by

Thomas Onega, Samuel Brunner

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Thomas Onega — Date: 4/27/20

Samuel Brunner — Date: 4/27/20

Ryan Moore — Date: 4/27/2020

Ryan Moore, Faculty Advisor

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2020

Table of Contents

List of Illustrations

TABLES

FIGURES

Abstract

The average cost to host a DJ for a 3-4 hour set is about $1200; for local venues, bars and events that can be an expensive investment (HireRush). There are other music applications that allow users to add songs to a community queue but fail at providing a solution for events and venues. BeUrDJ is a web-based application that is oriented to allow all music listeners at social gatherings to engage in the song selection. BeUrDJ allows all music listeners to easily input their music choice to the group. The DJ sets up each session where listeners can join, and each session is designed to be highly customizable. DJs can set filters for each session to match the atmosphere of the event and the queue will be updated with recommended songs if there are no songs left in the queue. The queue is then modified based on the upvotes and downvotes for each song. The song with the most amount of upvotes verse downvotes will be first in the queue. This allows all music listeners to easily input their opinion to the song selection. BeUrDJ provides listeners with an interactive music listening experience by giving the power to the listeners.

Problem Statement

Introduction

Music streaming is an undeniably fast-growing market that has impacted the way we listen to music. These music streaming services focus on providing instant music for a single listener. This has resulted in many users enjoying a wide variety of music and enjoying their music on their own. Unfortunately, this has not resulted in social listening behaviors from users. Spotify aims to provide a solution with a new social listening feature, but this feature is primarily targeted towards established friend groups. Our goal will be to create an application that provides an affordable DJ experience for any social event and offers ease of use for users.

Problem

Spotify is the second largest music streaming service in the United States with an average monthly user base of 47.7 million users (Watson, A.). It's a service that supports users on an individual level but struggles to appeal to the listener who enjoys listening to music at local venues. Spotify is currently developing a social listening feature that will allow users to DJ together. But this new feature does not provide a way to filter the songs that are being selected. This can create issues when the application is used in a setting with random input from a large crowd. Amateur DJ's and small venues with little extra funds to afford a premium and need a solution that is affordable and more effective for larger crowds.

Solution

Our BeUrDJ will give the power of the queue to the listener. The purpose behind our web app is to incorporate a shared queue among all users of the web app. Users have the ability to add songs to the host's queue. After the queue is generated with a list of songs, users are able to upvote and downvote songs. Based on the songs upvotes vs downvotes, this creates the queue for the DJ to play. The application will also give the DJ control over a variety of filters that will automate the DJ process for them. This automated filtering system will allow the host to focus less on micromanaging a queue and more on the audience. This proposed solution will create a more friendly music atmosphere for all listeners while still retaining the solo DJ experience.

Overview

The remainder of this report will outline in detail how the project has progressed during the Spring semester. The report includes the following sections: project description/objectives, methodology, user profile, application overview, timeline, and problems encountered.

Discussion

Project Description

BeUrDJ is a responsive web application that utilizes the Spotify API to create an easy to use DJ application. Using Spotify's API to play music the BeUrDJ web application enables listeners to input either an upvote or downvote to the songs on the queued songs. The Spotify API also provides song information that will be used to filter out songs that the DJ does/doesn't want played (based on the DJ's criteria).

Project Objective/Goals

Develop a web application that is compatible with all devices to allow all listeners to use BeUrDJ. BeUrDJ provides an opportunity for all users to input their opinion on the queue of the established listening session. It also allows listeners to input songs into the queue depending on how the DJ sets up the session.

The features the BeUrDJ will include:

- One DJ with the ability to set up a listening session.
  - Music in the queue will be playing from the DJs device
- The session will have features such as:
  - Listeners ability to add songs to the queue
  - Listeners ability to add songs based on song (BPM, Danceability, etc...)
  - The ability for listeners and the DJ to upvote and downvote songs on the playlist that decides the queue.
  - DJ has admin rights (Stop/Start/Skip songs)
  - Recommendation system for maintaining a constant loop of new songs.
  - Filters for the DJ to tweak the genres and types of music that listeners can search.

We had to abandon the goal of implementing a Karaoke feature due to limitations on time spent switching to the MVC framework.

Technical Approach

Network

Amazon Web Services is used to host BeUrDJ. This is a solution that provides many benefits to us, as developers, and to our users. AWS offers a service, Elastic Beanstalk, which makes hosting a .Net Core Web Applications easy, cheap and reliable. AWS offers a solution to deploying updates simple and effective by installing an Amazon Web Service Tool for Visual Studio. Updating our existing Web Application was as easy as selecting the BeUrDJ project and publishing to the designated Elastic Beanstalk environment on our AWS account. Using AWS Elastic Beanstalk, BeUrDJ is able to use a secure HTTPS domain.

