**Music Manager**

The goal of the exercise is to make a music manager web application. A Music Manager Web Application allows you to browse through a music library, manage songs and genres: add new entries and delete them (so the usual Create, Update and Delete stuffs (CRUD)).

Users can use a web browser for access to application. Communication between web GUI and server should go over RESTful methods.

On the server side, we expect the application to be deployed in a Tomcat application server. The Server side implementation is a POJO which is using Hibernate to interact with the database.

**Which tools should I use?**

We don't care too much but probably the easiest choice is Eclipse. For the build you can use Maven or just rely on the support of the IDE to build a working package. For manage your codes, you can use Github, git or SVN.

**How should the GUI look?**

The GUI can be very basic. I want to see a table, which lists all music files on the landing page. There should be a menu bar with an 'Add music' and “Add Genres” action, which will start a wizard to add a new entry. For each entry added, we will have an edit and delete button for it. Functions also should handle validation steps.

*Landing page/home page:*

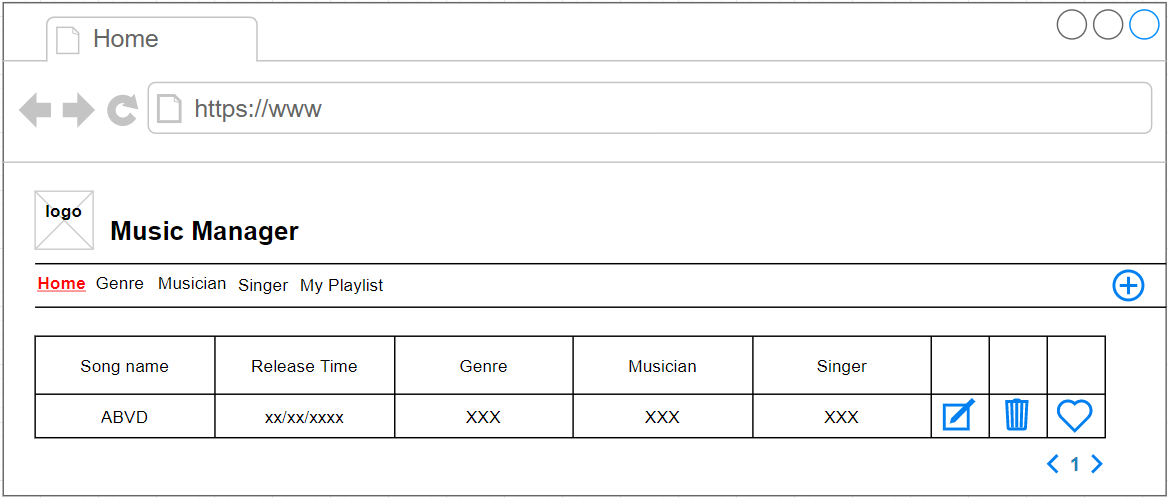
The landing page is the first page when loading a web app. On that page, we will show a list of songs including name, release time, genre, musician, and singer. Users can add new, modify, or delete songs by clicking on a button.

When deleting a song, a popup will appear to get the confirmation. Users also can add a song to My Playlist.

If the song is added to My Playlist. Users cannot delete them. They will get a message error.

Any update on the landing page also auto-updates to other client views.

Landing page will have a paging function. It will load about 10 songs per page.

