SCC31901 SYSTEM ANALYSIS AND DESIGN REPORT

Squashies Jukebox

Music Control App

Joshua Incollingo - 19049437

Themis Giaras - 19034124

Dharmjit Kailay - 17879659

# Executive Summary

In this report, we looked to establish a relationship between the client and software team in a professional and harmonious way. The system requirements were identified to provide an understanding on what is to be expected from the solution we have devised and how it will work. The two actors - the gym patron and the gym staff - were expanded and we looked to uncover the relationship between them, the use cases and the interfaces. Risks and constraints were also identified in order to understand current and future issues within this project. We provided details of every aspect of the solution - through detailed diagrams and screen designs.

These diagrams investigated the interaction/relationship between the users, the system and any other internal/external entities. This report can be seen as a communication channel for all parties involved (the client, testers, programmers, etc.). The screen designs gave an idea of how the final solution will look and how the interactions are brought alive using buttons, text input, etc. A test plan was implemented in this report with several uses cases to be tested with successful and unsuccessful data to see how the system will respond and if it responds the way it’s supposed to.

# Table of Contents

[Executive Summary 2](#_Toc29238719)

[Table of Contents 3](#_Toc29238720)

[Introduction 4](#_Toc29238721)

[Client Statement 4](#_Toc29238722)

[Problem Statement 5](#_Toc29238723)

[System Requirements 5](#_Toc29238724)

[Functional Requirements 5](#_Toc29238725)

[Non-Functional Requirements 6](#_Toc29238726)

[Risks and Constraints 6](#_Toc29238727)

[Constraints 7](#_Toc29238728)

[Design Considerations 7](#_Toc29238729)

[System Architecture 8](#_Toc29238730)

[Software Architecture 9](#_Toc29238731)

[Network and Communication Architecture 10](#_Toc29238732)

[Detailed System Design 10](#_Toc29238733)

[Actors 10](#_Toc29238734)

[Use Case Diagram 12](#_Toc29238735)

[Expanded Use Cases 12](#_Toc29238736)

[Sequence Diagrams 20](#_Toc29238737)

[Entity Relationship Diagram 22](#_Toc29238738)

[Database Schema 23](#_Toc29238739)

[Screen Designs 23](#_Toc29238740)

[Test Plan 26](#_Toc29238741)

[Features/Use Cases to be Tested 26](#_Toc29238742)

[Candidate Test Cases/Test Data 27](#_Toc29238743)

[Conclusion 28](#_Toc29238744)

# Introduction

This report will show the client and problem statements as to what the problem is and the solution to that problem. The system will have several requirements, which will be divided into two types - functional and non-functional. Functional requirements describe what the system should do, while non-functional requirements describe how the system will work.

Risks and constraints of the project will also be discussed - the risks being current issues, whereas constraints looking at the future and what could happen. There will be a series of diagrams that will further examine the relationship between the users, the application, the client server and any other external/internal components.

Screen designs represent what the outcome will aim to look like and several test data, with a test plan, will be conducted to ensure a higher rate of efficiency and a high success rate heading towards the end of the project.

# Client Statement

Our client, Angel Georgieff has asked for a jukebox application that will allow gym members to queue for songs on the Squashlands gym playlist. This application development idea was initiated after Squashie’s gym members put forward their feedback on being able to select songs from a playlist that they can work out to. Client expects the application to fulfil the following set of functions:

* It should allow gym patrons to queue for songs on the gym playlist
* The admin staff should be able to manage the song playlist and control song queues
* In addition, admin staff will be able to gather statistics on songs that are played (times of the day it is played, the frequency, etc.)
* The requests should be processed and streamed to multiple monitors across the gym

Team will be using Python script to create the database and web server for maintenance purposes. The main application will be android based that will be operated in a tablet at the gym and it will act as an HTTP request when the song is being chosen to the VLC, which will be updated across multiple screens around the gym. Our team is set to meet client expectations and provide top quality work.

# Problem Statement

A large group of people spend majority and a significant amount of time in the gym these days. Gym are constantly required to ensure the ultimate gym experience is provided to its members. Music plays a vital role in workouts and so often gym members demand facilities that will allow them to enjoy their workout session.

Squashlands Gym and fitness is a well-known established gym who want to implement their member feedback to help improve their workout experience. Members have given feedback that they would like to have control over the music when they workout.

We will develop a jukebox like app/service to manage the playing of the music in the gym. It will entail the development of mobile /tablet app that will allow gym management to control and manage playlists of music and allow members to queue up music to play. This will allow all members to enjoy their workout experience and create a better environment inside the gym.

