SE 3XA3: Development Plan
DJS

Team 12, DJS
Victor Velechovsky - velechva
Amandeep Panesar - panesas2
Taha Mian - miantm

December 8, 2016

Contents

List of Tables

List of Figures

This document describes the requirements for DJS. The template for the Software Requirements Specification (SRS) is a subset of the Volere template (Robertson and Robertson, 2012). If you make further modifications to the template, you should explicity state what modifications were made.

1 Project Drivers

1.1 The Purpose of the Project

The purpose of this project is to make it easier for people that attend social gatherings or events, to select songs and form their own playlist according to the mood or preference of the attendees. The current implementation (PlayMyWay) has an unflattering and difficult to use UI, as well as no easy way for the average person to integrate the software into their party. We plan on making a revised version that has an elegant web app interface, and an easy to install server.

Social gatherings are much more enjoyable when most of the attendees enjoy the music that is being played. This project was inspired by

Social gatherings are much more enjoyable when most of the attendees enjoy the music that is being played. This project was inspired by

1.2 The Stakeholders

1.2.1 The Client

Event Organizer
The client is the host of the social event or gathering, who is trying to save money by not hiring a DJ and simply relying on this system, which allows the attendees to chose what songs they would like to hear. Having users select music will put less stress on the event organizers, and allow them to focus on other aspects of the event, or let them enjoy themselves.
DJ
The client can also be a DJ, hired by a party planner, who is not willing to put up with people fighting over what song to play next.

1.2.2 The Customers

Event Attendee's
If the system is working properly and attendee's are voting for songs, then the event will be more enjoyable for them.

1.2.3 Other Stakeholders

No other stakeholders have been discovered.

1.3 Mandated Constraints

- The front-end of the product will take the form of a web-app that can run on any javascript-enabled browser. This means the app should be able to accommodate all common operating Systems (mobile and desktop) and all common Javascript-enabled browsers. Internet Explorer may be exempt
- The front-end web-app and server need to both be connected to the same WiFi network.
- The server needs to be stable with very low downtime, in order to prevent scenarios where the music stops playing accidentally.

1.4 Naming Conventions and Terminology

Shorthand	Explanation
JS	Javascript
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
Event	Any event that includes shared music listening. Party, Wedding, etc.
UI	Acronym for User Interface
BUC	Acronym for Business Use Case

1.5 Relevant Facts and Assumptions

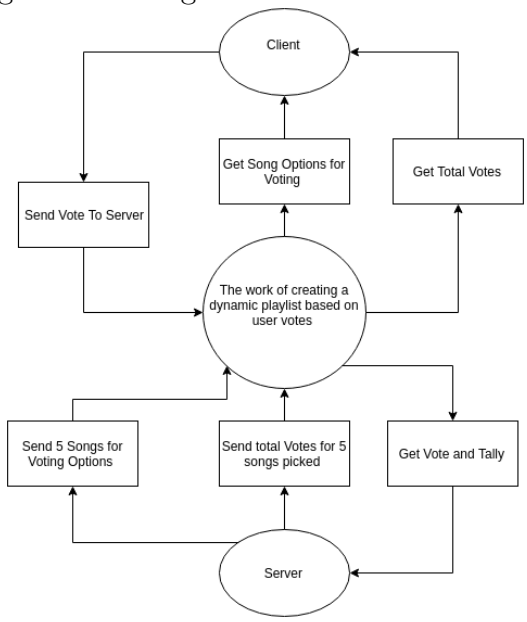
- We assume that users of the app are impatient (they don't want to spend a long time learning the software)

2 Functional Requirements

2.1 The Scope of the Work and the Product

2.1.1 The Context of the Work

Figure 1: A diagram of the context of work



2.1.2 Work Partitioning

Event Name	Input and Output	Summary of BUC
Front Facing Website Interaction such as Voting	User Interaction (in)	Record the user's interaction (voting) on the website counting up
Front Facing Website Displays Votes	Total Votes (out)	Transmit Total Votes To Front Facing Website
Calculate Most Votes and Select Song	Total Votes (in) Song Selected (out)	Get Total Votes and Show Song Selected On Front End
Reset Votes	Reset Votes (out)	Server should reset votes and counter

2.1.3 Individual Product Use Cases

The project will be used primarily for playing music decided by a dynamic playlist. The playlist is generated by the server picking five unique random songs. The list of songs picked by the server is then shown to the user where any song can be picked. The voting system is based on most votes to one song, thus if a song is the most voted the product will play that song. After the highest voted song has been selected and has been playing for more than thirty seconds the votes will be reset and the next song will be picked using the same process.