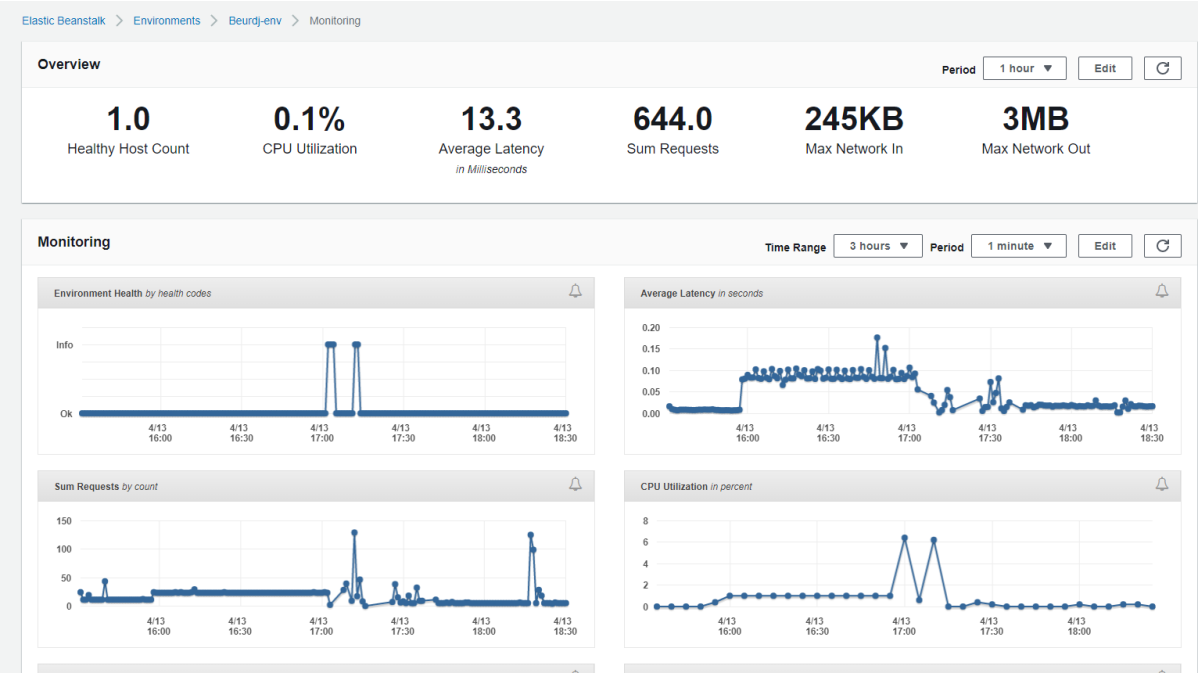
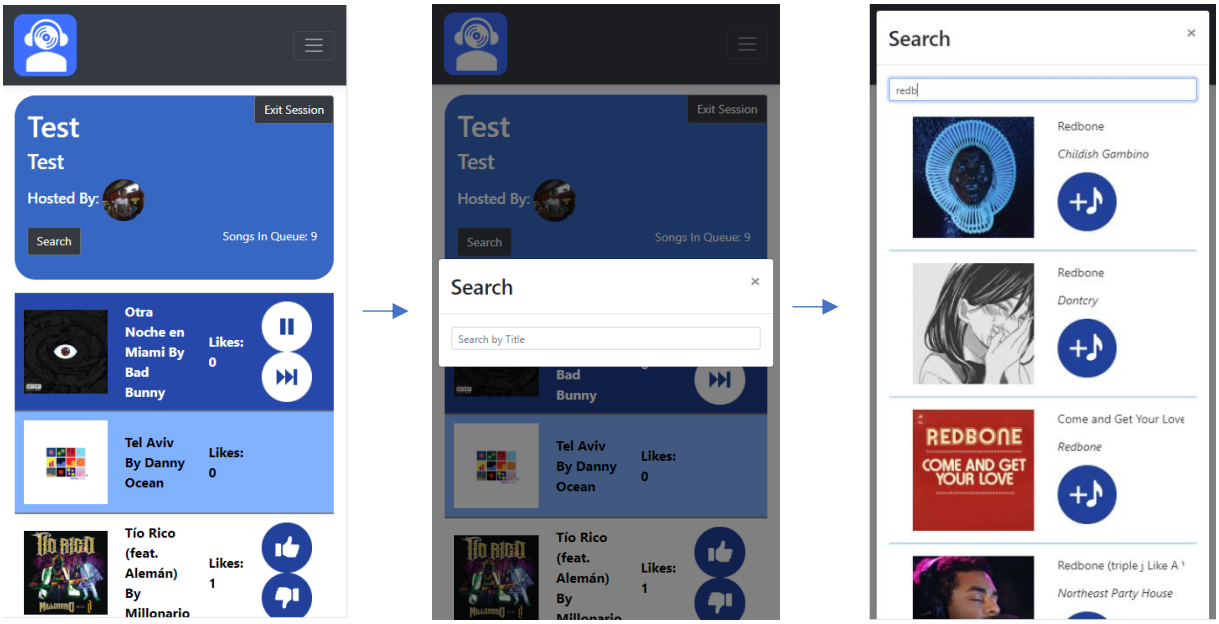


Figure 1. Amazon Web Services. The monitoring page for our Amazon Web Service environment.

Application

Utilizing MVC, we can develop quickly for three different supported platforms: Desktop. Android and iPhone. This will drastically increase our user base and allows for faster deployment and ease of access for our users. The web server backend will provide the application JSON objects for queues, search results, filtering, likes, session information, and DJ information.

Figure 2. User Adding Song to Queue. The front-end process a user will take to add a song to the queue after joining a session (displaying mobile).



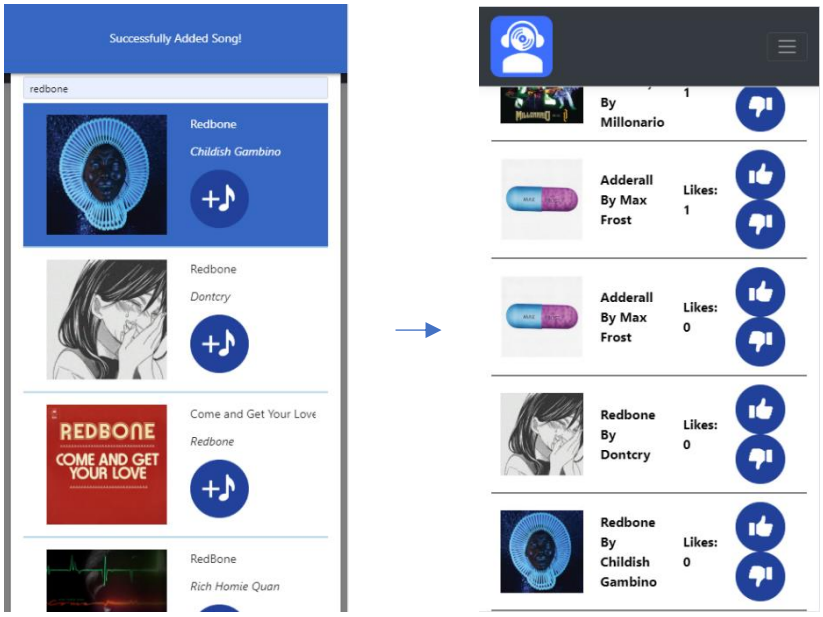


Figure 2. User Adding Song to Queue

Figure 3. User Liking Song. Process a user will take to like a song in the queue (displaying mobile). Liking the song will cause the song to move higher up in the queue. Only one vote is allowed per song per user.

