SCS31901 PROJECT PROPOSAL

Squashies Jukebox

Music Control App

Joshua Incollingo - 19049537

Themis Giaras - 19034124

Dharmjit Kailay - 17879659

# 

# 

# Executive Summary

In the heart of Liverpool, you will find one of the most prestigious gym centres in the state - that is Squashlands Gym and Fitness Centre. With a large range of equipment, wide range of supplements as well as low-priced memberships, patrons don’t think twice before they join the gym. As good as Squashlands Gym is, the feedback has been coming in regards to ways the gym can improve - and the major area in need for improvement is music.

Music is a prominent figure to anyone’s workout, and gym patrons at Squashlands prefer to have control over the music that’s being played throughout their workout. This kind of feedback is crucial for any business moving forward - failure to act upon this could result in loss of members, and potentially revenue. This calls for a jukebox application that allows gym patrons to select a song from the Squashlands local library and add it to a queue waiting for the song to be played. Should the Jukebox application be a success, this could open the door to an increase of memberships and guests coming into their gym.

The solution is a simple one - using Flask and VLC Media Player, with an Android application. An android application will be made for the gym patrons to use, so that they can use to search for the song of their choice and write their name. Once they add a song to the queue, the request will go back to the administration staff, who will have a server running picking up and sending back requests between the application and VLC Media Player, which will be playing across multiple TV screens in the gym. The patron’s selection and name will be used for the administration staff as part of their song analytics with a timestamp included, which can help them gather data on what songs are chosen at certain times of the day and how frequently.

Who are we? We are three third year ICT students studying at Western Sydney University at the Sydney City Campus. We have our fair share of experience when it comes to making applications. Over the past three years, we have worked extensively on the fundamentals of programming, design, servers and databases - which are key components needed to make this Jukebox application.

We are very excited for the prospect of working with a proud family-run business like Squashlands Gym and Fitness Centre and are confident in our ability to provide a high quality Jukebox application suitable to your requirements.

# Table of Contents

[**Executive Summary**](#_9e9n2ng5tvwk) **2**

[**Table of Contents**](#_4u8dr4dsm1l2) **3**

[**Introduction**](#_uvzme1kydsit) **4**

[**Client Details and Project Background**](#_d4142t6o8sfp) **5**

[**Problem Statement**](#_j281fomjbvbo) **6**

[**Three Alternative Solutions**](#_reqz85o10b73) **7**

[Alternative Solution 1: Mopidy (Raspberry Pi)](#_vxgfmc2u7xq9) 7

[Solution Description](#_bjcfqdxoktz) 7

[Business Case](#_q98jx3y5ebrz) 7

[Risk](#_bjcfqdxoktz) 7

[Alternative Solution 2: Flask with a Video Daemon (Windows 10 machine)](#_9p4z6r2dat2a) 8

[Solution Description](#_i7ijsdd4axkp) 8

[Business Case](#_l9j2x52lxqrp) 8

[Risk](#_i7ijsdd4axkp) 8

[Alternative Solution 3: Flask and VLC (Windows 10 Machine)](#_k87oax4xk6mz) 9

[Solution Description](#_b21owgg26d4l) 9

[Business Case](#_m4ypasyrutdq) 9

[Risk](#_b21owgg26d4l) 9

[**Recommended Solution with Justification**](#_mvccjbkwhep2) **10**

[**Rank High-Level Business Functions and Use Cases**](#_mylqsnydorsy) **10**

[**Development Release Schedule**](#_kslgjf1qf3ws) **12**

[**Conclusion**](#_9pwnoa2arsr0) **13**

# 

# Introduction

We will be identifying three possible solutions which involve different hardware and software components. The first involves Mopidy, which is a music server script that will run on the Raspberry Pi - a cheap, pocket-sized computer. The second solution and the third solution both utilize Flask, a Web Server Gateway Interface, with VLC in and Video Demon the media players.