2.2 Functional Requirements

- The web page created in HTML and Javascript will display graphics and other information to user like title.
- The Javascript will fetch the five options for music playback and will display them on the user's browser.
- The web page should also keep track of the number of votes for any song.

- The html page should also include a graphic where the user can place their vote.
- The html page should also only list valid options for music (should only list songs stored on the server).
- Cookies will also be created to only allow one vote for the user.
- After the song has been selected to play then the html page should update to only show songs that haven't been played and should reset votes.
- The server should pick 5 random unique songs until no unique songs are left and repeat process.
- The server should total votes and share the information with the user through the html page.
- The server should sort the songs after voting and play the most voted song.
- The server should also remember what song the user picked and then reset after voting has reset.
- The server should extract metadata from the music files and display the album art along-side the buttons in the web interface

3 Non-functional Requirements

3.1 Look and Feel Requirements

3.1.1 Appearance Requirements

Non-Functional requirement	
Description :	The product front end must be visually appealing and have no lag between interactions.
Rationale :	The user should not be frustrated because of poor design.
Fit Criterion :	The user interface should be clean and simple. The buttons used must clearly be distinguishable and labels should be clear of spelling mistakes.

3.1.2 Style Requirements

Non-Functional requirement	
Description :	The product should have a modern UI interface with bright colors.
Rationale :	The colors will add to features and buttons being distinguishable. Furthermore, the product will create a brand by introducing custom color (Facebook Blue).
Fit Criterion :	The product should look like something from 2016 and not look like something from the 1990s html era.

3.2 Usability and Humanity Requirements

3.2.1 Ease of Use Requirements

Non-Functional requirement	
Description :	The product should be easy to use and not confuse users by having simple text describing how the product works.
Rationale :	Users might not know what or how to vote so a label or walkthrough might help them understand the uses and limitations of the product.
Fit Criterion :	The product should have text explaining how voting works and how to vote for a certain song.

3.2.2 Personalization and Internationalization Requirements

Non-Functional requirement	
Description :	Any type of genre and song can be preloaded to the server.Furthermore the user can vote for any song they prefer.
Rationale :	Users can ask event planners to preload the server with songs to allow users to personalize the song list.
Fit Criterion :	Check the type of songs preloaded on the server.

3.2.3 Learning Requirements

Non-Functional requirement	
Description :	Should be easy to use.
Rationale :	All users should be able to easily vote for their favourite song.
Fit Criterion :	The product must show simple UI design to promote simplicity which allows for ease of use.

3.2.4 Understandability and Politeness Requirements

Non-Functional requirement	
Description :	The concept should be explained in the tutorial or instructions page clearly to allow understandability for the user.
Rationale :	All users should understand how the product works.
Fit Criterion :	The product must include an instructions, help, or walk-through page.

3.2.5 Accessibility Requirements

Non-Functional requirement	
Description :	The product should be easily accessed by users through the local wifi.
Rationale :	The users should be able to know where to go for the application to be able to use it.
Fit Criterion :	The URL that lets users votes must be short and easy to remember.

3.3 Performance Requirements

3.3.1 Speed and Latency Requirements

Non-Functional requirement	
Description :	The product should display and send votes the server quickly with no delay.
Rationale :	The users need to know if their song will be played next and the total amount of votes for the selected song.
Fit Criterion :	Product should show votes in real time with no delay or latency.