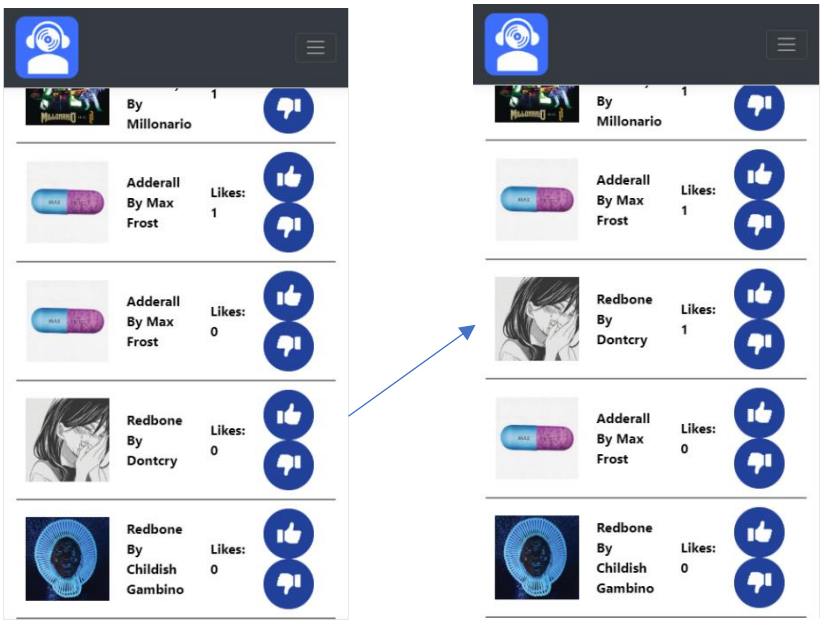


Figure 3. User Liking Song

Database

Using Microsoft SQL, the database will store users, queues and filters. The data will be using table formatting. All database operations will be interfaced using Dapper on our web server backend. Using this method, we will not require multiple logins for the user as well as limiting the number API requests required during a session.

User Profile

Potential Users

- *Venues*
- *Bars*
- *Social Gatherings*
- *Events*
- *Anyone attending these events/places*

Software, Interface, and Related Experience

Users should have experience with desktop and mobile applications, music streaming services, and familiarity with joinable user created lobbies.

Users that will be using the DJ role should have experience with creating/managing a queue and familiarity with controlling settings through mobile/desktop applications.

Experience with similar applications

Technologically adept users may have experience with joining a group session to collaborate with others. The application requires experience with music streaming services and users will

need a Spotify account (Premium or not) in order to use the application. Non-technologically adept users will not have issues using BeUrDJ because we designed the UI to be easy and straight forward to use. This is accomplished with icons and messages throughout BeUrDJ that help users understand what to do as well as keeping the UI as uncluttered as possible.

Task Experience

Users may have experience performing the following tasks using similar applications:

- Adding Songs to a Queue
- Searching for Songs
- Liking/Disliking
- Filtering Songs by Genre, Mood, etc.

Frequency of Use

This will depend on the user group defined in the application.

DJ Role:

Will initially create a session for other users to join. After the session is created, it is up to the DJ how much they wish to monitor the queue and update the filters that have been set for the session.

User Role:

Will initially join the session and add songs to the queue for the DJ to play. The user can like/dislike songs on the queue which will affect a songs position on the queue. Users are intended to frequently check on the queue and add songs throughout the course of the DJ session.

The application's DJ sessions were originally designed to be location based so all the users will need to be in the same location. Due to the announcement of Tech Expo going online, we have

allowed access to all sessions to all users in order to better demo the application. The DJ session is meant for situations where music will be played for small to large sized groups. The frequency will vary between user, but DJs will be using the application more frequently.

Key Project Design Requirements That the Profile Suggests

- ∉ Simple UI with a responsive design for all supported devices
- ∉ Limiting the amount of clicks a new user has to make in order to start using the app.
- ∉ Filtering options easily accessible and that are appropriately named for the DJ to utilize.
- ∉ Recommendation system for users to add songs off of.
    - This will relieve strain on the DJ having to monitor the queue as frequently.

Use Case Diagram

The diagram below, **Figure 4: Use Case Diagram**, displays the use case for BeUrDJ. The diagram shows the two different kinds of users we support, DJ and Listener, and how each feature functions.

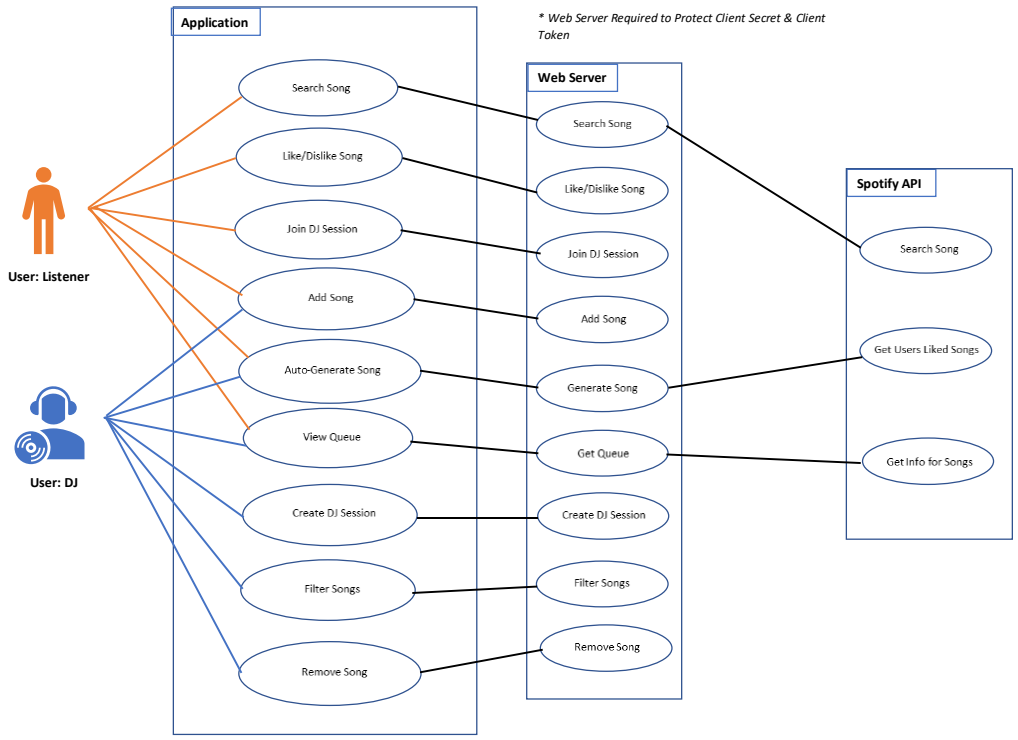


Figure 4. Use Case Diagram

Application Overview

BeUrDJ is built on a simple workflow that encourages ease of access for users. The workflow for the application will be shown and described with figures below.

Figure 5. Home. The home page for BeUrDJ (displaying desktop).