# System Requirements

## Functional Requirements

|  |  |
| --- | --- |
| **ID** | **Description** |
| FR01 | Gym patrons will be able to search for songs |
| FR02 | Gym patrons will be able to add songs to the queue |
| FR03 | Staff will be able to register/login |
| FR04 | Staff will be able to search for songs |
| FR05 | Staff will be able to add songs to the queue |
| FR06 | Staff will be able to remove songs from the queue |
| FR07 | Staff will have media controls for the music player |
| FR08 | Staff will be able to view statistics of queued songs |
| FR09 | Staff will be able to broadcast audio messages at set intervals |
| FR10 | Staff will be able to broadcast emergency audio messages |
| FR11 | Staff will be able to set messages to scroll across the screen |
| FR12 | Gym patrons will be able to request songs that do not appear when searching |

## Non-Functional Requirements

|  |  |
| --- | --- |
| **ID** | **Description** |
| NFR01 | Network Connectivity |
| NFR02 | Scalability |
| NFR03 | Security |
| NFR04 | Maintainability |
| NFR05 | Usability |
| NFR06 | Reliability |
| NFR07 | Response Time |

# Risks and Constraints

|  |  |  |
| --- | --- | --- |
| **Risk** | **Resolution** | **Type of Risk** |
| Members unable to attend meetings due to other commitments (jobs, assignments); leads to heavy workload | Time management to be done by every team member, so that enough effort is put into project. | Internal |
| High quality application expected by the client with limited information available | Team members met with client in the early weeks of the project to get an understanding of what the client was after | External |
| Few members inexperienced with chosen programming language - Python | Team member needs to focus on their part and do the proper research and self-study if required | Internal |
| Testing of the final app cannot be done as the required device is with client. | Team will have to discuss with client to be able to schedule a testing day. | External |
| Clients changing their mind in regard to the scope of the project | We meet with the client and discuss what the change is and then we as a team can plan our strategy on how to tackle this change | External |

## Constraints

Wireless Network Access

Availability of reliable wireless connection is necessary to ensure efficient and effective development of the app. It will also ensure teamwork quality and proper communication between members. If the wireless connection goes down, then there will not be any communication between the app and the server

User Data Security

Security measures such as encryption will need to be considered as user data is sensitively stored in the database.

## Design Considerations

Consistency

Having all the buttons the same font style and text size or an “add'' button in a table positioned in the same spot in every column ensures there is a consistent design throughout. A consistent design is seen as more professional and makes it easier for the user to spot where things are on the app.

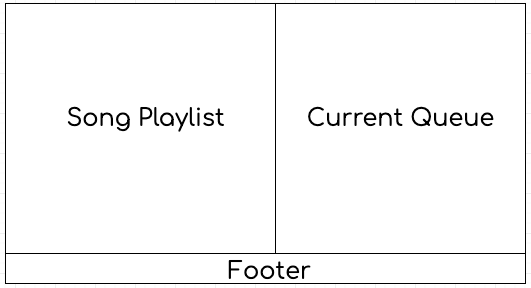
With the app, we plan to implement a consistent design, with our buttons having the “montserrat” font, which is seen as a simple yet modern font choice that is often good to use for apps and websites.

Pleasing Visual Design

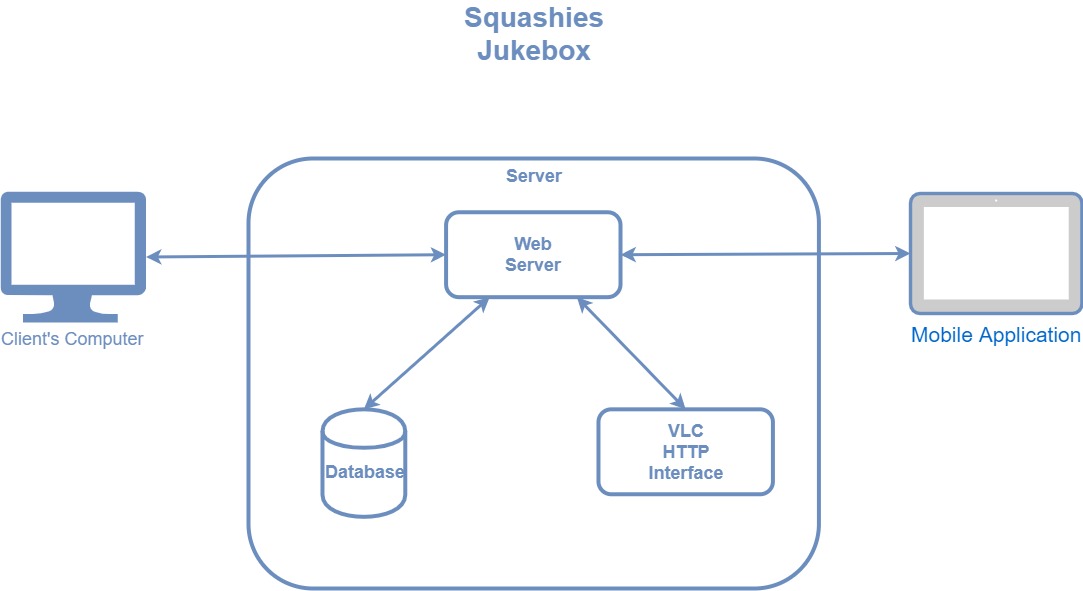
If the design of the app is poor (bad choice of colours, features are disjointed, etc.), then it’s highly unlikely any user would want to use the app which doesn’t help the user as well as the client and their business.