**

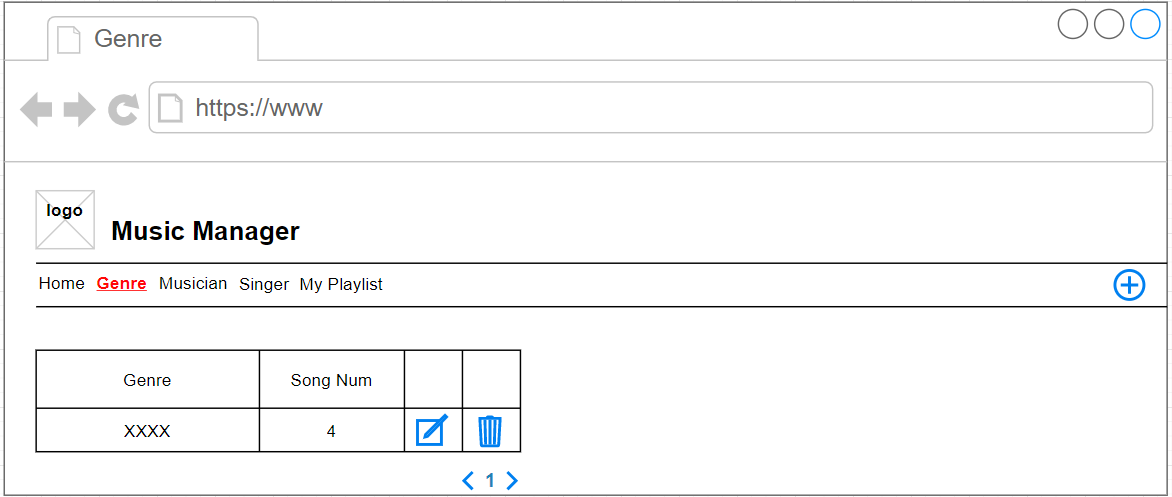
*Genre page:*

This page will show the list of genres and its songs.

When deleting a genre, a popup will appear to get the confirmation. If we have any songs mapping to genre, an error message will appear when deleting.

When adding a new genre, we need to validate that there is no existing genre (including upper/lower case).

This page will have a paging function. It will load about 10 genres per page.

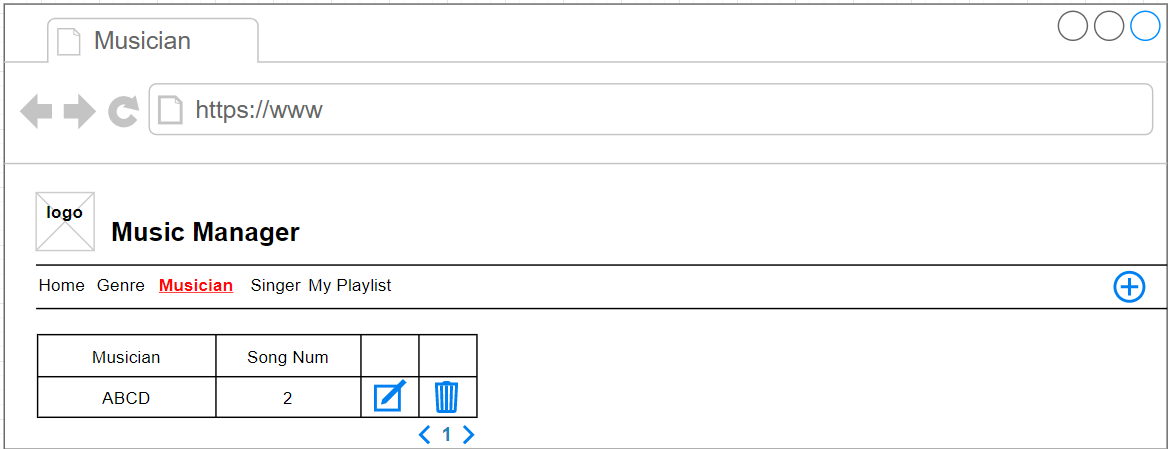
**

*Musician page:*

This page will show the list of musicians and their songs.

When deleting a musician, a popup will appear to get the confirmation. If we have any songs mapping to the musician, an error message will appear when deleting.

This page will have a paging function. It will load about 10 musicians per page.