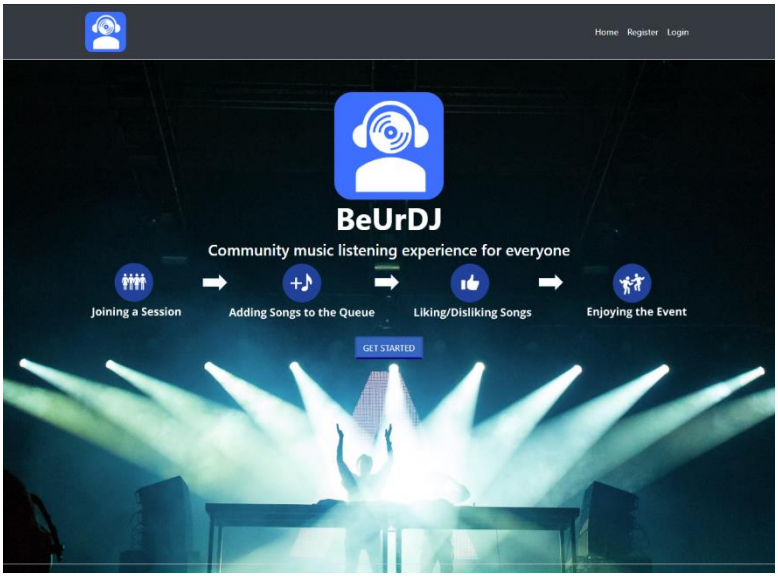


Figure 5. Home

Figure 6. User Register & Login. User creating an account and logging in (displaying desktop).

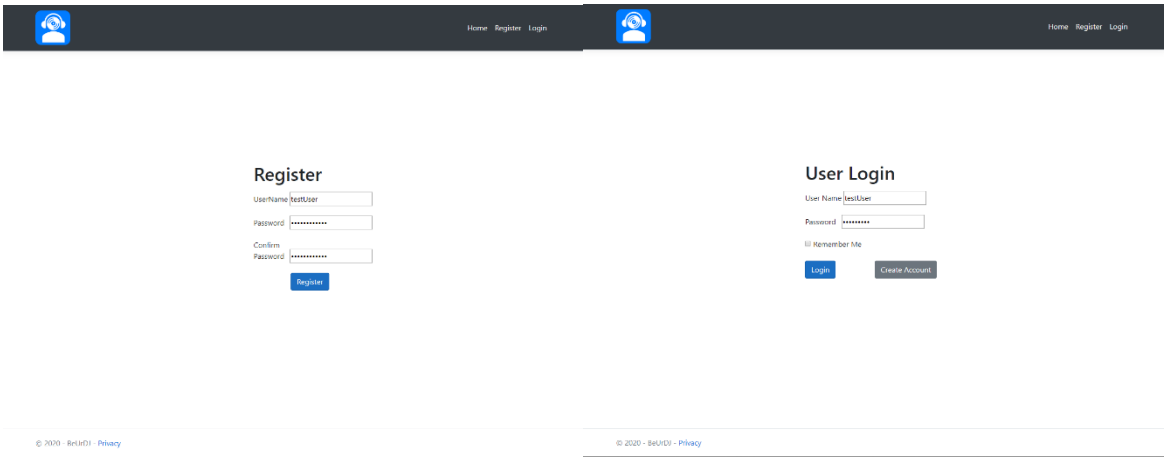


Figure 6. User Register & Login

Figure 7. Creating Session and DJ. User creating a session and a DJ name for the session (displaying desktop).

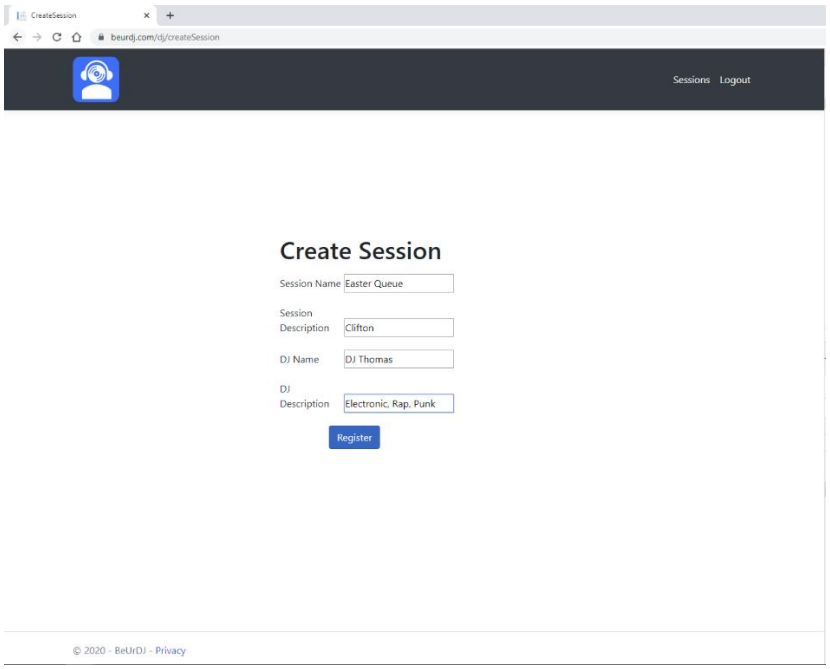


Figure 7. Creating Session and DJ

Figure 8. Queue Startup. Recommended songs will populate the queue when there are no songs in the queue (displaying desktop).

