Elective Project CJ1

Music Voting Server

Mykhailo Svyrydovych

Elvin Mammadov

Parisa Ostadzadeh

Inhaltsverzeichnis

[1. Introduction 3](#_Toc65432465)

[2. Motivation and objectives 4](#_Toc65432466)

[3. Preparation 4](#_Toc65432467)

[Gantt Chart 4](#_Toc65432468)

[Requirements 4](#_Toc65432469)

[Risks 5](#_Toc65432470)

[Technologies 5](#_Toc65432471)

[Frameworks and libraries 5](#_Toc65432472)

[Task tracking 6](#_Toc65432473)

[Version Control 7](#_Toc65432474)

[4. Use case diagram 8](#_Toc65432475)

[5. Client-server architecture and REST API 8](#_Toc65432476)

[Project REST API overview 9](#_Toc65432477)

[6. Browser App GUI and functionality overview 10](#_Toc65432478)

[Main Page 10](#_Toc65432479)

[All Songs 10](#_Toc65432480)

[Song Upload 11](#_Toc65432481)

[Voting 11](#_Toc65432482)

[Search 12](#_Toc65432483)

[Song Info 13](#_Toc65432484)

[Registration 13](#_Toc65432485)

[Log In 14](#_Toc65432486)

[Admin Navbar 14](#_Toc65432487)

[Admin song editing 14](#_Toc65432488)

[Admin Settings 14](#_Toc65432489)

[Current Song 14](#_Toc65432490)

[Mostly voted 15](#_Toc65432491)

[7. Database 15](#_Toc65432492)

[8. Mobile App GUI and functionality overview 16](#_Toc65432493)

[Main Page 16](#_Toc65432494)

[All Song 17](#_Toc65432495)

[Voting 18](#_Toc65432496)

[Song Upload 18](#_Toc65432497)

[Login 19](#_Toc65432498)

# Introduction

The topic of this project is development of a Web Application that can be ran on a Raspberry PI[[1]](#endnote-1) microcomputer. The application can be installed on an offline server (without Internet connection) and run entirely inside a local Wi-Fi network. The use case 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.

Users‘ Devices

Speakers

Web Application (backend)

Linux (Raspbian)

Raspberry PI

Laptop

Android

iOS

Admin laptop

General project scheme

# 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 timeframe. We used it to plan our work. The second tool that we used was YouTrack[[2]](#endnote-2) - 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 integrations 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.

# Preparation

## 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).[[3]](#endnote-3) 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 at the end of this report.

## Requirements

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

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

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

## 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 annex.

## 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 microcomputer, 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)

### Frameworks and libraries

Any web application can be generally divided into two parts: backend (running on the server) and frontend(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 backend 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 backend 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 frontend is JavaScript. And we decided to use React framework as it has very useful official documentation with tutorial and it is the most popular frontend 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 multiplatform 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.  
//Elvin – write about Flutter technology and why did we choose it for the project

### Task tracking

JetBrains YouTrack is a web-based issue tracking and project management platform.[[4]](#endnote-4) 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.[[5]](#endnote-5)

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).  
  
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

YouTrack Dashboard

Ein Bild, das Text, Screenshot, Monitor, Bildschirm enthält.

Automatisch generierte Beschreibung

YouTrack Kanban Board

### 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 codebase from human errors and makes bug fixing easier. [[6]](#endnote-6)

#### Benefits of version control systems

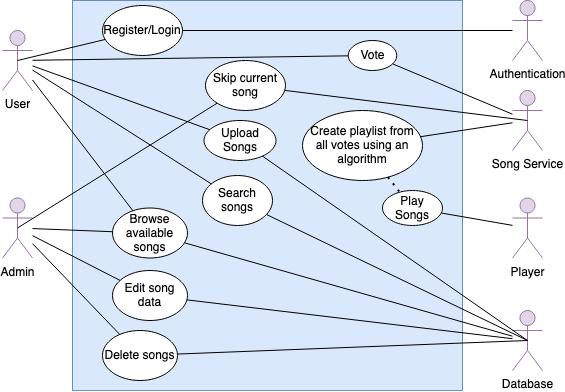
* Getting long-term history of every file
* Branching and merging. Allows to keep multiple streams of work separated for a while with a possibility to merge.
* Being able to trace each change mode to the program and track it with task tracking tools[[7]](#endnote-7)

#### 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 – 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.[[8]](#endnote-8) 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).