3.3.2 Safety-Critical Requirements

Not Applicable

3.3.3 Precision or Accuracy Requirements

Non-Functional requirement	
Description :	The song selected with the most amount of votes should be played.
Rationale :	The user expects that the most voted song should be played.
Fit Criterion :	The server should check to see if the song playing is the same as the most voted song.

3.3.4 Reliability and Availability Requirements

Non-Functional requirement	
Description :	The server should always be playing music and should never have any down time
Rationale :	Users want to listen to music for the entire event or session.
Fit Criterion :	The product should record the uptime to see if the server ever goes down or resets.

3.3.5 Robustness or Fault-Tolerance Requirements

Non-Functional requirement	
Description :	The server should only play music that is available on the server.
Rationale :	The users should only be able to pick from music that has been preloaded onto the server.
Fit Criterion :	The product should only show list of music that is readily available on the server.

3.3.6 Capacity Requirements

Non-Functional requirement	
Description :	The server should be able to store 32GB of music.
Rationale :	The users want a wide variety of music.
Fit Criterion :	The product should have more then one song stored in the desginated folder.

3.3.7 Scalability or Extensibility Requirements

Non-Functional requirement	
Description :	The server should allow for atleast 300 users.
Rationale :	The events users will attend will consist of hundreds of guests and can reach into the thousands.
Fit Criterion :	The product should have more then one song stored in the desginated folder.

3.3.8 Longevity Requirements

Non-Functional requirement	
Description :	The web interface and server should always be up until the event coordinator manually turns off each service.
Rationale :	The user should be able to vote for songs until the event the user is attending is over.
Fit Criterion :	The product should not turn off until event coordinator specifies.

3.4 Operational and Environmental Requirements

3.4.1 Expected Physical Environment

The physical environment does not effect non functional requirements

3.4.2 Requirements for Interfacing with Adjacent Systems

Non-Functional requirement	
Description :	The web browser should fluently communicate with the server and record user interaction.
Rationale :	The web browser should let the server know what the user wants a certain song to be played.
Fit Criterion :	The server should show and log each interaction with the web browser.

3.4.3 Productization Requirements

Non-Functional requirement	
Description :	The product should have a runnable install script for easy distribution, for at least one OS either Windows, Linux, or MacOS.
Rationale :	The admin should be able to install the product quickly
Fit Criterion :	The install script should install the product with no errors.

3.4.4 Release Requirements

The product will only be released once unless an OS update corrupts the product

3.5 Maintainability and Support Requirements

3.5.1 Maintenance Requirements

3.5.2 Supportability Requirements

3.5.3 Adaptability Requirements

3.6 Security Requirements

3.6.1 Access Requirements

Only one vote	
Description :	A user should only be allowed to vote once per 'round' of votes.
Rationale :	A single user should only be able to vote once since that's how democracy works
Fit Criterion :	Try and vote multiple times to check that it's not possible safeguards

Restarting phone/WiFi cannot bypass voting	
Description :	Restarting the phone or the WiFi should not allow the user to vote multiple times
Rationale :	A single user should only be able to vote once since that's how democracy works
Fit Criterion :	Try and vote multiple times by restarting phone, logging out, restarting WiFi and see if any of these methods break the safeguards

User access	
Description :	Only users attending the event should be able to vote
Rationale :	External actors should not influence the election
Fit Criterion :	Try to access the event without being connected to the WiFi

3.6.2 Integrity Requirements

3.6.3 Privacy Requirements

Private server data	
Description :	No private server data should be visible to anyone but the administrator
Rationale :	Server data should not be compromised
Fit Criterion :	Penetration testing

Other user's data	
Description :	No private data about other users should be visible to anyone but the administrator
Rationale :	User's sensitive data should not be compromised
Fit Criterion :	Penetration testing

3.6.4 Audit Requirements

Not Applicable

3.6.5 Immunity Requirements

Not Applicable

3.7 Cultural Requirements

There are no cultural requirements for this project.