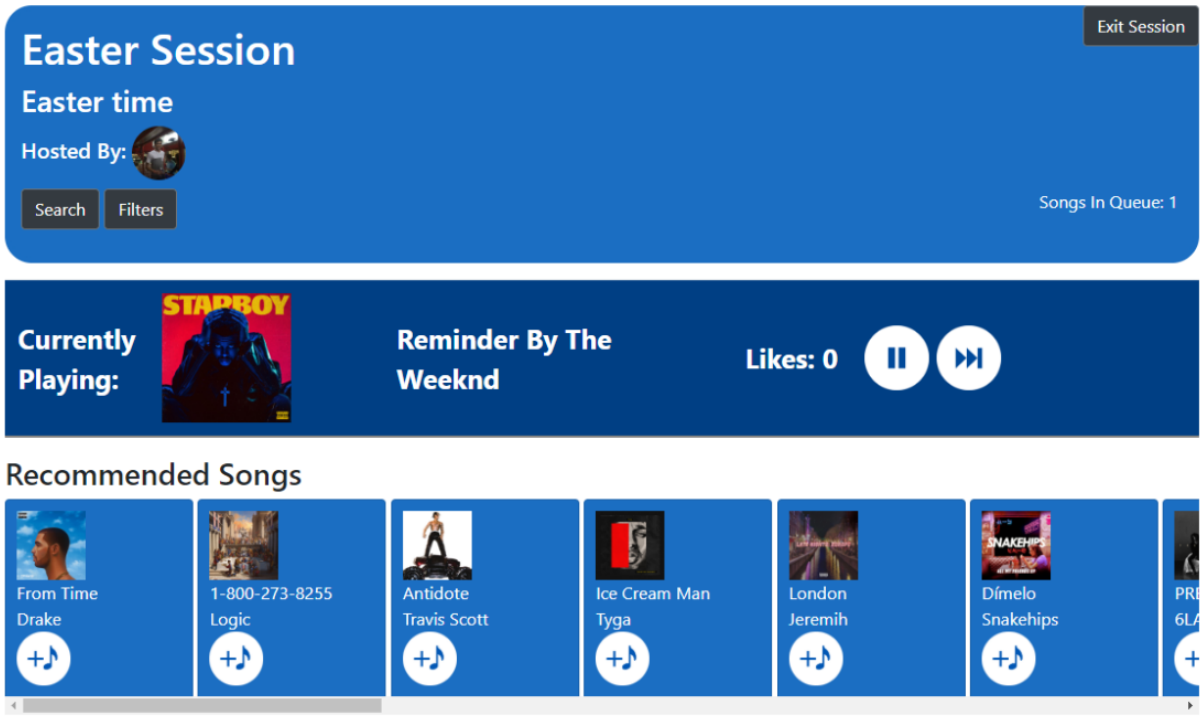


Figure 8. Queue Startup

Figure 9. Filters Effect on Search. Filters clearly displayed for user to view the currently set genre filters (displaying desktop).

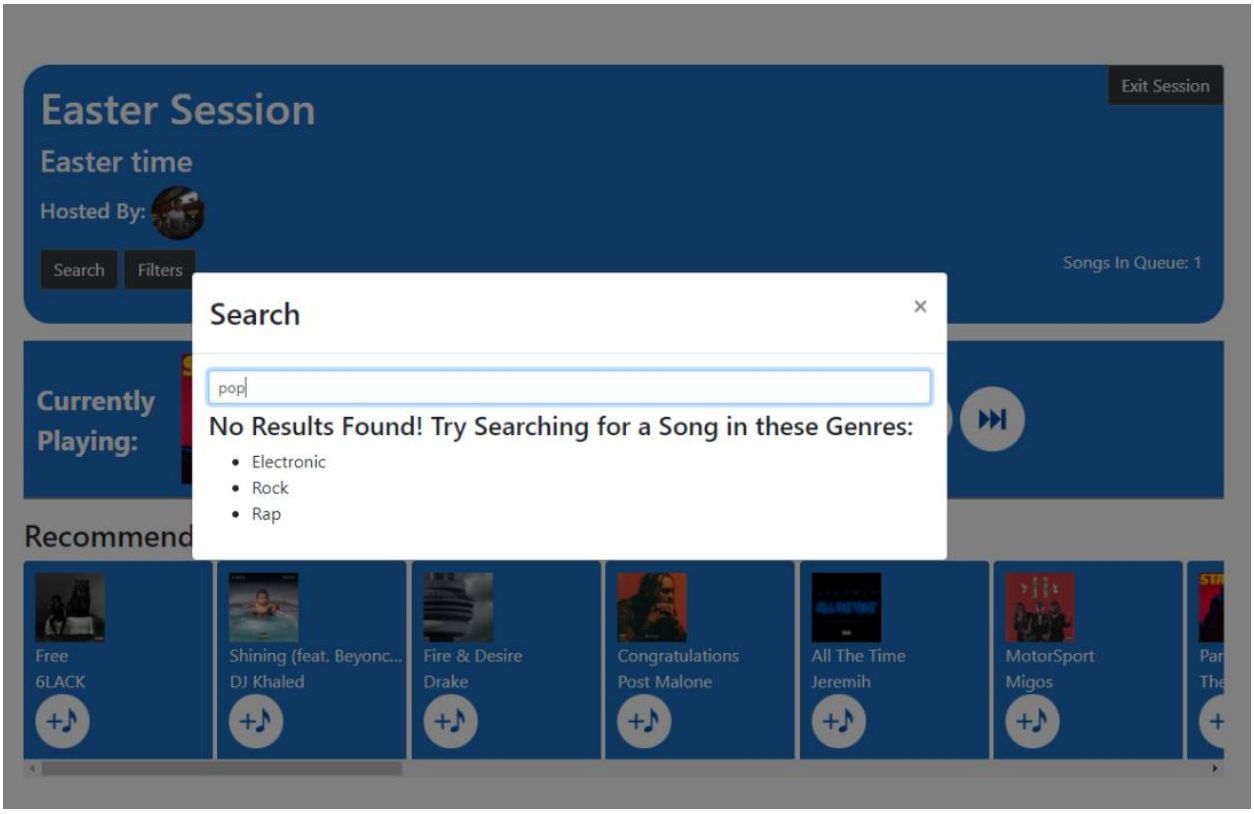


Figure 9. Filters Effect on Search

Testing

For our testing approach we focused on two approaches which are distinguished by frontend and backend testing:

1. Unit testing our application backend workflow before committing any new code to our master branch.
2. Team manual quality assurance (QA) testing took place before each new feature was committed to our master branch to ensure the frontend and backend were functioning properly.

Unit testing was done when a new endpoint for our backend was created. Before committing to our master branch, a developer would run all unit tests in the project to ensure that all tests pass.

The developer also needs to create a new test for any new endpoints that were created. This was done to ensure that, before moving to manual QA, all our backend services were functioning properly. This helped when debugging because the developer will know exactly what test which correlates to the designated functionality of the application are not functioning properly. This also helped conserve time at group manual QA testing because the developer will have already checked for backend functionality issues.