Simple Layout of the Android App

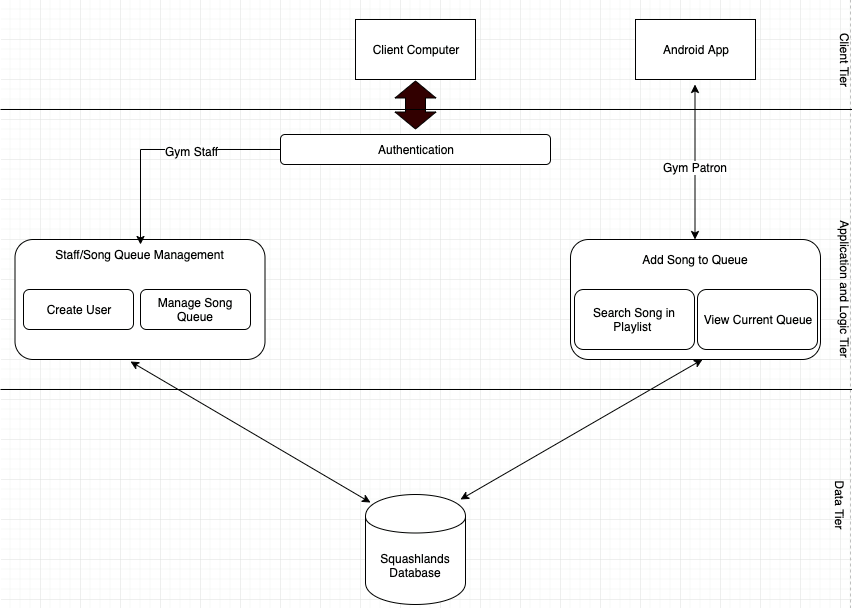
There are three main components for the layout of the app - the song playlist and the current queue each take half the screen while the footer goes across the bottom. The footer will be used as text that will scroll across the screen to give an appealing look to the app. In regard to the colour for the app, we choose between three colours - blue, black and white - based on the Squashland’s logo. The background will be black, the text will be white and headers/other features will be blue. The combination of these colours make the app have more of a professional and modern look and will appeal to users that come across it.



## System Architecture

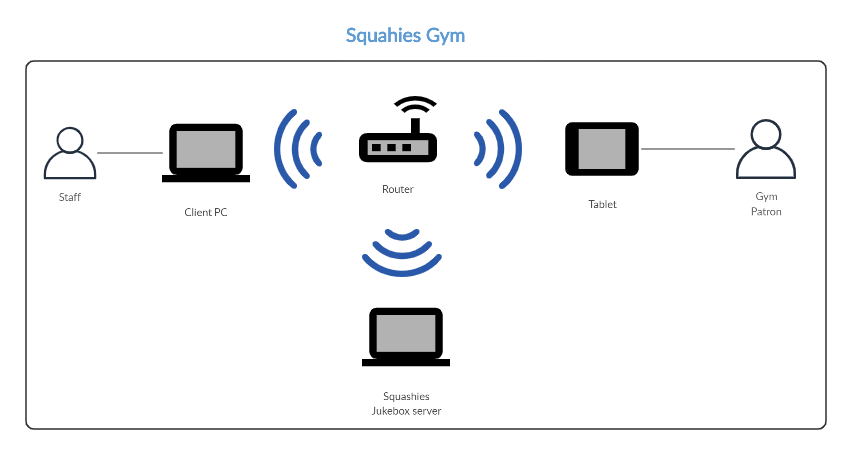


## Software Architecture



## 

## Network and Communication Architecture



# Detailed System Design

## Actors

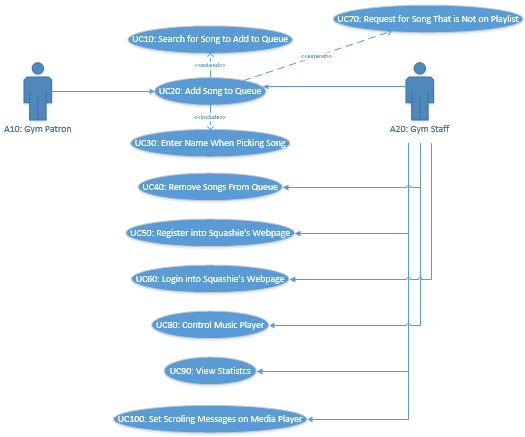
A10: Gym Patron

|  |  |
| --- | --- |
| **Actor** | A10: Gym Patron |
| **Type and Stereotype** | This is a primary abstract actor representing all the gym patrons that will use the Squashland’s Jukebox app. |
| **Description** | These are the users who will be using the app to request songs from a playlist and add them to a queue. |
| **Relationships** | None |
| **Use Case Interaction** | * UC10 * UC20 * UC30 * UC70 |
| **Interface Specifications** | Refer to Screen Designs |
| **Author and History** | Joshua Incollingo |
| **References** | None |

A20: Gym Staff

|  |  |
| --- | --- |
| **Actor** | A20: Gym Staff |
| **Type and Stereotype** | This is a primary abstract actor representing all the gym staff working at Squashlands Gym and Fitness |
| **Description** | The gym staff will be responsible for controlling and managing the song queue as the song requests come in from the patrons. They are also able to add songs from their web page in case not enough songs are being queued. |
| **Relationships** | None |
| **Use Case Interaction** | * UC10 * UC20 * UC30 * UC40 * UC50 * UC60 * UC70 * UC80 * UC90 * UC100 |
| **Interface Specifications** | Refer to Screen Designs |
| **Author and History** | Joshua Incollingo |
| **References** | None |

