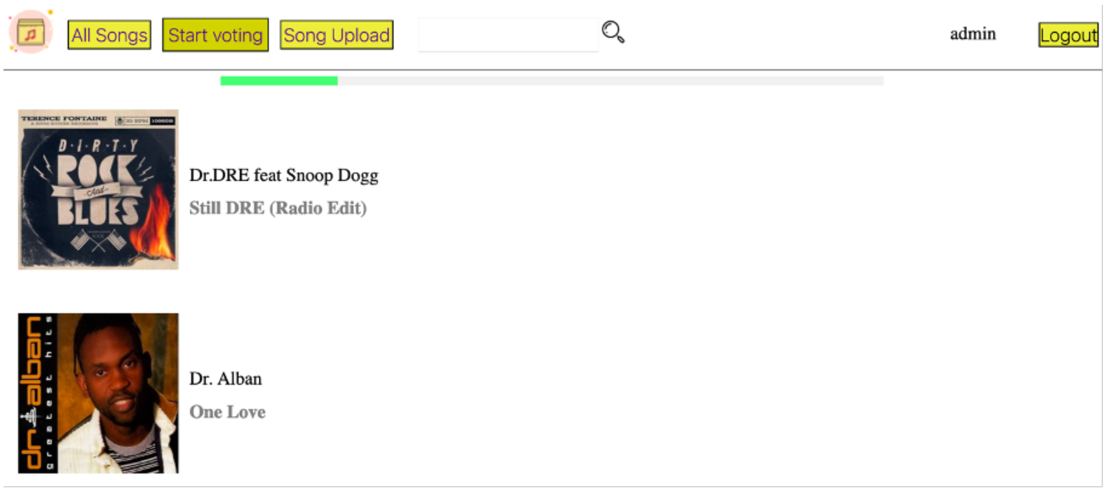


You can upload an mp3 file to the server:
Browse... No file selected.

Figure 6.3: Song upload

6.4 Voting

Voting is the main feature of the project. Users choose between one of two suggested songs until a playlist is formed. When all users finish voting, player plays the songs from the playlist. Voting page displays pairs of songs and a progress bar. User should click on a song to vote for it.



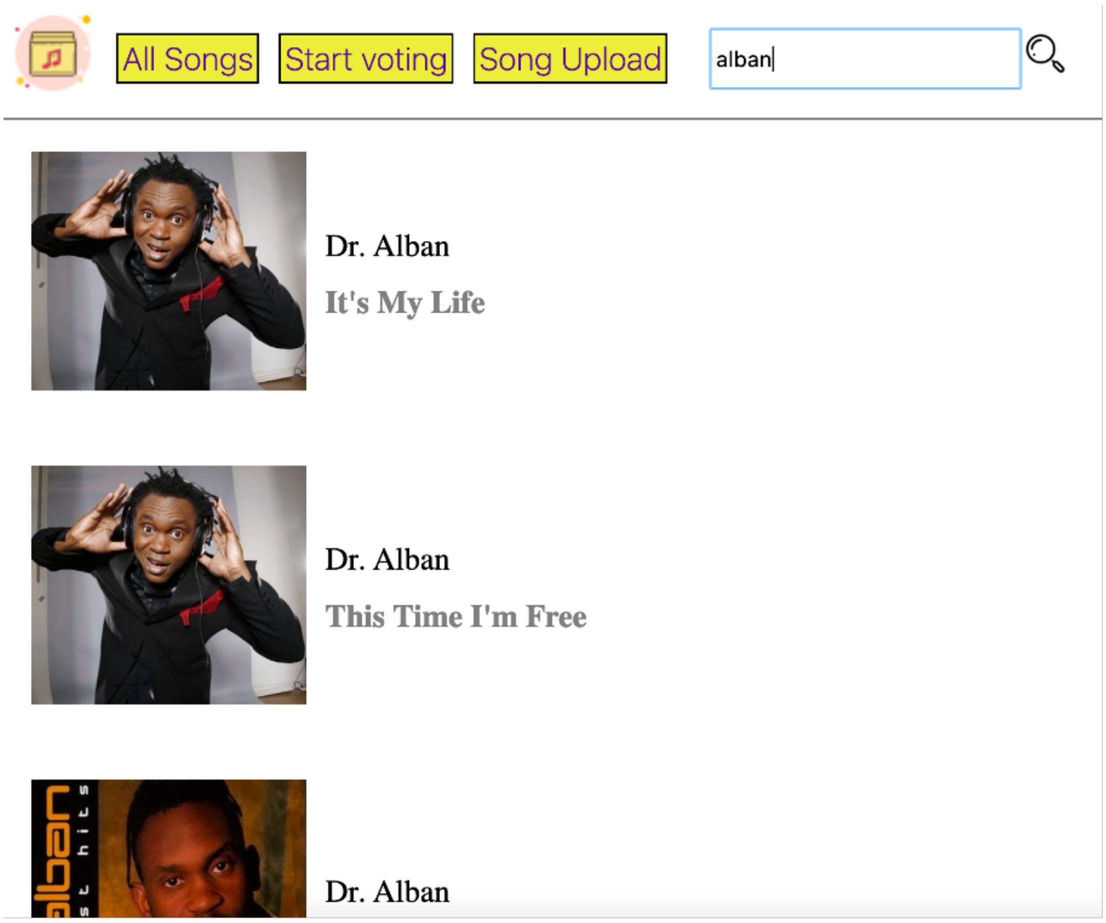
Dr.DRE feat Snoop Dogg
Still DRE (Radio Edit)

Dr. Alban
One Love

Figure 6.4: Voting

6.5 Search

Server supports search requests. It searches through the database and returns matching songs.