Out of these three alternatives, we will conduct stringent requirement analysis and risks involved in each one to arrive at the recommended solution.

The report will then go on to describe the high-level business cases and use cases in a tabular form which will be the blueprint for the basic design and user testing.

The development release schedule will then describe the timeline of consecutive releases – with each release tackling business requirements in order of priority.

# 

# Client Details and Project Background

Name: Angel Georgieff

Email: [angel@pageone.net.au](mailto:angel@pageone.net.au)

Location: 1 Shaw Rd, Ingleburn NSW 2565

A very important aspect of people’s workout routine these days is music. Patrons want to be able to select songs from a playlist that they are able to workout to. Squashlands want to be able to provide this for their members and guests, therefore the idea came for the Squashie’s Jukebox App.

This jukebox application should allow gym patrons to queue for songs on the Squashland’s playlist. For the administration staff, they will be able to manage the song playlist and control the song queues from the patrons, as well as gather statistics on the songs being played, what time of the day they’re being played, how many times the song has been chosen, etc.

In order to meet the client’s requirements, the project team needs to create a Python script to create the database and web server and to maintain them, and an Android tablet app which the gym members will use to choose the songs and add them to the queue. It will act as an HTTP request when the song is being chosen to the VLC, which will stream to multiple monitors across the gymnamism.

# 

# Problem Statement

There are many public places that are now starting to implement a jukebox-like app such as crowd DJ, and based on feedback, members at the Squashlands Gym and Fitness Centre prefer to have some form of control over the music selection.

This is a problem as Squashies Gym currently does not offer any way for the gym patrons to control the music playing in the gym.

Feedback from members is very important for any business. If they don’t respond to this kind of feedback, members may lose interest and not return to their gym - and when they lose customers, they lose money.

# 

# Three Alternative Solutions

## Alternative Solution 1: Mopidy (Raspberry Pi)

### Solution Description

Mopidy is a music server script that when run, allows users to play music locally or from any external applications like Spotify, SoundCloud to Google Play. It is a pre-existing application that has some very similar concepts to what we are attempting to create for this Jukebox application. It also has some open source mods that can be utilised and changed as need be.

The Raspberry Pi is a cheap, pocket-sized computer. It’s capable of doing everything you’d expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games. Due to the Raspberry Pi’s small size and price, it is a small alternative to a normal device and can do pretty much everything that any normal size device can do while taking up less room and costing much less to set up.

### Business Case

This solution allows for a more portable and cheaper alternative that has an already existing open-source music management server with many already existing extensions/modules that suit the needed application well.

### Risk

* Mopidy only works with audio, not music video files, which doesn’t meet the client’s requirements as it needs to be able to display the music videos to multiple monitors around the gym.
* Mopidy is not a well-known piece of software within the group and will involve a lot of work in order to understand how it functions

## Alternative Solution 2: Flask with a Video Daemon (Windows 10 machine)

### Solution Description

Flask is a Web Server Gateway Interface (WSGI) application framework. It is designed to help get started by making it quick and easy, while also allowing an application to easily scale up. It allows us to choose tools and libraries we want to use without any dependencies.

A daemon is a program that runs continuously in the background and whose main purpose is to handle service requests that a computer systems is waiting to receive. The daemon forwards requests to other programs as needed. The idea behind this solution is to have the Web Server script using Flask handle all the requests between the patrons and staff, as well as the statistics and alerts, as well as a daemon running in the background to handle any requests with displaying the music videos.

### Business Case

This solution allows for a more easier to manage system and an easily expandable/moddable application. On top of this the system uses a daemon which through running in the background, may eliminate clutter of application on the screen.

### Risk

* Creating daemon applications is something the group has never had any experience with so this may lead to issues
* Due to the need to create 3 separate applications this may increase the time taken on the overall project

## 

## Alternative Solution 3: Flask and VLC (Windows 10 Machine)

### Solution Description

Flask is a Web Server Gateway Interface (WSGI) application framework. It is designed to help get started by making it quick and easy, while also allowing an application to easily scale up. It allows us to choose tools and libraries we want to use without any dependencies.