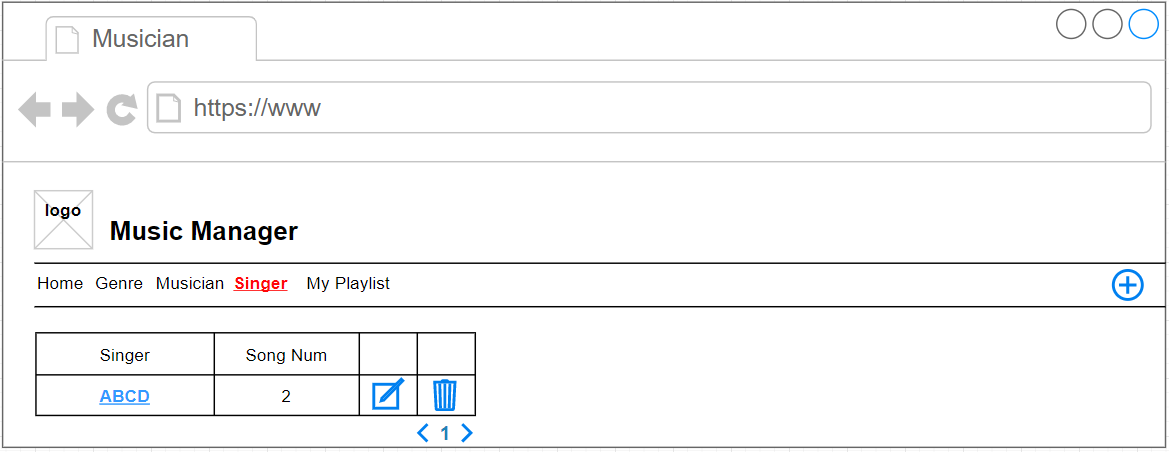
**

*Singer page:*

This page will show the list of Singers and their songs.

When deleting a singer, a popup will appear to get the confirmation. If we have any songs mapping to the singer, an error message will appear when deleting.

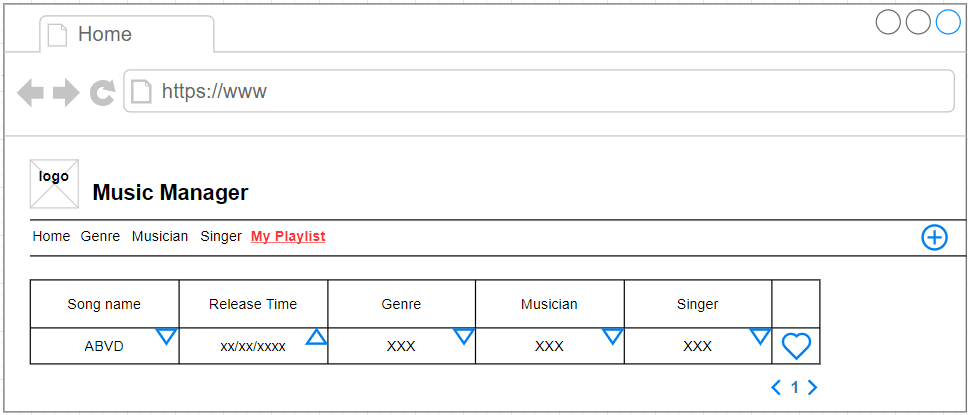
This page will have a paging function. It will load about 10 singers per page.

**

*My Playlist page:*

This page will display a list of songs. Which are added to the playlist by the user. Users also can remove a song from their playlist, a confirmation popup will appear when removing. This list also supports short songs by columns.