Dr. Alban
It's My Life

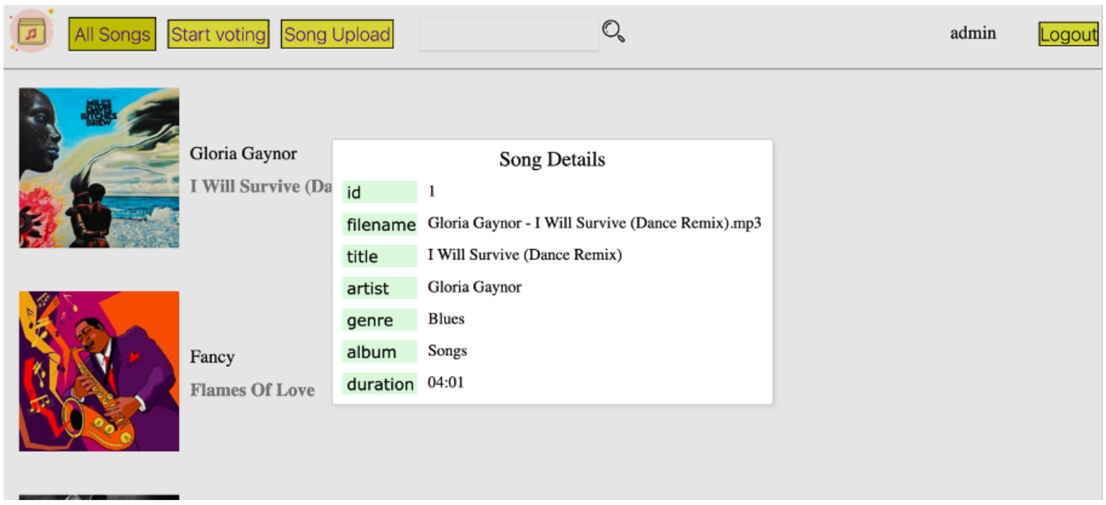
Dr. Alban
This Time I'm Free

Dr. Alban

Figure 6.5: Search

6.6 Song info

When user clicks on a song, a modal dialog with all song data is shown.



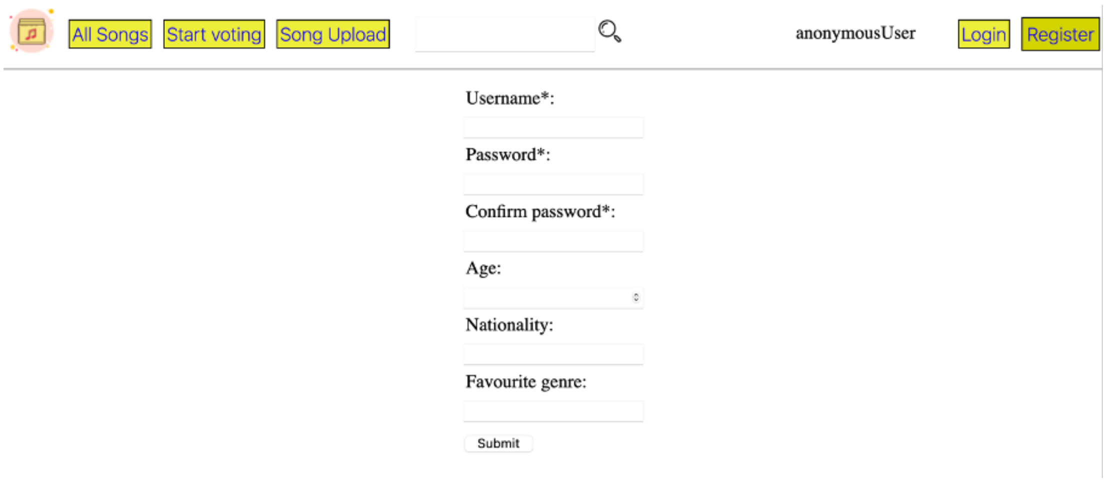
Song Details

id	1
filename	Gloria Gaynor - I Will Survive (Dance Remix).mp3
title	I Will Survive (Dance Remix)
artist	Gloria Gaynor
genre	Blues
album	Songs
duration	04:01

Figure 6.6: Song info

6.7 Registration

User can register. Then his choices will be stored in the database.



