Software Quality Plan

Section 1: Product Information

Description:

A web application that is intended to be implemented into an audio competition organizations existing website. Used by judges at a car audio competition event to streamline the process of recording and sorting the data from each competitor. The application is designed to take most of the manual work out of tracking the names, scores, and classes of each competitor so the judge can spend more time running the meter. This can lead to a much more efficient process which will get the competitors in and out of the competition lanes much faster.

There are a total of nine (9) pages to be implemented into an organization's website; The 'main' screen, which shows a table of all previous competitions in order by date, and a button to add a new competition. The 'input' screen is where you input your new event and all subsequent scores during said event. The 'summary' screen is an overview of the current event with a table showing all scores with buttons to sort by name, class, or score. The 'standings' page is used for the secondary display at the audio event, this display faces the crowd and cycles through the scores for each class in the current event. The 'winners' screen sorts and displays the winners of each class for the judge to give out awards at the end of an event. Once an event is complete, the judge then takes the text file generated by the application and uploads it to the organization's website using the 'upload' page. Upon uploading the file the application will add the file to an archive and sort/update the database with the new scores and events. The 'backup' page is where these archived files go for auditing at the end of the audio season. All users can use the 'lookup' page to search for event information from any event in the archive. The 'tutorial' page is intended for judges to learn how to use this application in a simple step by step process.

Intended Market:

Audio organizations that hold events where audio cars are acoustically measured to see which is the loudest in a specific category.

Quality Expectations:

Depending on how big a specific audio event is, there may be a rush to get everyone through the competition lanes. So the application should not throw any fatal errors, and should be simple enough that non tech-savvy people can input values properly 100% of the time. Uptime of the application is dependent on the uptime of the organization's web platform.

Section 3: Process Descriptions

Process Descriptions:

Process descriptions are an integral part of this project. Each process must be followed to ensure that all pieces of the program are being created at the same level of quality. This also helps us manage all requests and changes to the program.

One of the most important processes will be the change request process. Change requests must be submitted on the proper, formal change request form and submitted to the team for review. Every change request must be properly reviewed by the team before moving forward or throwing the request out. If the change request is accepted, it will be ranked by priority. High priority change requests will be taken care of as soon as possible, and pushed out in an immediate update. This includes issues such as security flaws. Medium and low priority change requests will go out in the next scheduled update, and will include issues such as coding errors, spelling errors, and customer suggestions. Bad change requests such as employment requests and spam will be immediately rejected and not included in the updates. When change requests are accepted, they must be documented in the change management log by a member of the review team.

Version release processes will also be important in this project. After version release, the cycle will begin with issue reporting and collection of requests for changes or new features that will be documented as stated in the change request process. The team will sort accepted requests, and begin the planning and design stage. Written user stories will be written and compiled, and then assigned to the development team. Using the stories, the team will build the software. After, the team will use review documents to go through the code for errors and refactoring. At this stage, the new software will be run against the tests written from the software requirements. If this is successful, the new version will be pushed out in a release. Support will be in place to help the public with the new update, and the cycle will continue again with the collection of issues and change requests.

Section 4: Quality Goals

Quality Goals:

SoundScorez seeks to deliver high quality usability of the software application by vigorously testing its functional processes and system attributes. Our application's primary focus is on efficiently streamlining the scoring process of car audio events through automation while providing accurate displays of any results. From this brief overview we can then begin to breakdown the intended users of our system as well as their associated tasks. For the sake of simplicity we can organize users into two separate categories: facilitators such as the event coordinator, event judge and webmaster, and website end-users such as fans of the sport or competitors who wish to conveniently browse event results. Facilitators are concerned with the interface and functionality of portable app such as being able to create a new event instance, sorting results by categories such as competitor, score, and classes; being able to successfully upload TermLab files to the official website, and auditing results both before submission and on the actual website.

We will consider the interface design and assess whether or not the application is easy to navigate. Does every element provide a significant purpose and do those elements convey a specific meaning to the user? Will first time users be able to grasp the flow of the process and functions during the course of an event? This is purely subjective so we are always open to any

critical feedback from our intended user audience. As for the application software itself, we must test its deployment over a wide variety of devices such as both iOS and Android mobile smartphones as well as tablets with various display sizes. Encompassing that idea is our measurement of code quality, specifically portability and the ability to run our source code in different machines and platforms.

Other qualities of our code necessary for evaluation include its reliability, maintainability, testability, and reusability. Our code will be subject to numerous debugging and testing activities. This may include testing the runtime execution speed of the primary functions of our application with focus on optimization while still providing reliability. In order to measure the quality of both the code and the overall system design, it is vital to establish clear process standards throughout development as well as verifying that documentation captures the important ideas in a uniform way. As with all processes aforementioned, our organization will determine if they achieve the level of quality that meets our standards or if they require additional improvements.

Section 5: Risks and Risk Management

Key Risks:

SoundScorez key risks when it comes to the overall quality of the project come with the diverse team with different programming backgrounds and programming styles, the ground-up build style of the entire program, and the lack of communication with the end-client. The different programming backgrounds and styles can cause the code to be sloppy and unformatted, as well as creating incompatibilities between certain objects and functions within the program.

Since we are starting from scratch, we are not using any existing working systems for our program that we can work off of. Since everything is going to be built from the ground up, everything must be made in house and therefore, won't have a guarantee of working correctly. The lack of communication with the end-client can cause the program to sway off course and not deliver the perfect product to the client.

Mitigation:

- The team can write out a set of rules regarding how the code will be written and the styles that are used to make the objects and functions within the program.
- Each object and function of the code must be compatible and flexible so they can be easily implemented in other parts of the program. All of these objects and functions should be thoroughly tested to make sure they work in accordance to the team's standards before being implemented to the greater program.
- The team should make efforts to keep the client in-the-loop about everything that the team has been working on or having trouble implementing. Having a close relationship with the client will make for a more satisfactory product.