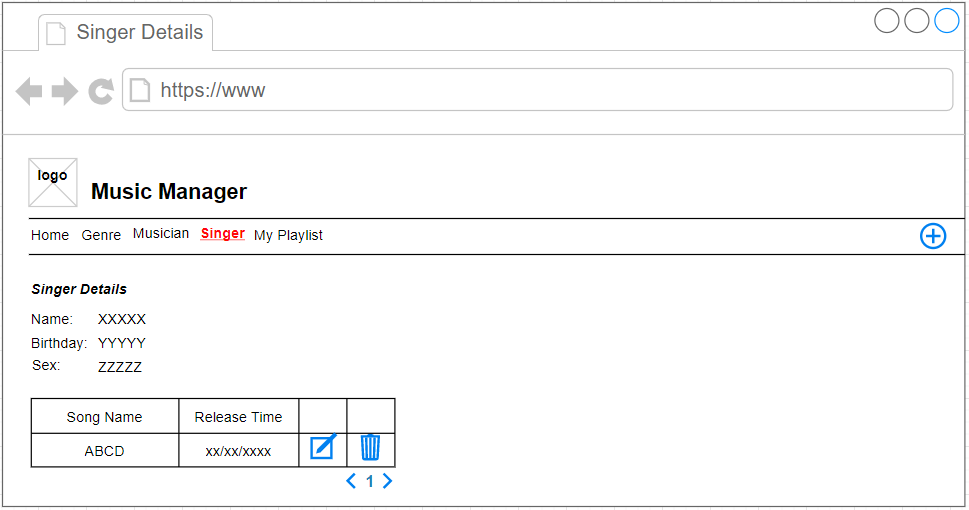
This page will have a paging function. It will load about 10 songs per page.



*Singer Details page:*

This page will display info of the singer and list of their songs. Users also can modify/remove a song, a confirmation popup will appear when removing.

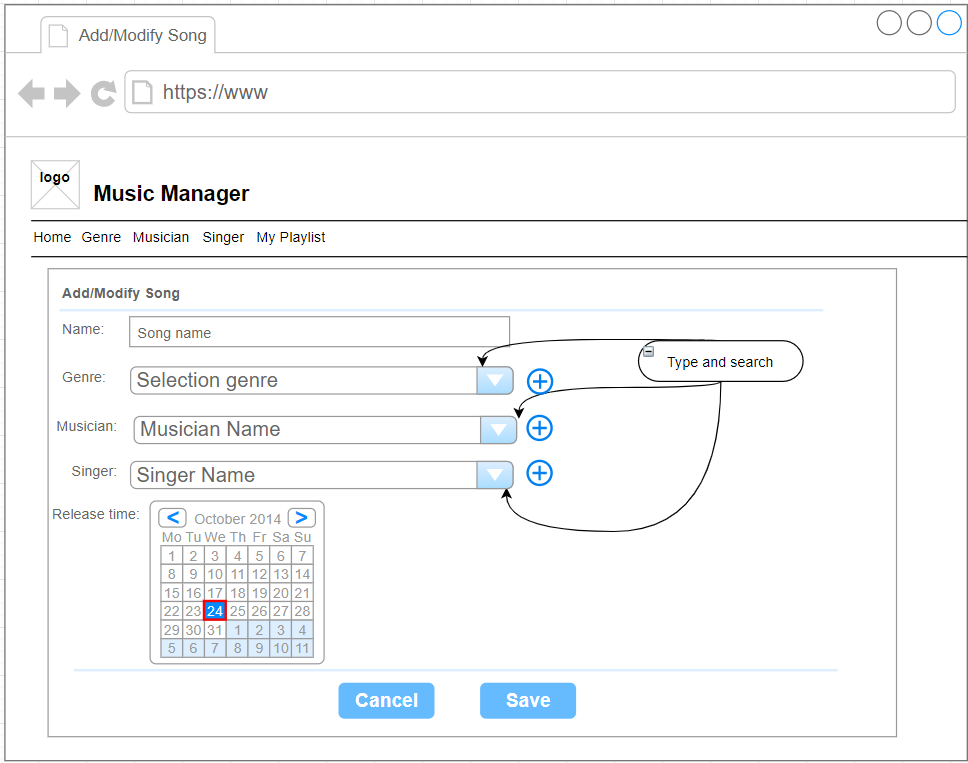
This page will have a paging function. It will load about 10 songs per page.