Username*:

Password*:

Confirm password*:

Age:

Nationality:

Favourite genre:

Figure 6.7: Registration

6.8 Log in

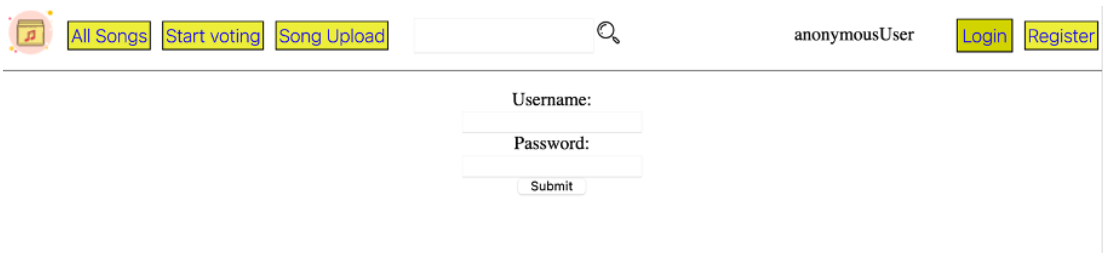


Figure 6.8: Log in

6.9 Admin NavBar

Admin has a separate web page which has a bit different navbar. It contains some admin specific functionality.

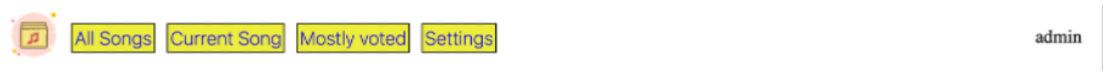
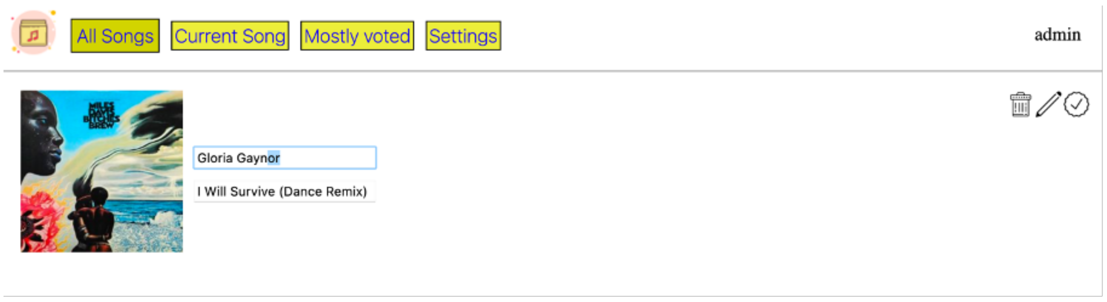


Figure 6.9: Admin navbar

6.10 Admin song editing

Admin can edit song name and artist name. It got updated in the database. Also admin can completely delete a song.



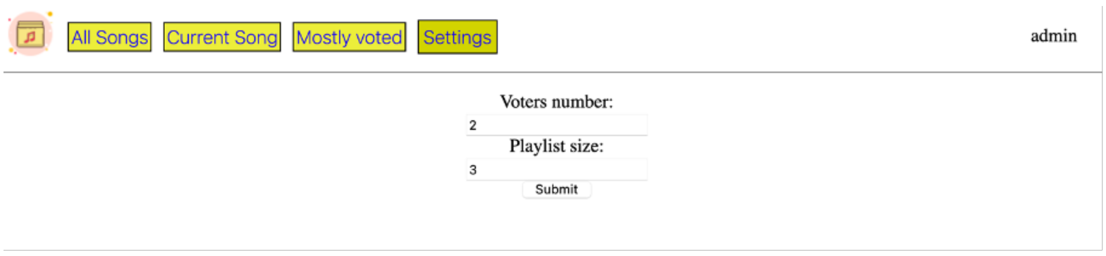
Gloria Gaynor

I Will Survive (Dance Remix)

Figure 6.10: Admin song editing

6.11 Admin settings

Admin can set voters number (after all people sent their votes, player starts to play music). And admin can set playlist size that will be generated from the votes.

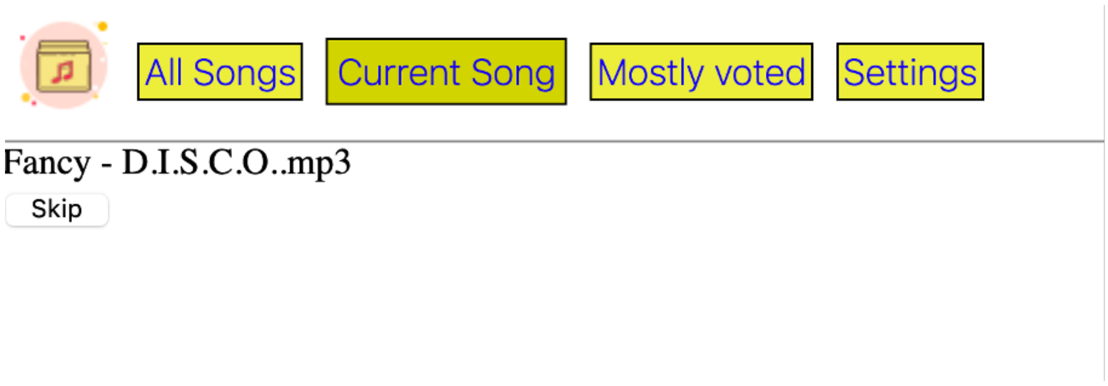


Voters number:
2
Playlist size:
3

Figure 6.11: Admin settings

6.12 Current song

Admin can see which song is being played currently and can skip to the next one.