Figure 10. Unit Test. Shows how the tests are displayed after all tests are ran for each code commit:

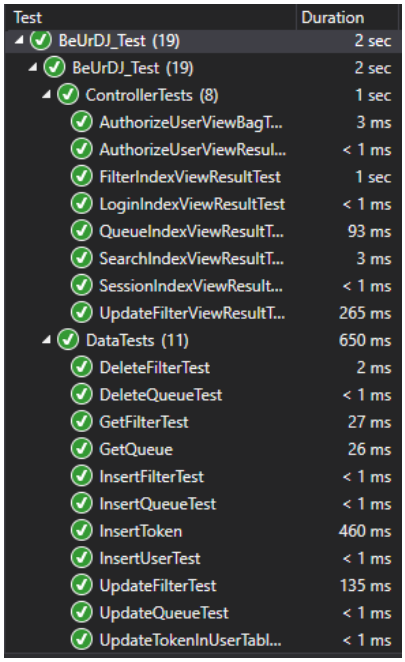


Figure 10. Unit Tests

Team manual QA testing was done before a new feature was added to the project. This entailed:

1. The developer updating a shared excel spreadsheet when a new feature is being worked on and outlining the steps and expected result of the step.

2. All members being present for testing and being able to run the application on their computer.
3. Project manager creates a new sheet recording the date the testing took place and the results of each QA test workflow.
4. The team running through QA test workflows outlined in the spreadsheet and noting any issues they see throughout the workflow.
5. Each time a QA test workflow is completed, the project manager describes the status of the workflow with the following:
    a. Pass: The test workflow was a success for all users.
    b. Fail: The test workflow had issues in one or more steps for one or multiple users.
    c. Void: The test workflow is documented, but the feature is still under active development.
6. If any QA test workflows failed, the issue is documented, and a task card is placed in the defect column on the group scrum board.

This process was done to ensure that all features were looked over by all members of the team. This was intended to provide many eyes on the features being tested and allows the team to have certainty that a feature will not cause any issues when released. Performing testing as a team also ensured that all members of the team are aware of new releases and have an opportunity to ask questions regarding the status of the project.

Figure 11. Example Manual QA Test Case Steps shows an example of the procedure the group will do collectively to test a specific feature.

User - Add Song

Steps	Expected Result
User Login w/ username: admin + password: beurdjtesting	Redirected to 'Session Join' page
User clicks on 'Easter' Session button	User is prompted to enter session code
User enters 'beurdjtesting' into textbox and submits	Redirected to 'Queue' page
User clicks search button	Redirected to 'Search' page
User types a song name into the search bar	Song search results appears below search bar
User clicks ' + ' button to add the song to the queue	Song is added to the queue in the database
User clicks ' + ' button to add another song to the queue	Song is added to the queue in the database
User clicks 'Current Queue' button	Redirected to queue page for the session they joined
Look to see if newly added songs are in the queue	Newly added songs are in the queue

Figure 11. Example Manual QA Test Case Steps

Scope of Testing

The test cases we covered were mainly to test the major features of our application. This entailed testing all possible data calls to ensure database integrity and functionality. While unit testing was used for data call testing and controller endpoints, we were able to test manually the functionality of our site (end-to-end testing).

Figure 12. Test Cases below shows how the unit tests are organized in our test project:

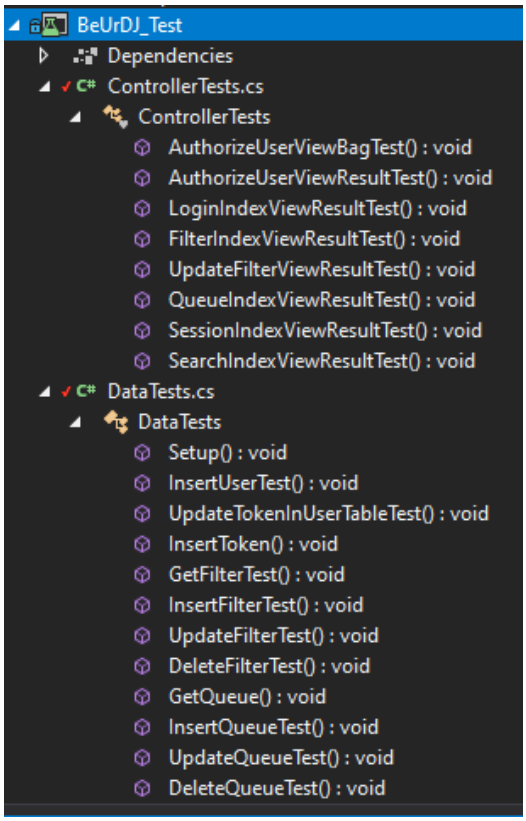


Figure 12. Test Cases

Objectives

Our Test Plan had to accomplish the following objectives:

1. All controller endpoints and repository methods that accessed the database had to have unit tests before they were committed to the master branch
2. During team QA testing, all members of the team have to be present.
3. QA test workflows have to encompass all features for each available role (DJ and User).
4. All failed QA test workflows are to be documented on the shared excel spreadsheet and a task card is to be created in the defect column of the task board.

5. All defect task cards are to be resolved and closed before the IT Expo.
6. All QA test workflows are to be passed before the IT Expo (this includes workflows that have a status of 'Void').
7. All unit tests are to be run, and all must pass before the IT Expo.

Logging Test and Procedures

As described in the **Overview/Methodology** section, logging project tests occur during our team's QA testing sessions. During each test workflow, the project manager is responsible for documenting the status of the workflow. If the workflow fails, the project manager documents a 'Fail' status for that workflow and the team collectively looks for the issue that is occurring. If an issue is discovered, the project manager documents the location of the error and provides any information on the issue that they can (this may include screenshots, exception information, etc.). The project manager then creates a task card under the defect column on the project scrum board. The team QA test meetings occur every Saturday as of 1/18/2019 and are scheduled at a flexible time between 12pm - 7pm depending on the team members' availability.

Figure 13. Example Manual QA Process Logging shows an example of the procedure the group will do collectively to test a specific feature.

QA Results 2/8					
ID	Work Item	Title	Status	Resolved	Issue
1	User Test Case	User - Add Song	Pass		
2	User Test Case	User - Like/Dislike	Pass		
3	User Test Case	User - Shareable Link	Pass		
4	User Test Case	User - Search	Pass		
5	DJ Test Case	DJ - Create Session	Pass		
6	DJ Test Case	DJ - Add Filters	Pass		
7	DJ Test Case	DJ - Queue Playing	Pass		
8	DJ Test Case	DJ - Automated Song	Void		

Figure 13. Example Manual QA Process Logging

Budget

Table 1. Project Budget defines the budget for BeUrDJ. By hosting on AWS, the backend costs were small and affordable. We were able to build BeUrDJ using Visual Studio Community edition which is free for all university students to cut down the final cost. The simulated cost outlines the project would take 3 months to create at 40 hours a week with 2 developers making $30 per hour totaling $14,400. The actual cost of the project was $37 after all hosting fees were incurred and the custom domain was purchased.