## Use Case Diagram



## Expanded Use Cases

UC10: Search for Song to Add to Queue

|  |  |
| --- | --- |
| Use Case: | UC10: Search for Song on Playlist |
| Actors: | A10: Gym Patron, A20: Gym Staff |
| Description: | This use case will allow gym patrons to pick the song that they would like to listen to and therefore begin the communication between the app and the web server. |
| Pre-Condition: | If they are a gym staff member they must be logged into the webpage |
| Post-Condition: | Gym patron/staff has found the song they were looking for in the search list. |

Typical Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Clicks on search bar | Nil | 2 | Asks user to enter input search bar |
| 3 | Searches song they would like to listen to | Filters search based on user input | 4 | Shows the list based on the user input on search bar |

Alternative Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Clicks on search bar | Nil | 2 | Asks user to enter input search bar |
| 3 | Searches song they would like to listen to | Filters search based on user input | 4 | Shows an empty list as the input does not match any song from the playlist, which indicates to the user that they may have either misspelt the name of the song or the song doesn’t exist on the playlist |

UC20: Add Song to Queue

|  |  |
| --- | --- |
| Use Case: | UC20: Add Song to Queue |
| Actors: | A10: Gym Patron. A20: Gym Staff |
| Description: | Once the gym patron finds the song they are looking for, they will be able to add it into the queue, by clicking on the “+” (add) button. |
| Pre-Condition: | If they are gym staff, they must be logged onto the webpage |
| Post-Condition: | Gym patron has added a song to the queue before being prompted to enter their name (UC30) |

Typical Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | User clicks on the “+” button adjacent to the song they want to add to the queue | Redirects user to alert dialog | 2 | Alert dialog that requires users to enter their name (see UC30: Enter Name When Picking Song) |

Alternative Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | User tries to click on the song title rather than the “+” button | Nil | 2 | Nothing will happen, indicating to the user that’s not where they need to click to add the song |

UC30: Enter Name When Picking Song

|  |  |
| --- | --- |
| Use Case: | UC30: Enter Name When Picking Song |
| Actors: | A10: Gym Patron |
| Description: | Before the song can be successfully added to the queue, the app will ask the user to enter their name, as part of the statistics which will be used by the gym admin. |
| Pre-Condition: | Gym patron must choose a song from the playlist |
| Post-Condition: | Gym patron has added a song to the queue and their name has been submitted. |

Typical Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Once the user enters their name, they will be required to enter their name, that is valid | System checks to see that it is a valid name being entered (without special characters, numbers) | 2 | Toast message saying that the song the user chose will be added to the queue |

Alternative Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Once the user enters their name, they will be required to enter their name, however it is not valid (numbers or special characters) | System checks to see that it is a valid name being entered (without special characters, numbers) | 2 | Error message saying that the name is not valid |

UC40: Remove Song from Queue

|  |  |
| --- | --- |
| Use Case: | UC40: Remove Song from Queue |
| Actors: | A20: Gym Staff |
| Description: | Once the gym patron finds the song they are looking for, they will be able to add it into the queue, by clicking on the “+” (add) button. |
| Pre-Condition: | They need to be a logged-in staff member |
| Post-Condition: | Gym patron has added a song to the queue before being prompted to enter their name (UC30) |

Typical Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Staff click “Media” tab | Nil | 2 | Takes staff to “Media” page |
| 3 | Staff will click “Remove” next to the song they want to remove from the queue | System will check to make sure the song is not currently playing | 4 | Song of choice is removed |

Alternative Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Staff click “Media” tab | Nil | 2 | Takes staff to “Media” page |
| 3 | Staff will click “Remove” next to the song they want to remove from the queue | System will check to make sure the song is not currently playing | 4 | Song of choice is currently playing and song is not able to be removed |

UC50: Register into Squashie’s Webpage

|  |  |
| --- | --- |
| Use Case: | UC50: Register into Squashie’s Webpage |
| Actors: | A20: Gym Staff |
| Description: | When the gym staff attempt to use the Squashies webpage for the first time, they will need to create an account in order to access the queue and control/manage it accordingly |
| Pre-Condition: | Nil |
| Post-Condition: | The gym staff successfully registered into the Squashlands webpage/system |

Typical Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Staff enter a unique ID and password | Nil | 2 | Asks for validation for the password |
| 3 | Staff re-enters password as part of re-confirmation | System makes sure that the two input match perfectly | 4 | Confirms new registered account for the system |

Alternative Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Staff enter a unique ID and password | Nil | 2 | Asks for validation for the password |
| 3 | Staff enters a different password compared to the first one | System makes sure that the two user inputs match perfectly | 4 | Error message saying the passwords do not match |

UC60: Login into Squashie’s Webpage

|  |  |
| --- | --- |
| Use Case: | UC60: Login into Squashie’s Webpage |
| Actors: | A20: Gym Staff |
| Description: | The gym staff will have to log into the Squashies server in order to access the song queue, |
| Pre-Condition: | Needs to have a registered staff account |
| Post-Condition: | Staff member is logged into the Squashies webpage |

Typical Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Staff enters their username and password | System checks to see if the username and password match the Squashies database | 2 | Directs staff to home page |

Alternative Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Staff enters their username and password that is not valid | System checks to see if the username and password match the Squashies database | 2 | Error message saying that either the username or password is incorrect |