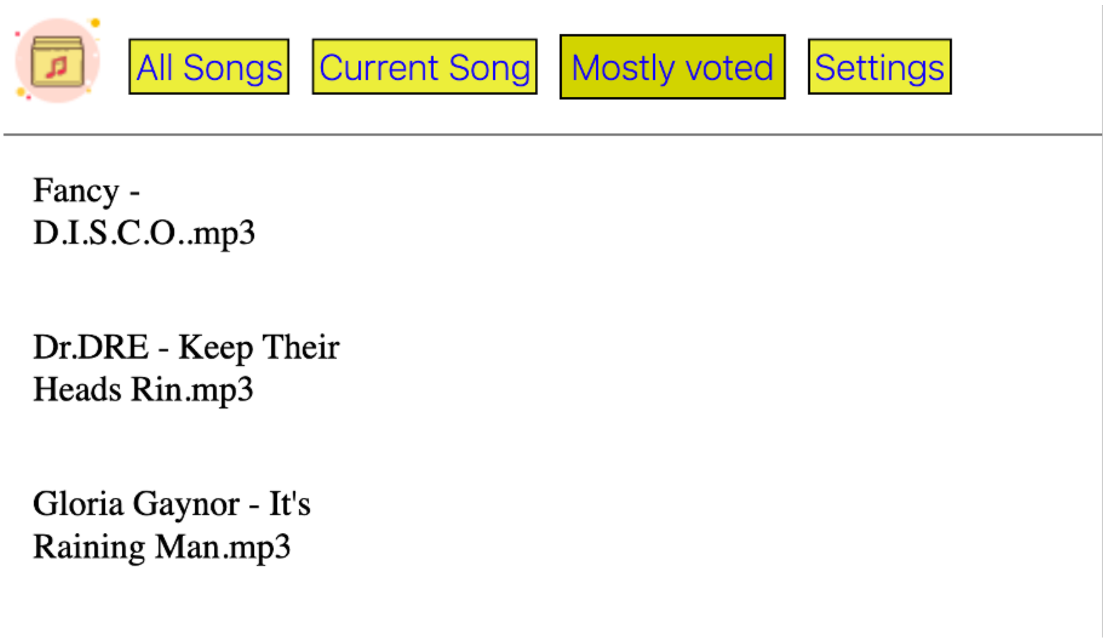


Fancy - D.I.S.C.O..mp3

Figure 6.12: Current song

6.13 Mostly voted

Admin can see a list of mostly voted songs.



Fancy - D.I.S.C.O..mp3

Dr.DRE - Keep Their Heads Rin.mp3

Gloria Gaynor - It's Raining Man.mp3

Figure 6.13: Mostly voted

7 Mobile App GUI and functionality overview

In this project for Mobile App we used Flutter framework. Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase. Flutter apps are written in the Dart language and make use of many of the language's more advanced features. For our project we have been creating one application which we can use in both Android and IOS platforms.

7.1 Main page

First page is our Main page for MVS Mobile App. In this page for now we have 4 sections: All Songs, Start Voting, Song Upload, Login. With using these sections, we can go further pages.

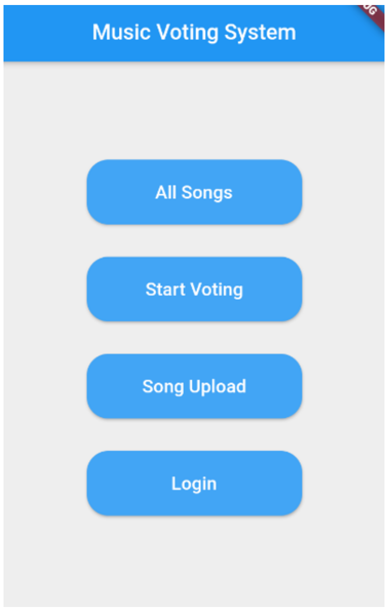


Figure 7.1: Main page

7.2 All songs

This Page is basically for listing all songs in server. Cards were used for all information about songs. As we can see from below picture all details about song was demonstrated.

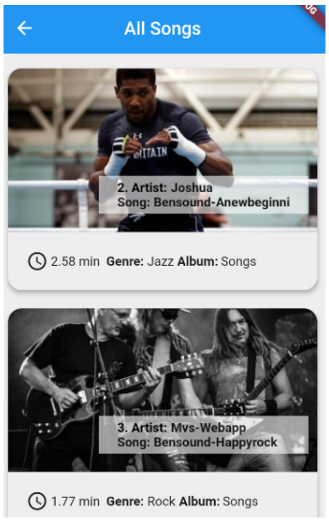
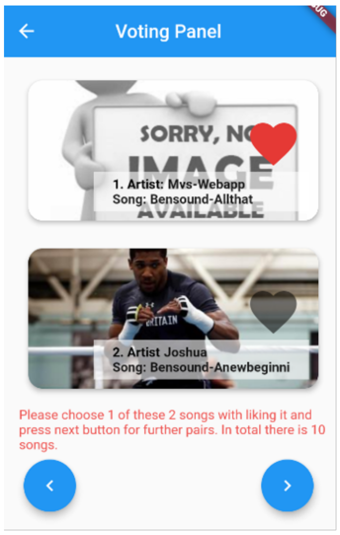

All Songs
2. Artist: Joshua
Song: Bensound-Anewbeginni
2.58 min Genre: Jazz Album: Songs
3. Artist: Mvs-Webapp
Song: Bensound-Happyrock
1.77 min Genre: Rock Album: Songs

Figure 7.2: All songs

7.3 Voting

This part is important for us, because in this section users can vote for liked by them songs. Voting page is in the form of slides. With next and back button users can go to next pairs of song for voting. Favorite button is for choosing liked song for further voting.