*Add/Modify Song:*

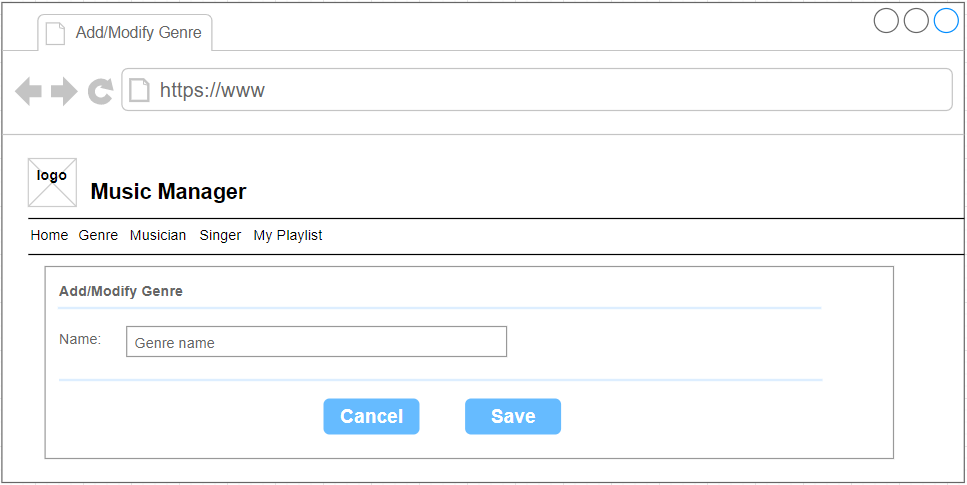
This page will help to add/modify a song.

Users also can add new gener/musician/singer when they don’t exist. A popup or navigating page for adding will be shown when clicking on the add button.

**

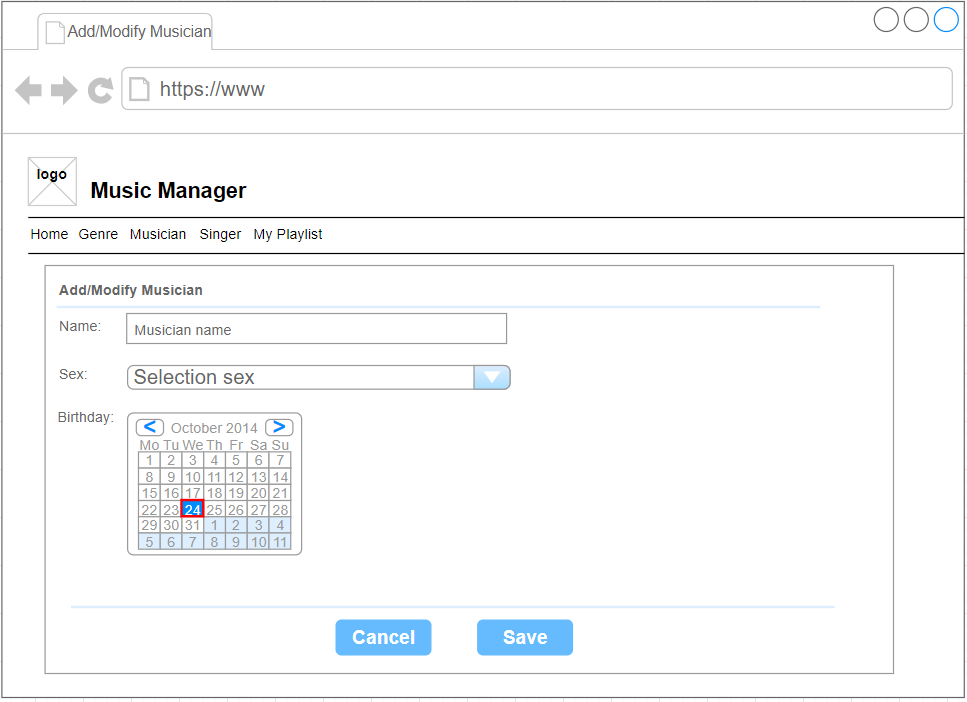
*Add/Modify Genre:*

This page will help to add/modify a genre. Users cannot add a duplicated genre including upper/lower case.