UC70: Request a Song That is Not on Playlist

|  |  |
| --- | --- |
| Use Case: | UC70: Request a Song That is Not on Playlist |
| Actors: | A10: Gym Patron |
| Description: | Gym patrons will be able to request for a song that is not available |
| Pre-Condition: | Nil |
| Post-Condition: | The request has been sent to the Squashies staff and the system |

Typical Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Gym patron clicks on link/button to request a song that’s not on the playlist | Nil | 2 | Redirects patron to request page |
| 3 | Patron enters song name and their email address | Validating email address given by the patron | 4 | Success message that the request has been sent |

Alternative Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Gym patron clicks on link/button to request a song that’s not on the playlist | Nil | 2 | Redirects patron to request page |
| 3 | Patron enters song name and an invalid email address | Validating email address given by the patron | 4 | Error message that the email address is not valid |

UC80: Control Music Player

|  |  |
| --- | --- |
| Use Case: | UC70: Control Media Player |
| Actors: | A20: Gym Staff |
| Description: | The gym staff are able to control the music player whether it’s to play/pause the song, or go back/forward songs |
| Pre-Condition: | They need to be logged in as a staff member |
| Post-Condition: | The staff will be able to pause or play the media player, select the next song or go back to the previous song |

Typical Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Staff plays the “play” button | VLC Play Function | 2 | Media player plays song |
| 3 | Staff plays the “pause” button | VLC Pause Function | 4 | Media player pauses song |
| 4 | Staff plays the “next song” button | VLC Next Song Function | 5 | Media player plays next song |
| 6 | Staff plays the “previous song” button | VLC Previous Song Function | 7 | Media player plays previous song |

UC90: View Statistics

|  |  |
| --- | --- |
| Use Case: | UC90: View Statistics |
| Actors: | A20: Gym Staff |
| Description: | The gym staff are able to access the songs that have been selected, which patrons have chosen what song and the timestamp of that selection and they can select between certain dates, from certain times and whether they want a list view or a count of songs |
| Pre-Condition: | They need to be logged in as a staff member |
| Post-Condition: | The gym staff can access the statistics for song selections |

Typical Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Staff clicks on “Statistics” tab on webpage | Nil | 2 | Directs staff to “Statistics” |
| 3 | Staff goes through the prompts and chooses to see statistics between which dates, times, and whether they want it in list or count form | Creates an SQL Query based on user input and searches through the database | 4 | Displays the statistics in table form on the right-hand side of the page |

Alternative Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Staff clicks on “Statistics” tab on webpage | Nil | 2 | Directs staff to “Statistics” |
| 3 | Staff goes through the prompts and chooses dates that are in the future (therefore not valid) | Creates an SQL Query based on user input and searches through the database | 4 | Error message saying that “No results showing” |

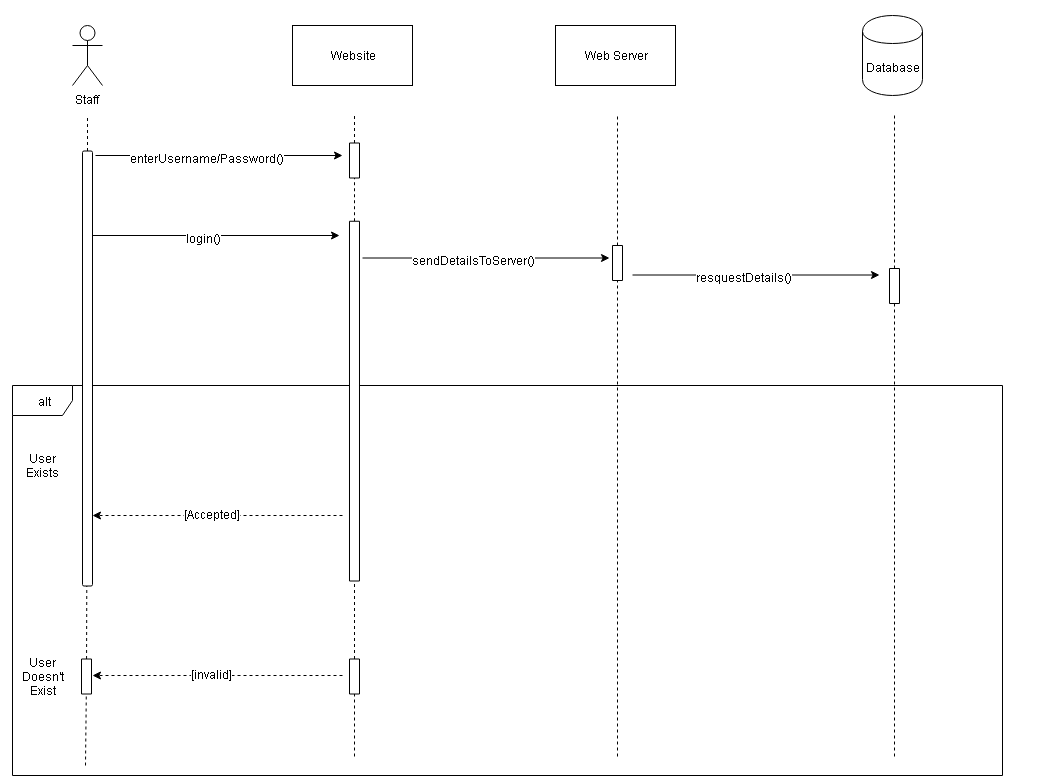
UC100: Set Scrolling Messages on Media Player

|  |  |
| --- | --- |
| Use Case: | UC100: Set Scrolling Messages on Media Player |
| Actors: | A20: Gym Staff |
| Description: | The gym staff will be able to create a message while the media player is running |
| Pre-Condition: | They must be logged into the webpage as a staff member |
| Post-Condition: | A message will scroll across the screen |