1. Artist: Mvs-Webapp
Song: Bensound-Allthat

2. Artist Joshua
Song: Bensound-Anewbeginni

Please choose 1 of these 2 songs with liking it and press next button for further pairs. In total there is 10 songs.

Figure 7.3: Voting

7.4 Song upload

In this page user can upload his songs from phone to server and later could be used by users for voting.

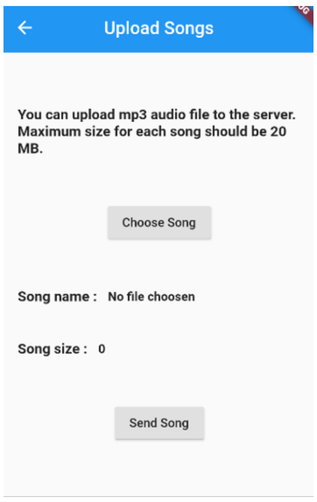

Upload Songs
You can upload mp3 audio file to the server. Maximum size for each song should be 20 MB.
Choose Song
Song name : No file choosen
Song size : 0
Send Song

Figure 7.4: Song upload

7.5 Login

Login section is for signing in and for registration user in server.

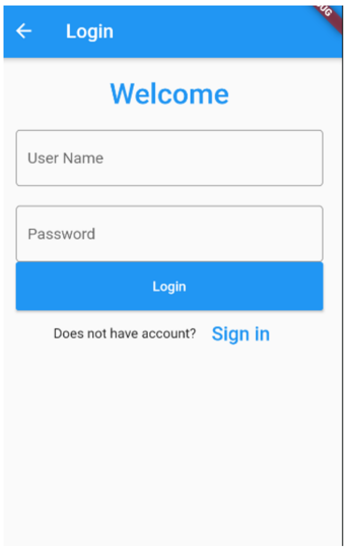


Figure 7.5: Login

8 Server-side code overview

As UML Class diagram for the whole Java project will be to huge(it can be found as pdf file in project root directory), we will provide a diagram and description of the most important classes.