3.8 Legal Requirements

3.8.1 Compliance Requirements

Non-Functional requirement	
Description :	Music that is played by someone should have legal rights to be played publicly
Rationale :	It is illegal to play music that you do not own the rights to play
Fit Criterion :	We use a website that offers Royalty free music for the testing that can found here

Non-Functional requirement	
Description :	Make sure that the original project we are trying to recreate allows us to look at the original project and take some things from their project.
Rationale :	We have to make sure we do not copy someone else's idea because it is illegal to copy work that you do not have the rights to.
Fit Criterion :	The open project we are recreating has an open MIT license that can be found here. We are allowed to use their project in any way we like.

3.8.2 Standards Requirements

There are no standard requirements for this particular project.

3.9 Health and Safety Requirements

This section is not in the original Volere template, but health and safety are issues that should be considered for every engineering project.

4 Project Issues

4.1 Open Issues

The most important issue right now is how exactly do we make sure that a user can only vote once per song, without requiring people to sign up for an account. We also want the user to be able to change their vote multiple times when selecting the next song and it is in queue.

4.2 Off-the-Shelf Solutions

The project that we are modeling (PlayMyWay) already does most of what our project will do.

As for other solutions, there are various libraries that we will use in our development. These libraries will help with various functionality, and include (but are not limited to):

- Express.JS (Server framework for Node.JS)
- Angular.JS (Front-end framework for the webapp)
- nodeunit (Unit testing package)
- Mocha (general testing package)

4.3 New Problems

DJ.Js is based off the open source project called PlayMyWay that can be found here. We are going to recreate the project, in javascript. The original project was written in Jade, which is a Object Oriented programming language based on Java. Jade is kind of outdated and not as universal as javascript, which is the golden standard for web page applications. We don't need any new installations just a device that can access the internet and has a internet browser that runs javascript.

4.4 Tasks

We are given an outline of the things we need for this project which includes a proof of concept, testing plan, a design, and a final presentation. The proof of concept will be early in the project but will allow us to make a very basic

outline of the project like be able to get requests from a device to a server vice versa. The test plan will tell us how we are going to test our product so we know whether or not the project has successfully fulfilled our requirements The design of the project will be more specific to coding and will come later but can be broken down in these simple steps:

1. Build a node.js sever using Amandeeps Raspberry Pi for the
2. Build the UI using javascript
3. Testing (unit,general testing)

The Final Presentation will be the last step in the project when it is complete and fully functional.

4.5 Migration to the New Product

There is no transition needed because we are recreating a web app, that already exists.

4.6 Risks

Many risks can occur while trying to implement our product which include:

- The product is a webapp and therefore relies on an internet connection
- Bugs or catastrophic errors in the server could cause the music to start playing sporadically
- Only people with a device that can connect to the internet can use the web app

4.7 Costs

There will be no monetary costs, the music we are using to test our webapp will be unlicensed and free to play, and because the project we are recreating is an open source project which we are free to use in any way. The only other cost is the amount of effort and time we put in the project which should not be more than 100 hours, but could take longer.

4.8 User Documentation and Training

We will integrate a small help button at the end of the webpage that will describe how the webpage works. After this short tutorial the user should know how the website works.

4.9 Waiting Room

Something that is part of our vision is having all genres of music, having a diverse list of music so users can select songs that cater to their tastes. We also want to add "moods" that correspond to certain event you are in, so a wedding would be more of a happy mood, where as a party will have a Festival will have a celebratory mood.

4.10 Ideas for Solutions

References

James Robertson and Suzanne Robertson. *Volere Requirements Specification Template*. Atlantic Systems Guild Limited, 16 edition, 2012.

5 Appendix

This section has been added to the Volere template. This is where you can place additional information.

5.1 Symbolic Parameters

The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

Table 1: **Revision History**

Date	**Version**	**Notes**
Wed. Oct. 5	0.1	Basic Outline
Wed. Oct. 5	0.2	Requirements added
Thurs. Oct. 6	0.3	Section 1 added and formatting
Thurs. Oct. 6	0.4	First draft
Thurs. Oct. 6	0.5	Formatting and minor changes
Fri. Oct. 7	0.6	First Revision complete
Sat. Oct. 8	0.7	Section 4 Complete
Mon. Oct 11	1.0	Revision 0
Dec 6	1.1	Revision 1