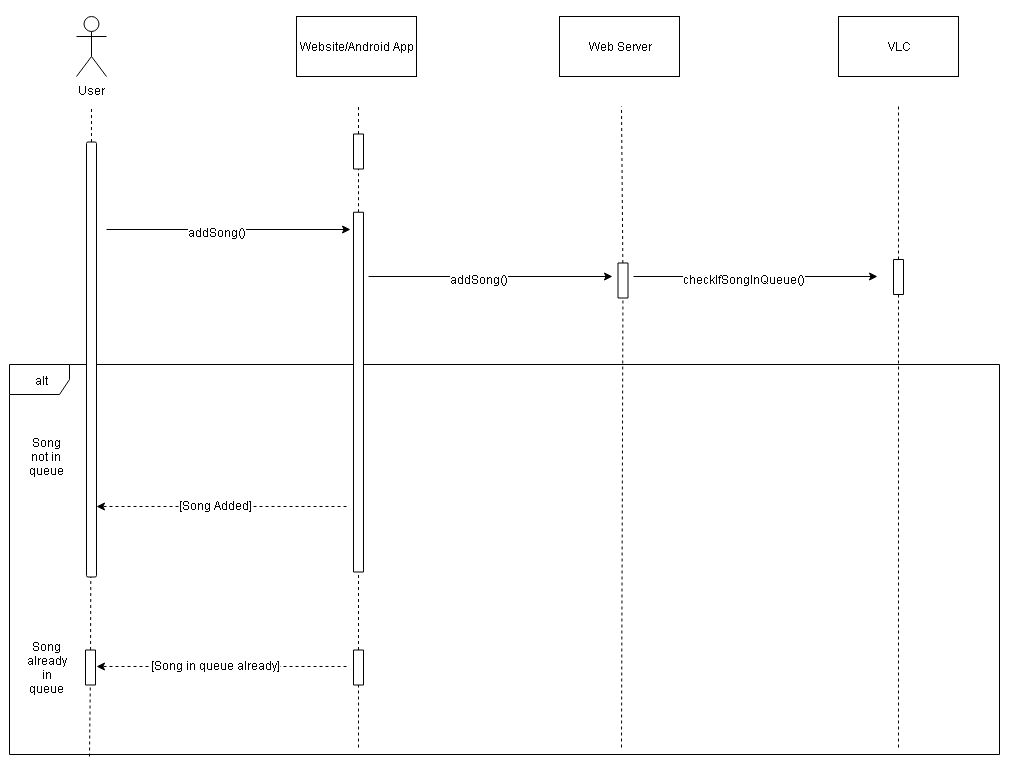
Typical Course of Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Actor Action** | **System Event** | **#** | **System Response** |
| 1 | Staff clicks on “Announcements” tab on webpage | Nil | 2 | Directs staff to “Announcement” tab |
| 3 | Staff enters message that they want scrolling on the media player | Processes the message and sends it through to the media player | 4 | Message appears on media player |