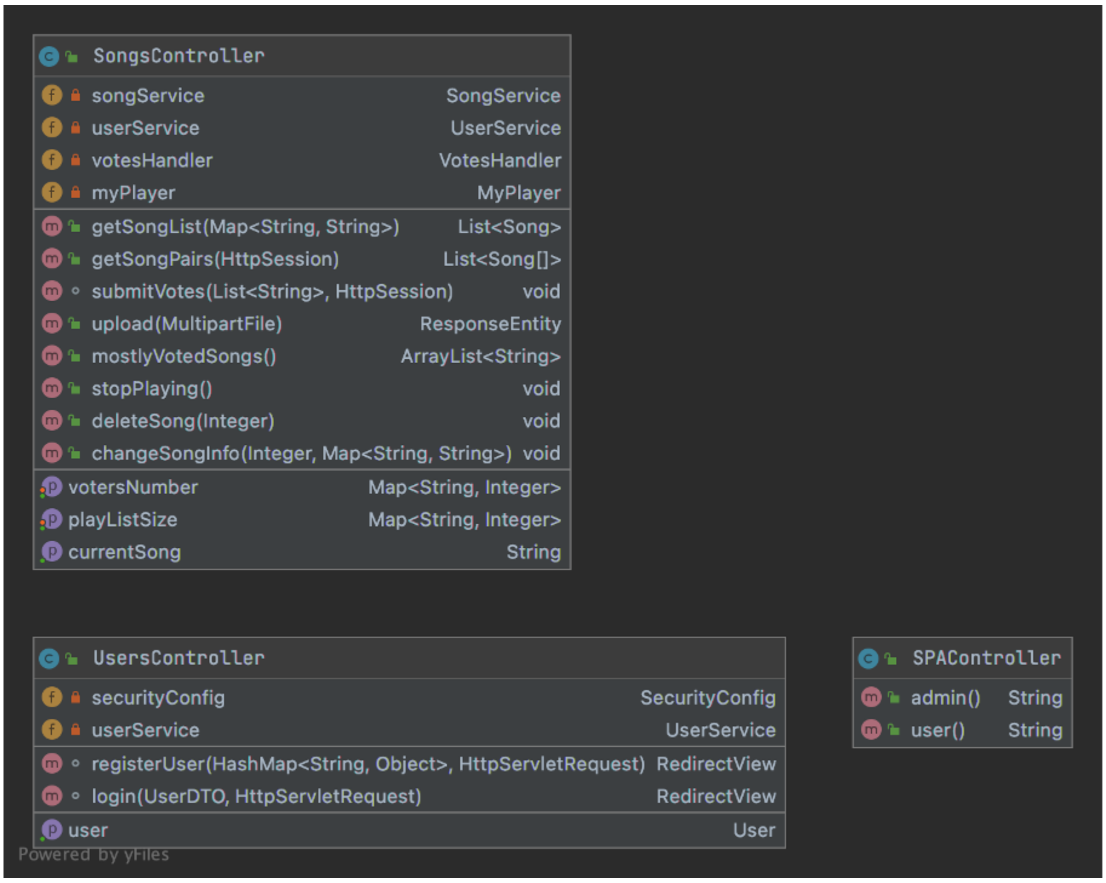

SongsController
songService SongService
userService UserService
votesHandler VotesHandler
myPlayer MyPlayer
getSongList(Map<String, String>) List<Song>
getSongPairs(HttpSession) List<Song[]>
submitVotes(List<String>, HttpSession) void
upload(MultipartFile) ResponseEntity
mostlyVotedSongs() ArrayList<String>
stopPlaying() void
deleteSong(Integer) void
changeSongInfo(Integer, Map<String, String>) void
votersNumber Map<String, Integer>
playListSize Map<String, Integer>
currentSong String
UsersController
securityConfig SecurityConfig
userService UserService
registerUser(HashMap<String, Object>, HttpServletRequest) RedirectView
login(UserDTO, HttpServletRequest) RedirectView
user User
SPAController
admin() String
user() String
Powered by yFiles

Figure 8.1: Controller package UML diagram

Controllers are the connection point between Client and Server. Controller accepts http requests from client and sends back some data. The main controller is Song Controller, it has various methods that have access to the database with song information. Also it handles the voting and playing process. User controller is responsible for users authorization. SPAController (Single Page Application) – redirects browser paths to a single page. E. g. mvs:8080/registration and mvs:8080/allSongs are redirected to index.html. It makes it possible to follow Single Page App architecture on the frontend.

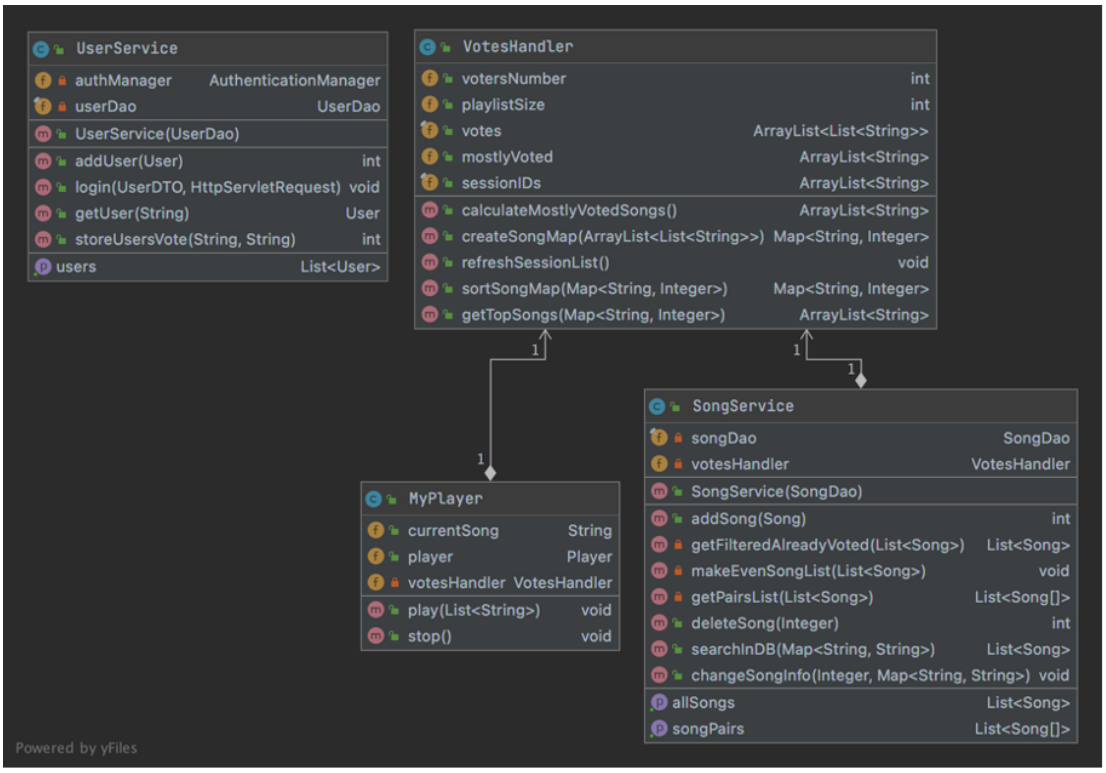

UserService
authManager AuthenticationManager
userDao UserDao
UserService(UserDao)
addUser(User) int
login(UserDTO, HttpServletRequest) void
getUser(String) User
storeUsersVote(String, String) int
users List<User>
VotesHandler
votersNumber int
playlistSize int
votes ArrayList<List<String>>
mostlyVoted ArrayList<String>
sessionIDs ArrayList<String>
calculateMostlyVotedSongs() ArrayList<String>
createSongMap(ArrayList<List<String>>) Map<String, Integer>
refreshSessionList() void
sortSongMap(Map<String, Integer>) Map<String, Integer>
getTopSongs(Map<String, Integer>) ArrayList<String>
1
1
1
1
MyPlayer
currentSong String
player Player
votesHandler VotesHandler
play(List<String>) void
stop() void
SongService
songDao SongDao
votesHandler VotesHandler
SongService(SongDao)
addSong(Song) int
getFilteredAlreadyVoted(List<Song>) List<Song>
makeEvenSongList(List<Song>) void
getPairsList(List<Song>) List<Song[]>
deleteSong(Integer) int
searchInDB(Map<String, String>) List<Song>
changeSongInfo(Integer, Map<String, String>) void
allSongs List<Song>
songPairs List<Song[]>
Powered by yFiles