# Use case diagram



# 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.[[9]](#endnote-9)

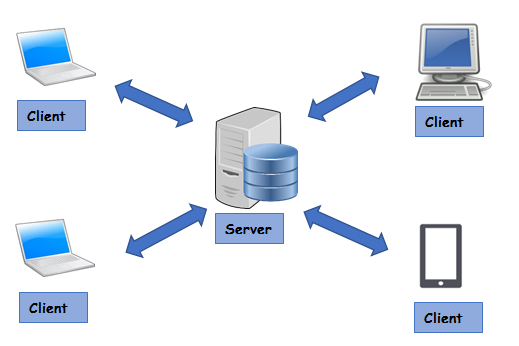


Figure. Client-server architecture[[10]](#endnote-10)

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 of functions to allow communication between computer programs and make data exchange. Nowadays REST API architecture pattern are widely used.[[11]](#endnote-11)

Generally, REST API architecture should conform following principles[[12]](#endnote-12):

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

## 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 |

# 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.

## Main Page

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

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## 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.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## Song Upload

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

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## 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.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## Search

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

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## Song Info

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

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

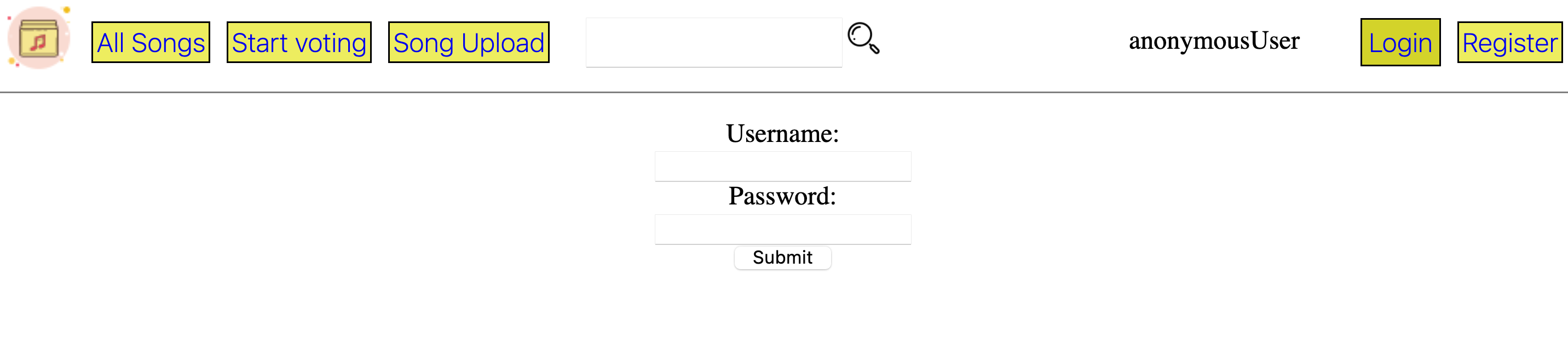