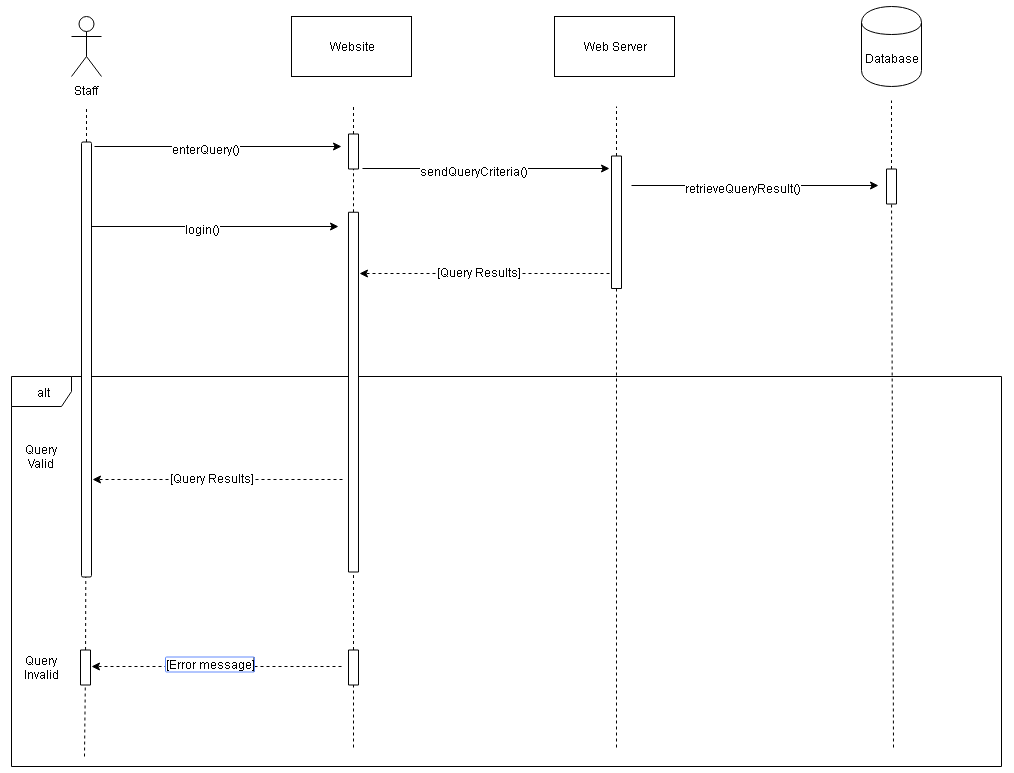
## Sequence Diagrams



**Sequence Diagram 1:** Login for Staff Squashlands Webpage



**Sequence Diagram 2:** Add Song to Queue



**Sequence Diagram 3:** Query Song for Statistics

## Entity Relationship Diagram

staffName

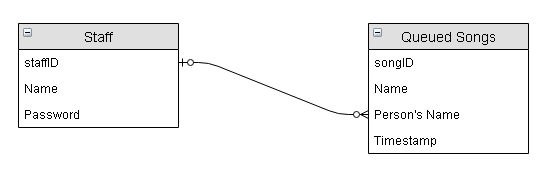
password

songTitle

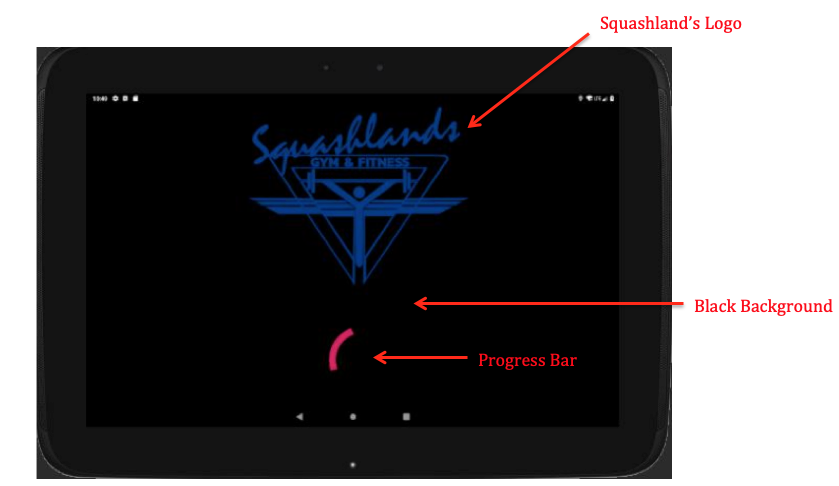
name

timestamp

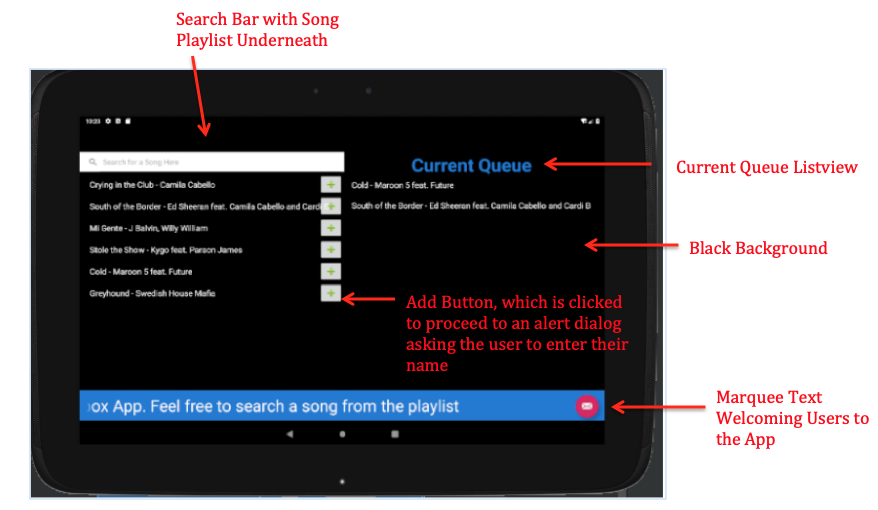
## Database Schema



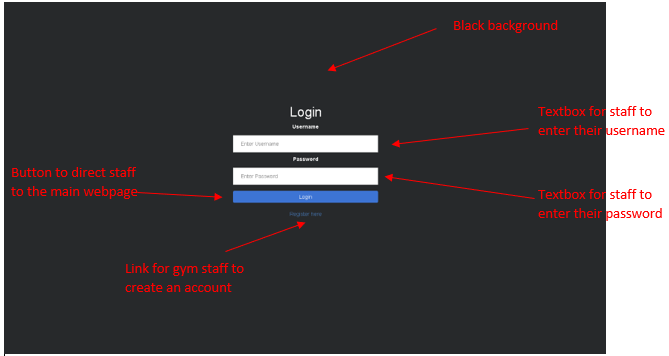
## Screen Designs



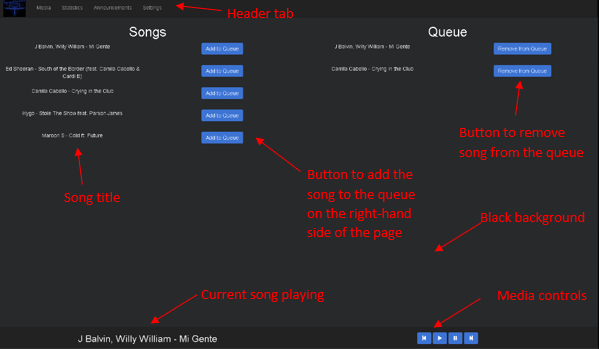
**Screen Design 1:** Android App Introduction Page; on for 3 seconds then moves to the next screen