**

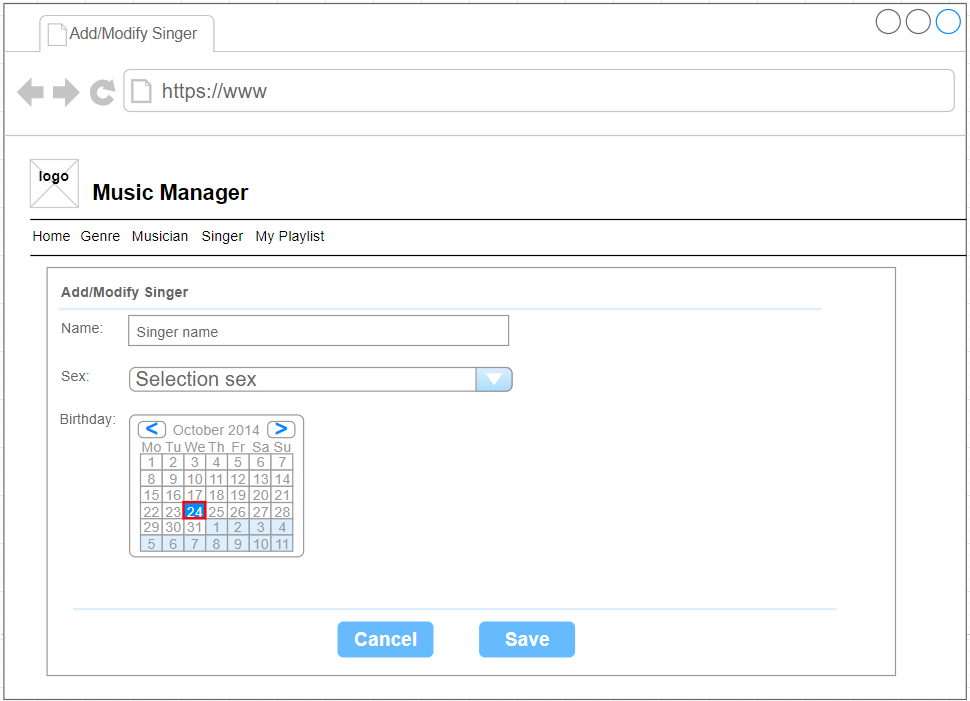
*Add/Modify Musician:*

This page will help to add/modify a musician. Users cannot add a duplicated musician including upper/lower case.



*Add/Modify Singer:*

This page will help to add/modify a singer. Users cannot add a duplicated singer including upper/lower case.



Technology requirements

**Java basic:**

Overview of Java architecture and basic concepts of OOD

Be able to implement Object Oriented based applications

Be able to design and implement effective Java programs

**Networking programming in Java:**

Network fundamental concepts

Be able to implement RESTful API

**Design Pattern:** understand common patterns: Singleton, Factory, Decorator, Proxy and Adapter.

**FreeMaker :**

Be able to implement GUI templates by FreeMaker.

**Hibernate:**

Overview of Hibernate Architecture

Be able to define configuration data (Can use annotations)

Understand POJO programming model and common Hibernate APIs for database operations

Understand HQL (Hibernate Query Language) and native SQL

**Spring framework/Spring boot**: understand and be able to apply two new concepts: IOC (Inversion of Control) and AOP (Aspect Oriented Programming).

**JUnit:**

Overview of JUnit

Be able to implement a JUnit class for verification of the implementation class.

The implementation codes need to cover more than 90%.

**JavaScript (Angular/AngularJS/ReactJS/VueJS):**

Overview of javascript.

Understand and be able to implement a web GUI by (AngularJS/ReactJS).

**Tomcat Application Server:**

Be able to configure (stop, start …etc) the server.

Deploy/Undeploy an application on the server.

**MySQL DBMS**