Software Requirements Specification

Online audio streaming website

Prepared by Popescu Alexandru Iulian

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Popescu Alexandru	03.03.2021	First iteration of the document	0.5

Popescu Alexandru	10.03.2021	Big refactoring	1.0

1. Introduction

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for the "Audio Streaming Website" (ASW) software. It will illustrate the purpose and complete declaration for the development of the system. It will also explain system constraints, interface and interactions with other external applications..

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team

1.4 Product Scope

The streaming website is a web-based application where people can listen to music that is already added by an admin into the database. They can play it, favorite certain liked songs and sort them into albums. There will also be a profile for each account where the user can customize his/her info. The stretch goal is to also make a page where the user will have a quiz: a random song from their database will be played and he/she will have to guess the song name.
Besides the database, the users will have to go through an authentication step in order to access the functionalities. The code is split into 3 layers:

- Data Access
- Logic
- UI

1.5 References

[1] Get started with ASP.NET Core MVC
[2] Bootstrap 3 Tutorial
[3] Font Awesome
[4] https://stackoverflow.com/
[5] SRS: Software Requirement Specifications Basics – BMC Software | Blogs

2. Overall Description

2.1 Product Perspective

The block diagram is composed of a SQL database, the authentication part and 3 layers of code:

- Data Access
- Logic
- UI

The admins will be able to upload songs from their own device. They can edit details about the song (description, album). The user can favorite, sort them by certain criteria and play any song.

2.2 Product Functions

With the website, the admins will be able to upload songs from their own device into the database. They can edit details about the song (description, album). The user can favorite and sort them by certain criteria. The stretch goal is to also make a page where the user will have a quiz: a random song from their database will be played and he/she will have to guess the song name.

2.3 User Classes and Characteristics

There will be two types of users: administrators that can upload songs to the website and users that can listen to them, add them to their favorites or add them to certain playlists.

2.4 Operating Environment

This program will operate in the following operating environment:
• Apple Mac OS X
• Linux/Unix
• Microsoft Windows

2.5 Design and Implementation Constraints

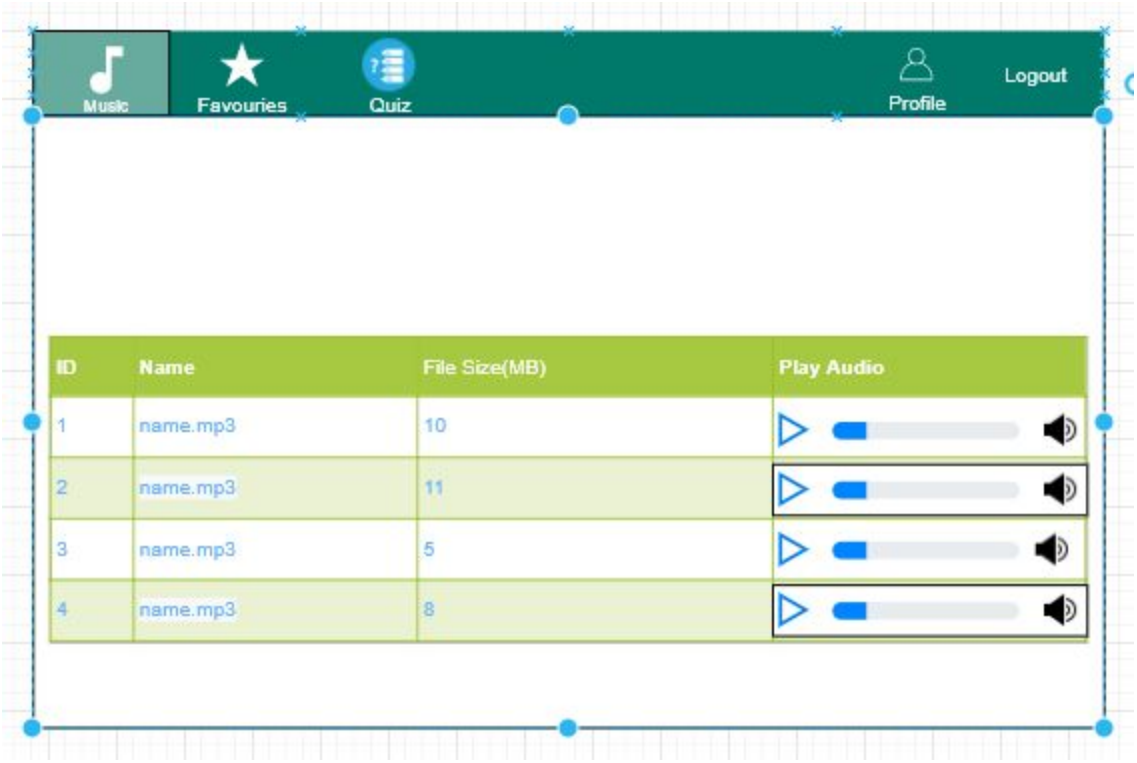
The program is created using C# programming language as well as JavaScript, HTML and CSS.

2.6 Assumptions and Dependencies

3. External Interface Requirements

First iteration of the wireframe for the UI:

3.1 User Interface



3.1 Hardware Interfaces

3.2 Software Interfaces

The software interface should follow the Model-View-Controller (MVC) model for rendering and modeling data objects.

3.3 Communications Interfaces

The communication architecture must follow the client-server model. Communication between the client and server should utilize a REST-compliant web service and must be served over HTTP Secure (HTTPS).