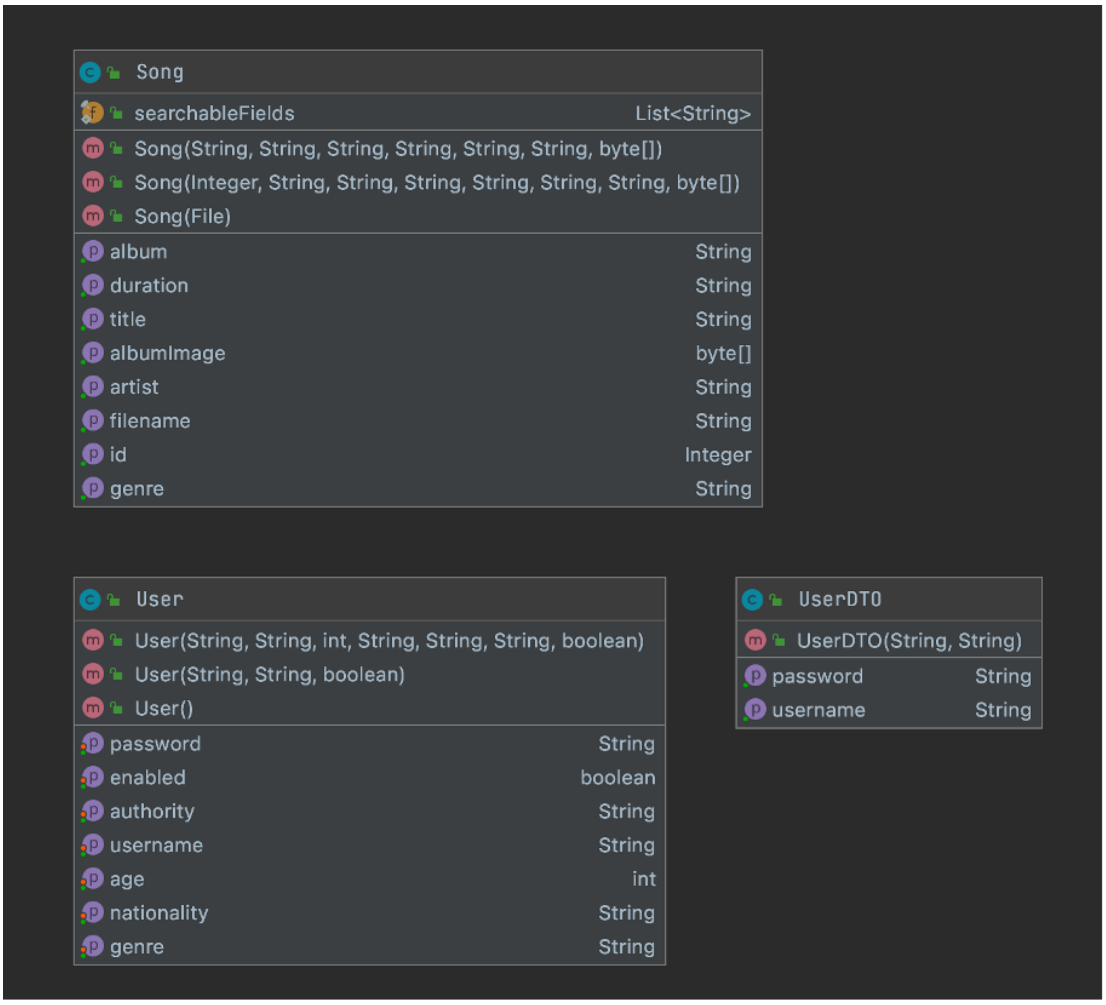
Figure 8.2: Services package UML diagram

VotesHandler – stores settings: voters' number and playlist size. It accepts votes from users and calculates a playlist accordingly.

MyPlayer – plays mp3 files using 3rd party library(JavaZoom Player)

SongSevice – high layer to perform CRUD operations on songs (create, read, update, and delete).

UserSevice - performs CRUD operations on users info and provides logging in.



Song	
searchableFields	List<String>
Song(String, String, String, String, String, String, byte[])	
Song(Integer, String, String, String, String, String, String, byte[])	
Song(File)	
album	String
duration	String
title	String
albumImage	byte[]
artist	String
filename	String
id	Integer
genre	String

User	
User(String, String, int, String, String, String, boolean)	
User(String, String, boolean)	
User()	
password	String
enabled	boolean
authority	String
username	String
age	int
nationality	String
genre	String

UserDTO	
UserDTO(String, String)	
password	String
username	String

Figure 8.3: Models package UML diagram

Song – represents a song.