Item	Unit, Hours	Unit Price	Line Item Total
Simulated Labor Cost	480	$30.00	$14,400.00
Custom Domain Name	-	$12.00	$12.00
AWS Elastic Beanstalk Web Application environment (t2.micro model)	n/a	$15.00	$15.00
AWS Relational Database Service	n/a	$5.00	$5.00
Route Service(HTTPS Certificate)	n/a	$5.00	$5.00
		Total:	$37.00

Table 1. Project Budget

Gantt Chart

Figure 14: Gantt Chart. Displays our timeline and progress on BeUrDJ.

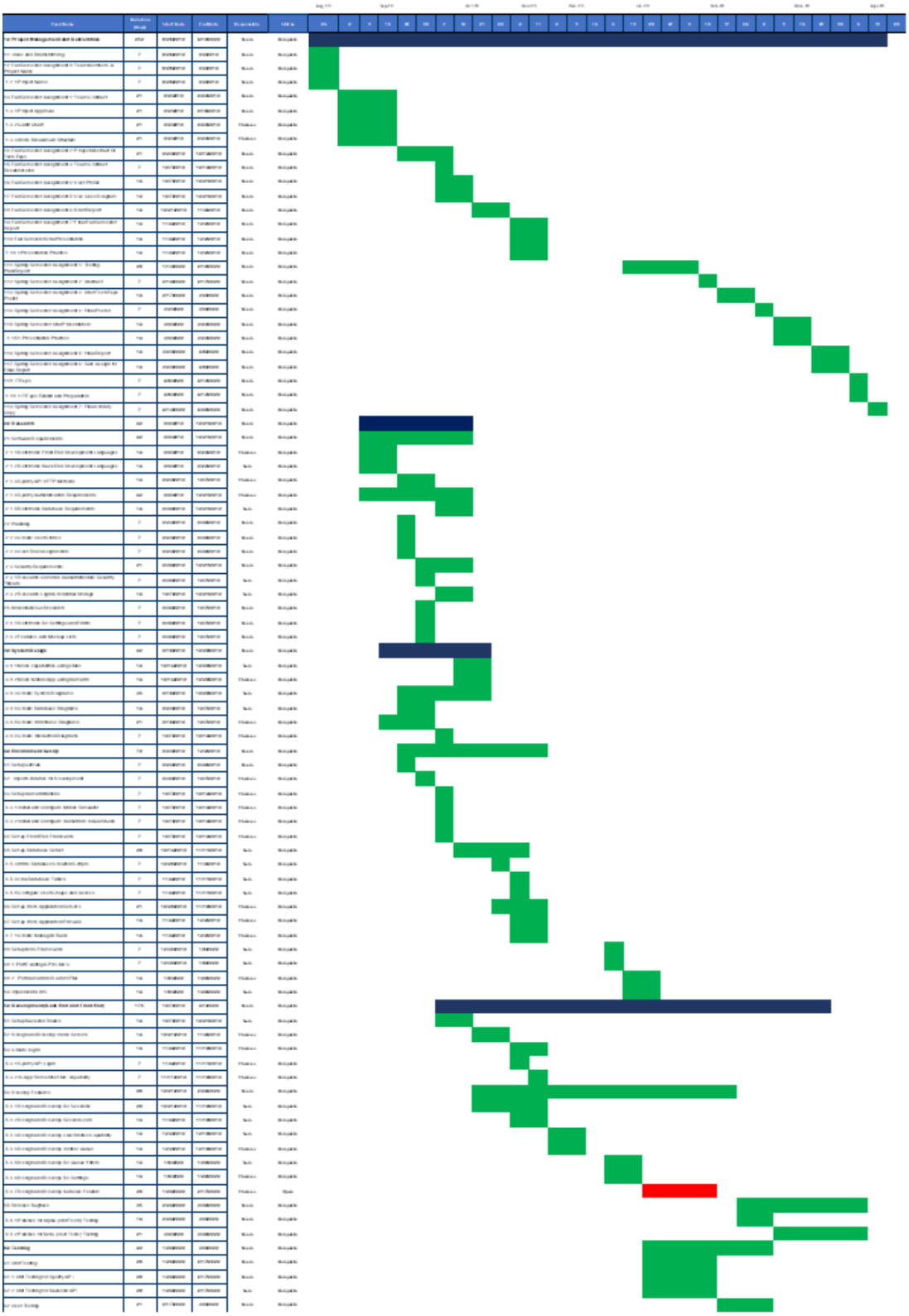


Figure 14. Gantt Chart

Problems Encountered

During the Spring Semester we have encountered a few issues that we have worked together to solve. Some of those issues include the following:

1. **Xamarin & API Framework**
    a. Problem
        i. Xamarin was restricting our ability to work with the Spotify Web SDK (used to create an audio player in the DJ's browser and coordinates playing the next song).
        ii. The only major benefit of using Xamarin is the flexibility of easy porting to other devices.
    b. Solution – Technology shift to MVC, Razor Pages. & Bootstrap.
        i. Provided greater flexibility in the services we can utilize such as modals, the Spotify Web SDK, and fast deployment.
        ii. The API framework is very similar to MVC so our API could be ported easily.
    c. Unresolved – Spotify Web SDK
        i. The Spotify Web SDK is only supported on desktop at the moment due to the technology being in beta. Due to this, we only support desktop for audio playback.
2. **Azure Devops to AWS**
    a. Problem

i. Due to the cost to host one IIS .Net environment we needed a hosting platform that was as reliable at a more effective price point

b. Solution – Amazon Web Services (AWS)

i. Implemented AWS Elastic Beanstalk environment to host a Windows Server that is reliable. By using a t3.micro environment are web application is able to support multiple users while keeping cost less than $10 a month.

3. **HTTPS domain service**

a. Problem

i. Including HTTPS was a challenge due to the limited knowledge behind the technology that allows for secure SSL connections.

b. Solution

i. Many videos and documents were watched and read to understand the technology behind HTTPS. It required a depth of knowledge of AWS and the domain DNS service to correctly use this technology. By creating an Elastic Beanstalk environment with a load balancer, we were able to forward all HTTP request as HTTPS on the designated port.