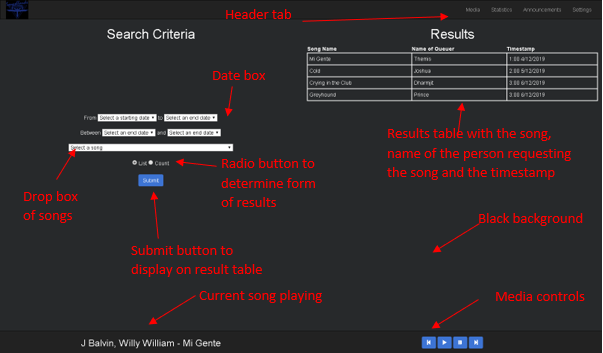
**Screen Design 2:** Android App Main Page



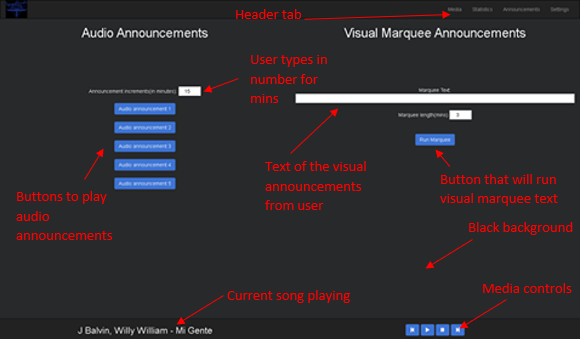
**Screen Design 3:** Login Page for Staff Squashies Webpage



**Screen Design 4:** Main Page for Staff Squashlands Webpage



**Screen Design 5:** Search Query/Statistics Page on Squashlands Webpage



**Screen Design 6:** Announcements – Audio/Visual Marquee

# Test Plan

## Features/Use Cases to be Tested

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Feature/ Use Case to be Tested** | **Types of Testing** | **Pass Fail Criteria** | **Personnel** | **When and Where** | **Training** | **Risk** | **Contingency** |
| Add Song to Queue | Design Requirement Testing | If the song can be added with a valid name entered by user | Team Member: Josh  OR  Client: Angel | Week 11 at Develop Site or Remote | Not needed | Button size may not big enough | Ensure that the buttons are the right size based on the table |
| Remove Song from Queue | Acceptance Testing | If the song can be removed from the queue by staff | Team Member: Themis | Week 11 at Develop Site or Remote | Not needed | Staff try to delete current song from queue which is playing | Error message to alert staff that they can’t do this |
| Register into Squashies Webpage | Acceptance Testing | If the staff member is able to make an account | Team Member: Themis | Week 11 at Develop Site or Remote | Not needed | Pass word not strong enough | Perhaps provide hints of what makes a strong password |
| Login into Squashies Webpage | Acceptance Testing | If the staff member can log into their account | Team Member: Themis | Week 11 at Develop Site or Remote | Not needed | Not the correct pass word | Perhaps a “forget password” link |

## Candidate Test Cases/Test Data

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case/Feature** | Register into Squashies Webpage | | |
| **Interface Ref:** | Screen Design 3: Login Squashlands Staff (this screen design will be very similar to this “Register” one) | | |
| **Test Purpose** | To test if the gym staff are able to successfully create an account and register into the Squashies webpage | | |
| **Expected Results** | See individual test data sets | | |
| **Success/Failure** | **Success** | | |
| **Test Results** | | | |
|  | **Test 1** | **Test 2** | **Test 3** |
| **User ID/Username** | gym101 | josh32 | 20345 |
| **Password** | Hello1 | Yeh021 | joker120 |
| **Re-enter Password** | Hello1 | Ygr032 | joker120 |
| **Expected Results** | Pass | Fail | Pass |
| **Success/Failure** | Success | Success | Success |

## 

|  |  |  |
| --- | --- | --- |
| **Use Case/Feature** | Remove Song from Queue | |
| **Interface Ref:** | Screen Design 4: Main Page of Squashlands Webpage | |
| **Test Purpose** | Staff will be able to remove a song in the queue | |
| **Expected Results** | See individual test data sets | |
| **Success/Failure** | Success | |
| **Test Results** | | |
|  | **Test 1** | **Test 2** |
| **Song Title** | Mi Gente | Crying in the Club |
| **“Remove Button” Clicked** | Yes | Yes |
| **Is Song Currently Playing?** | No | Yes |
| **Expected Results** | Pass | Fail |
| **Success/Failure** | Success | Success |

# Conclusion

To summarise, in the above system analysis and design report we have covered the client and problem statement, identified system requirements, risks and constraints, detailed diagrams with expanded actors and use cases, screen designs and a test plan. By doing all the above tasks, we have developed a strong communication with all parties involved in this project and moving forward we push towards implementing the final system before handing over the project to the client.