User – represents a user.

UserDTO – this class is used to transfer user data from client to server during logging in.

9 Database

The database stores the data of songs and users. SQLite database has been used for the project. SQLite is a perfect database engine for providing lightweight local data storage for an individual application. It does not use client-server architecture as MySQL(it consists of a single file and all reads and writes are provided on a single file), but provides enough functionality for our app. It works great for a medium-traffic web-applications.[1]

[1]https://sqlite.org/whentouse.html

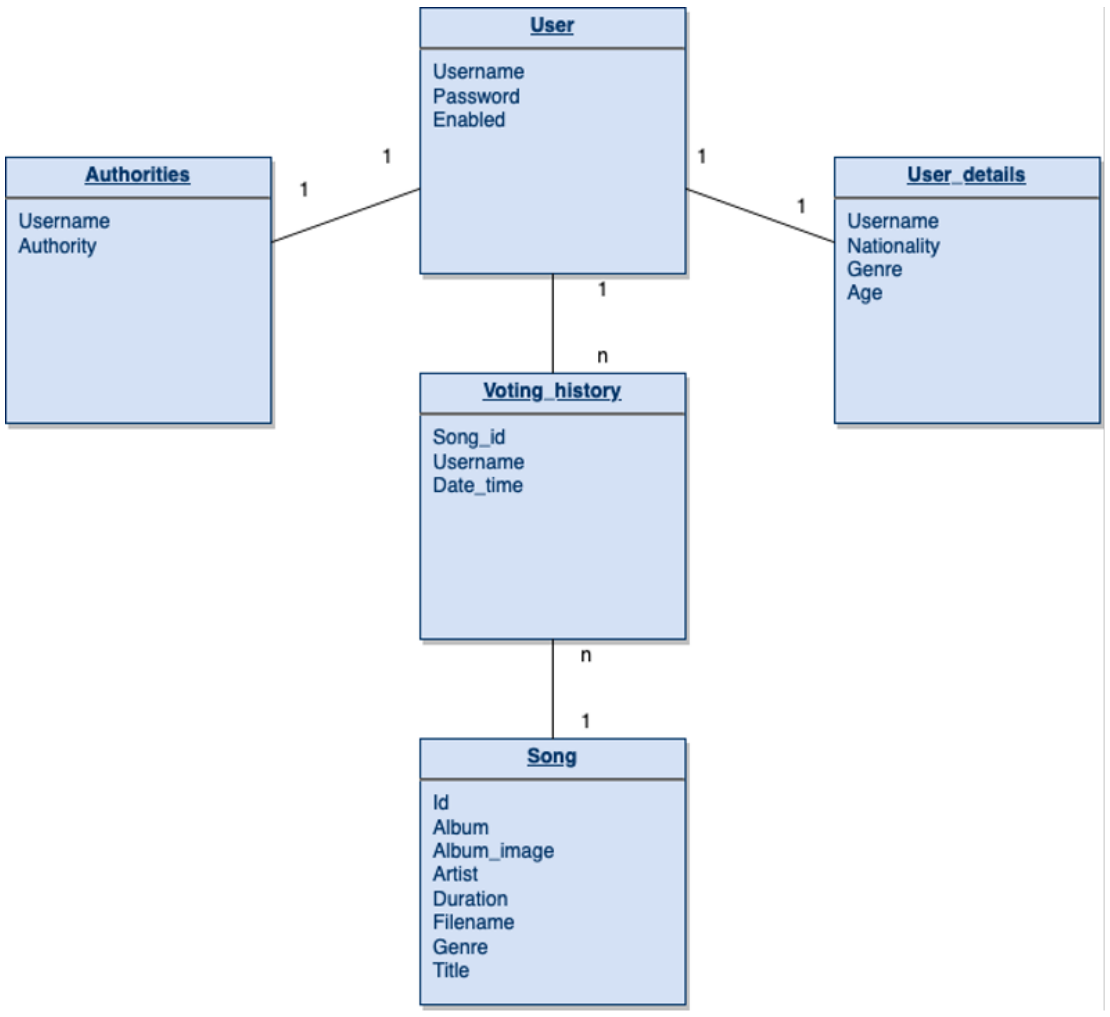


Figure 9.1: Database scheme

10 Summary

As an outcome of the elective project a web- and mobile app has been developed. The working server prototype has planned functionality and can be ran on any Linux, Windows or Mac computer and mobile app can be installed both on Android and iOS devices. The application is open-ended and functionality can be extended in future. The git repository will be opened for public use. It is important to be mentioned that the second achieved goal is the team work. Project development process was successfully integrated with Git version control system and YouTrack task tracking system. All the participants had a chance to get familiar with these tools that are common in real development process.

Bibliography

Spring Boot in action - Craig Williams, 2016

Rod Johnson - Professional Java Development with the Spring Framework, 2005

Thomas, M.T. - React in action, 2018

Jon Loeliger - Version control with Git, 2009

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 15. March 2021 Mykhailo Svyrydovych, Elvin Mammadov, Parisa Ostadzadeh