Future Recommendations

When we first started BeUrDJ, we wanted to reach out to as many users as possible. At first, we chose a cross platform application, Xamarin forms, but had issues with the technology in the

backend of the application. If we had the chance, we would have started BeUrDJ as a web-based application. This could have been avoided if we were able to research a little more deeply to verify what Xamarin is capable of and what it is not capable of. Thomas and I both agreed that the site could use more styling. We wanted to focus on creating a smooth backend performance, so users don't experience downtime. Focusing a lot on the backend resulted in less time on the frontend. One issue we would like to resolve if we had more time is caching more available data on the client-side as we could run into issues with a large user base. I was able to share BeUrDJ with some friends and they let me know that BeUrDJ should try to incorporate a way for all users to listen to the music instead of just the DJ. We both agreed that this could be a fun feature if users are remote and wanted to listen to the same queue at the same time, though that is not BeUrDJ's intended purpose. We want to continue hosting this project but there is a cost associated with the large user base we want using it. If we were able to raise money for hosting, we could host BeUrDJ indefinitely for all users to enjoy!

Conclusion

Fall Semester:

During the last semester both of us have learned quite a bit of how Xamarin Forms work, the technology needed to successfully use a 3rd party API (Spotify), and learning a new server admin technology. Most of the work has taken place with server work, connecting to Spotify's API, and building our own Web API for security and better functionality of the native app. Our Web API is currently running on an Azure App Service that has the capability for our application and Spotify's API to make HTTP requests. This took quite a bit of time and background work to get fully working. Once we had this setup, the Web API that is ran on our App Service Plan is enabled for Continuous Integration and Continuous Deployment. This will help us immensely in the coming months of development due to the easiness of pushing a git commit and having our backend server automatically build and deploy to the designated deployment slot. Now, we have been able to setup simple HTTP methods for Spotify's API from our Web API. This will allow a secure connection to Spotify's API making our app safe from potential hackers. In conclusion, our Web API has the capability of making HTTP request to Spotify's API to obtain the needed data to support the native UI features we have laid out. Now that the initial work for the Web API and Azure technology is set up, the next few months will be focused on getting complete functionality to the user interface.

Spring Semester:

The shift in technology was a very tough decision to make for our team. We think it was the right decision because our application gained a lot of new functionality that was more difficult or

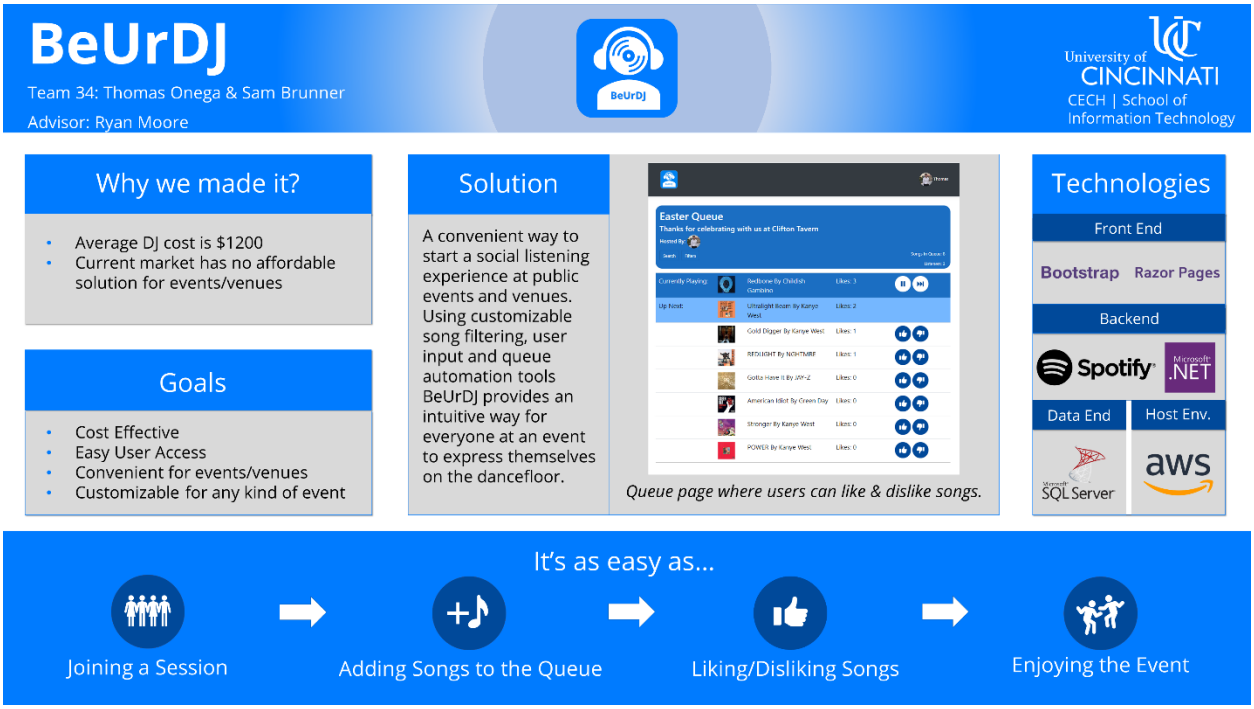
impossible with Xamarin. Changing the host platform to AWS also required more time than we anticipated. This required a great depth of learning a to understand how to use AWS effectively. BeUrDJ has been a great learning experience; learning how to decide what technology to use, fitting in unplanned in a limited timeframe and working as a team were learning opportunities not presented in a normal classroom environment. In the end, we have completed all our functional requirements and have a live site at beurdj.com.

References

- Watson, A. (2019, August 9). Top U.S. music streaming services by users 2018. Retrieved November 10, 2019, from https://www.statista.com/statistics/798125/most-popular-us-music-streaming-services-ranked-by-audience/.

- HireRush. (n.d.). Retrieved November 10, 2019, from https://www.hirerush.com/cost-guides/dj.

Appendix A


BeUrDJ
Team 34: Thomas Onega & Sam Brunner
Advisor: Ryan Moore
BeUrDJ
University of CINCINNATI
CECH | School of Information Technology
Why we made it?
Average DJ cost is $1200
Current market has no affordable solution for events/venues
Goals
Cost Effective
Easy User Access
Convenient for events/venues
Customizable for any kind of event
Solution
A convenient way to start a social listening experience at public events and venues. Using customizable song filtering, user input and queue automation tools BeUrDJ provides an intuitive way for everyone at an event to express themselves on the dancefloor.
Easter Queue
Thanks for celebrating with us at Clifton Tavern
Queue page where users can like & dislike songs.
Technologies
Front End
Bootstrap
Razor Pages
Backend
Spotify
Microsoft .NET
Data End
Host Env.
Microsoft SQL Server
aws
It's as easy as...
Joining a Session
Adding Songs to the Queue
Liking/Disliking Songs
Enjoying the Event