VLC is a free open-source cross-platform media player and streaming media server. VLC has an array of interfaces such as ncurses, remote control, telnet and HTTP which allow for tasks to be controlled automatically or remotely. This feature is perfect in this solution as it allows the Flask Web application to control the instance of VLC through it’s HTTP interface allowing for automatic control, as well as controlling any queue requests from the mobile app running on the tablet or from the staff’s web extension.

### Business Case

This solution allows for an easier to manage, and easy expandable/moddable application. The system uses VLC which already has pre-built playlist/queue functions as well as a media library that can be used to store all music and easily send information of its contents between processes. VLC also allows for flexibility due to the fact that it supports many media file extensions/types, so if the music video file extensions/types are changed, it shouldn’t affect the system.

### Risk

* Flask
  + Not async friendly
  + Has limited support and documentation
  + Lack of database

# 

# Recommended Solution with Justification

We recommend using the Flask Web Server framework using VLC Media Player

**Justification**

Flask is easy to pick up and easy to maintain framework. It allows for easy customization no matter what type of application you are creating/running. VLC is open-source and has many already useful modules and interfaces that can easily be integrated into our current application without the need for a lot of work and effort. Due to VLC supporting many file extensions/types, changes to the file extensions/types should not affect the system. Windows 10 is also very helpful as it is a versatile OS with an array of useful features and is easy to use.

# 

# Rank High-Level Business Functions and Use Cases

Ranked High-Level Business Function

|  |  |  |
| --- | --- | --- |
| **BF ID** | **High-Level Business Functions** | **Ranking** |
| BF1 | Song Search Management | Essential |
| BF2 | Song Queue Management | Essential |
| BF3 | Staff Queue Management | Essential |
| BF4 | Ad Management | Essential |
| BF5 | Song Analytics (Statistics) | Essential |

Use Cases

|  |  |
| --- | --- |
| **Function** | **Type** |
| 1. **Song Search Management**    1. Patron is required to enter their name on tablet    2. Patron searches up a song through a search dialog | Essential |
| 1. **Song Queue Management**    1. When a patron finds the song they’re looking for, they select the song to add it to the queue awaiting to be played | Essential |
| 1. **Ad Management**    1. A pre-recorded announcement is played every 15 minutes, unless there is a need for an emergency announcement | Essential |
| 1. **Song Analytics**    1. When a patron selects a song and then records their name to the app, the details will be sent back to the main server/database and will also include the timestamp | Essential |
| 1. **Staff Queue Management**    1. When a song is being queued and waiting in the playlist, admin staff will have the ability to go into the playlist and remove any songs for whatever reason | Essential |

# 

# Development Release Schedule

|  |  |
| --- | --- |
| **Case** | **Released Schedule** |
| Song Search Management | 5/12/2019 |
| Song Queue Management | 5/12/2019 |
| Ad Management | 7/12/2019 |
| Staff Queue Management | 9/12/2019 |
| Song Analytics | 11/12/2019 |
| Marquee - Scrolling Message | 20/01/2020 |
| Exporting Feature CSV | 16/01/2020 |
| Requests for Songs Outside the Playlist | 13/01/2020 |

# 

# Conclusion

To summarise, in the above proposal we have covered the project background, problem statement, provided three alternate solutions, provided the recommended solution with justification, ranked the high-level business functions and use cases, and provided a release schedule. By doing all the above tasks we have gotten a better idea of the project scope and possible solutions that may greatly suit it, as well as give ourselves a release schedule to better maintain priority over certain application functions that meet the needs of the high-level business functions and use cases. To move forward with this proposal, we need to make a decision based on the provided solutions so that we may begin working on the application’s prototype.