4. System Features

ID: FR1

TITLE: Admin login
DESC: The admin will be able to login and upload songs from their device

ID: FR2

TITLE: User registration
DESC: The user must provide user-name, password and email address. The user can choose to provide a regularly used phone number.

ID: FR3

TITLE: User log-in
DESC: Given that a user has registered, then the user should be able to log in to the application.

ID: FR4

TITLE: Retrieve password
DESC: Given that a user has registered, then the user should be able to retrieve his/her password by email.
DEP: FR1

ID: FR5

TITLE: Search
DESC: Given that a user is logged in to the mobile application, then the first page that is shown should be the music search page. The user should be able to search for a song, according to several search options.
The search options are Name, Length*, Genre and Author. There should also be a free-text search option.
DEP: FR4

ID: FR6

TITLE: Search result in a list view
DESC: Search results can be viewed in a list. Each element in the list represents a specific song. Each element should include the song name, singer, genre and a music playback device. There should be maximally 100 results displayed. If the result contains more songs than what can be displayed on the screen at one time, the user should be able to scroll through them.
1. Song name
2. Singer
3. Genre
4. Playback
RAT: The way results should be displayed in a list.

ID: FR7

TITLE: Admin upload
DESC: The admin should be able to upload songs from their device to the database

ID: FR8

Title: Admin editing
Desc: The admin can edit the songs description: name, singer, etc.

ID: FR9

Title: Admin ban
Desc: The admin can ban certain accounts

ID: FR10

Title: Albums and filtering
Desc: Users can add songs to their custom albums and filter songs using certain criteria

ID: FR11

Title: Music streaming
Desc: All users should be able to play the songs that exist in the database

ID: FR12

Title: Quiz
Desc: Users can listen to a random song from the database and they will have to guess the title

ID: QR1
TITLE: Prominent search feature
DESC: The search feature should be prominent and easy to find for the user.

DEP: none

ID: QR2
TITLE: Usage of the search feature
DESC: The different search options should be evident, simple and easy to understand.
RAT: In order for a user to perform a search easily.
DEP: none

ID: QR3
TITLE: Usage of the result in the list view
DESC: The results displayed in the list view should be user friendly and easy to understand. Selecting an
element in the result list should only take one click.
DEP: none

ID: QR4
TITLE: Usage of the information link
DESC: The information link should be prominent and it should be evident that it is a usable link. Selecting the information link should only take one click.
DEP: none

3.3.6 Response time

ID: QR5
TAG: ResponseTime
GIST: The fastness of the search
SCALE: The response time of a search

3.5.1 Reliability

ID: QR6
TAG: SystemReliability
GIST: The reliability of the system.
SCALE: The reliability that the system gives the right result on a search.

3.5.2 Availability

ID: QR7
TAG: SystemAvailability
GIST: The availability of the system when it is used.
SCALE: The average system availability (not considering network failing).

ID: QR8
TITLE: Internet Connection

DESC: The application should be connected to the Internet.
DEP: none

Security

ID: QR9
TAG: UserLoginAccountSecurity
GIST: Security of accounts.
SCALE: If a user tries to log in to the web portal with a non-existing account then the User should not be logged in. The user should be notified about log-in failure.

ID: QR10
TAG: AdminLoginAccountSecurity
GIST: Security of accounts.
SCALE: If an admin tries to log in to the web portal with a non-existing account then the admin should
not be logged in. The admin should be notified about log-in failure.

ID: QR11
TAG: UserCreateAccountSecurity
GIST: The security of creating account for users of the system.
SCALE: If a user wants to create an account and the desired user name is occupied, the user should be asked to choose a different user name.

ID: QR12
TAG: UserCreateAccountSecurity
SCALE: If a user wants to create an account and the desired username is occupied, the users should be asked to choose a different user name.

3.5.4 Maintainability

ID: QR13
TITLE: Application extensibility
DESC: The application should be easy to extend. The code should be written in a way that it favors implementation of new functions.
DEP: none

ID: QR14
TITLE: Application testability
DESC: Test environments should be built for the application to allow testing of the applications different
functions.

DEP: none

3.5.5 Portability

I**D: QR15**
TITLE: Application portability
DESC: The application should be portable with any browser
DEP: none

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Checking the fact that the system must perform as what every user expects. So in every action-response of the system, there are no immediate delays. In case of opening windows forms, of popping error messages and saving the settings or sessions there is delay much below 2 seconds, In case of opening databases, sorting questions and computing there are no delays and the operation is performed in less than 2 seconds for opening ,sorting, computing > 95% of the files

5.2 Safety Requirements

In case of a potential loss of connection between the client and the server the clients test progress so far is lost. When the client finishes its test(by pressing the finish button) then its progress is sent to the server and logged.

5.3 Security Requirements

ASP.NET Core contains features for managing authentication, authorization, data protection, HTTPS enforcement.

ASP.NET Core Identity:

Is an API that supports user interface (UI) login functionality.
Manages users, passwords, profile data, roles, claims, tokens, email confirmation, and more.

5.4 Software Quality Attributes

Availability: Checking that the system always has something to function and always pop up error messages in case of component failure. In that case the error messages appear when something goes wrong so as to prevail availability problems.

Usability: Checking that the system is easy to handle and navigates in the most expected way with no delays. In that case the system program reacts accordingly and transverses quickly between its states.

Functionality: Checking that the system provide the right tools for editing question databases, creating session tests and analyzing the test sessions. In that case the tools that the Database editors are the ones that provide that attribute.