## Registration

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

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## Log In



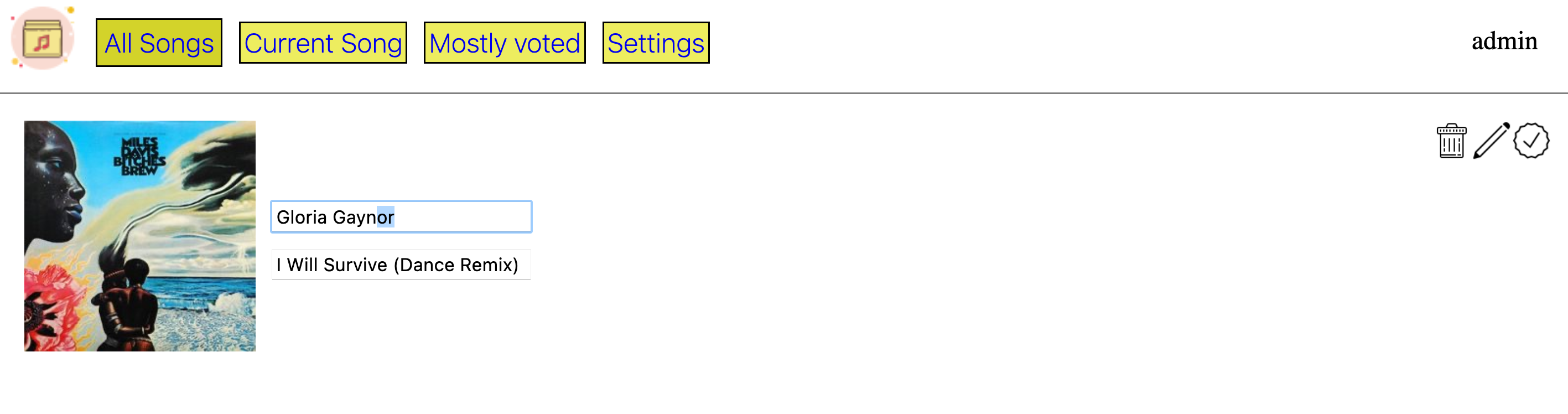
## Admin Navbar

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



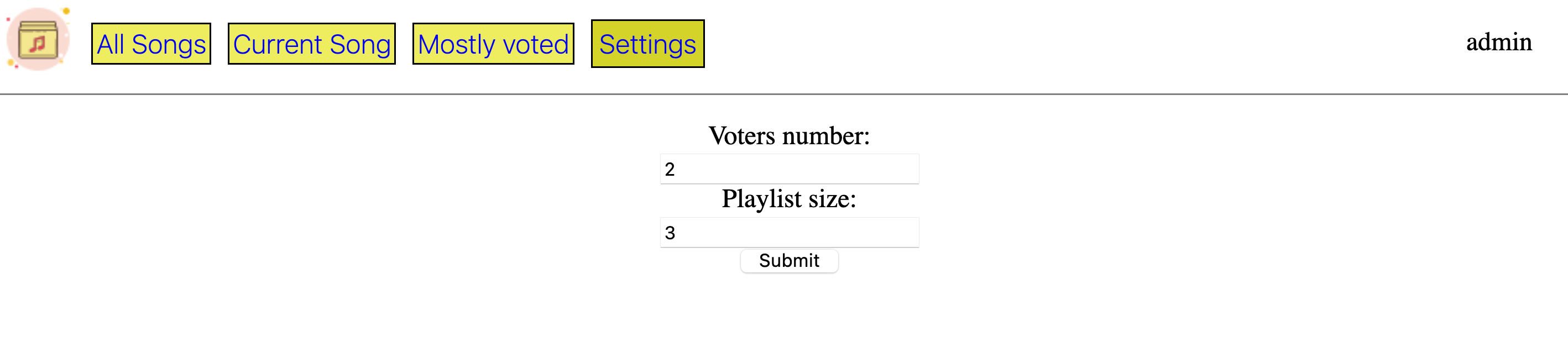
## Admin song editing

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



## 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.



## Current Song

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

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## Mostly voted

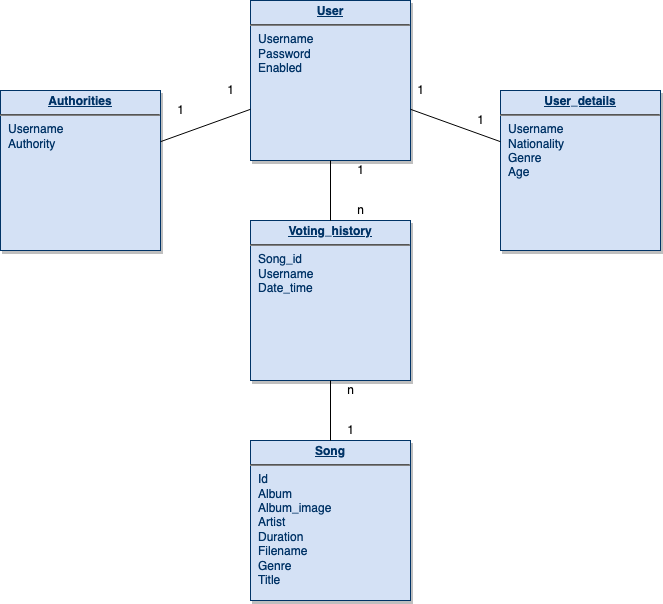
Admin can see a list of mostly voted songs.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

# 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.[[13]](#endnote-13)



# 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.

Below sections are just draft version of Mobile App.

## Main Page

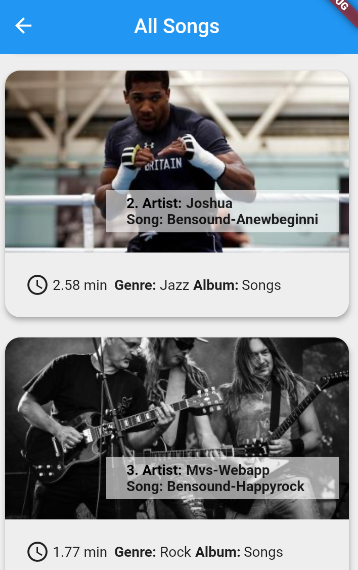
First page is our Main page for MVS Mobile App. In this page for now we have 4 sections: All Songs, Start Voting, Song Upload, Login. With using these sections, we can go further pages.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## All Song

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.



## 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.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## Song Upload

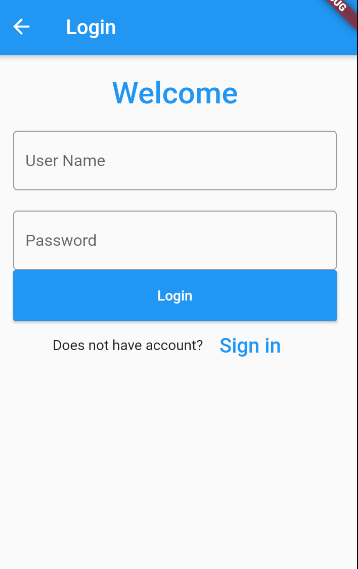
In this page user can upload his songs from phone to server and later could be used by users for voting.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## Login

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



# 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.  
  
Ein Bild, das Text, Screenshot, schwarz enthält.

Automatisch generierte Beschreibung  
  
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.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

1. https://www.raspberrypi.org [↑](#endnote-ref-1)
2. https://www.jetbrains.com/youtrack/ [↑](#endnote-ref-2)
3. https://en.wikipedia.org/wiki/Gantt\_chart [↑](#endnote-ref-3)
4. https://www.jetbrains.com/help/youtrack/standalone/YouTrack-Documentation.html [↑](#endnote-ref-4)
5. https://en.wikipedia.org/wiki/Kanban\_(development)#Kanban\_boards [↑](#endnote-ref-5)
6. https://www.atlassian.com/git/tutorials/what-is-version-control [↑](#endnote-ref-6)
7. https://www.atlassian.com/git/tutorials/what-is-version-control [↑](#endnote-ref-7)
8. https://bitbucket.org/product/version-control-software [↑](#endnote-ref-8)
9. https://www.educative.io/courses/learn-rest-soap-api-test-automation-java/JEK9Mv7RLA9 [↑](#endnote-ref-9)
10. https://www.omnisci.com/technical-glossary/client-server [↑](#endnote-ref-10)
11. Marc Masse - REST API design rulebook [↑](#endnote-ref-11)
12. https://restfulapi.net/rest-architectural-constraints/ [↑](#endnote-ref-12)
13. https://sqlite.org/whentouse.html [↑](#endnote-ref-13)