HU Project 2: MusiChat Social Media Website

Quintin Hurrell

ISIT Department, Harrisburg University

ISIT: Project 2

Professor Lateef

April 5, 2023

Project 1

Introduction

A graduation project is a big endeavor, one that should not be taken lightly. A well designed project can lead a student to meeting contacts, getting job offers, or even just getting freelance work. After heavy consideration on what I should do for my Harrisburg University project, I chose a Social Media website. Now, on the surface, this may sound very generic (but there is a catch). My project will be on creating "MusiChat", the Social Media solution for musicians. This platform will specialize in the communication/networking for musicians all around the world. There are plenty of online music forums out there, but none of them hold up to the standards of larger social media platforms. This reality allowed me to aspire to design this one-of-a kind website.

As a jumping off point, I wanted to take inspiration from the popular social media sites. This opportunity allowed me to pay very close attention to the aspects that I liked, but also didn't. I found myself enjoying the design of "Reddit" the most. The anonymous design would work well for the music industry, but it's still important to connect your posts to some sort of account. This top-level design option will have a few benefits that will work for the music industry. First of all, it will allow for famous people to use the platform, but still choose to keep their identity hidden. On the flipside, this would allow for up and coming musicians to share their work to the same audience as someone who already has a following. So, this design choice allows for a very customizable MusiChat experience.

My goal with MusiChat is to bring the music community together, both physically and digitally. I want to create a site that would've helped me as a young and learning musician. My

goal is to allow musicians to network, share content, react to content, and to gain a following. I believe that having this hub for music fellowship will truly take the music community to a new level.

Research Design

Explanation

For my graduation project at Harrisburg University, I have decided that I will create a social media website for musicians. I chose this task because out of all of the musician websites that exist, very few of them are user friendly. Most of these sites will take a very unappealing HTML forum approach, which I plan to stay away from. My goal is to create a site where all musicians can come together, share ideas, and create content. This may sound very similar to these forum websites, but what they lack is the ability to be able to interact with posts in a modern way. I plan to take inspiration from sites like Reddit that include voting systems. I think that post ranking is extremely important when it comes to user interactions. These elements are just a few things that I would like to implement into my site.

Methodology

There are a few technologies that I plan to use in my implementation. Throughout my time so far at HU, I have been working with the programming language Python. Now, as a base language, Python isn't necessarily the most convenient language for web development, but it can be. There are two main web frameworks that are popular within Python (Django, and Flask). Out of these two frameworks, I decided to go with Flask for my website. Through my research, I

think that Django is more in-depth, but Flask is more convenient for small project websites. The main reason I chose flask is because it supports the Jinja syntax. Jinja is programming syntax that allows you to write pythonic code within an HTML file. Flask's use of Jinja and easy syntax will allow me to implement a good website in a reasonable amount of time.

Next, I spent a lot of time learning Python and HTML, but I haven't dug into CSS or Javascript yet. I was researching how to implement CSS and Javascript without actually having to learn the languages. Throughout my studies I found an organization called "Bootstrap". Bootstrap is an organization that creates open source CSS and Javascript code that can be dropped into projects to easily bring life to them. For my project, I want to use the template feature of Bootstrap. For example, if I create a text form to post, it would look like generic HTML. But, if I were to put a Bootstrap template in my HTML, it would be given a sleek and modern look. These templates also allow for easy size, color, and style changes. So, you can see why I would want to use this technology to my advantage in this project.

Lastly, I need a place to store all of the user data. In theory, I could host a server on some sort of physical computer, but that is such a hassle. Not only that, but it probably isn't a good idea to store all of that data on a disk that could fail. So, the only reasonable alternative is using a database. I've done some research on databases, mostly because they are what I know the least about. I learned that using SQL tables would be the most beneficial implementation for my situation. That being said, it is not that easy to just say "I choose to use SQL tables". There are actually many Python libraries that allow you to pass and take data from an SQL database. A lot of these libraries are really in-depth, but I have chosen to work with SQLite, due to its simple nature. All I really need to pass to the database are files that would be linked to an account, so

the tables don't need to be super complex. So, this SQL implementation would be a good mix between simplicity and efficiency for my project implementation.

Implementation

Now I'd like to discuss how I plan to use these technologies in my implementation. First off, my main Python file will act as the server for my site. The Flask library will be passing and receiving data as needed in the Python file. This library allows for easy site route creation, while not compromising on functionality. The form information will be passed through the HTML files, into Flask/Python, and then stored in the database.

As of right now, I plan to create 5 routes using Flask, all pages doing a different task. Each of these routes will have the personalized Bootstrap template that I implement/alter. This implementation will allow my site to be far more appealing than that of a basic HTML forum.

Lastly, my database will be the most intricate part of my website. As of right now, I want to have tables that link text, video, and audio to whatever the user is named. This table will then be connected to the post table, which will be linked to a number (in order of the users posts). The significance of this is that each post will also be its own entity. This allows us to create another table of comments, that would be linked to the post table. I want to allow everything to be its own entity to create full malleability of the data. I believe that these implementation decisions will allow for an extremely functional and efficient social media site.

Summary

Through using the 4 main technologies (Python, Flask, SQL, and Bootstrap), I will be able to create an efficient site for the administrator and user. The Python programming language will act as my server, allowing me to perform calculations and pass variables. The Flask framework will allow me to create routes and pass variables from HTML to Python. SQL will

allow me to organize my data and create tables that can be hosted on a cloud server. And Bootstrap will allow me to give my site a sleek look, without all of the extra added time. I am super excited to start working with these technologies, and am even more excited to start working on my project.

Literature Review

For my graduation project at Harrisburg University I decided that I am going to make a social media platform that is designed specifically for musicians. This idea has been used many times with websites like “Gear Gods”, “Music Banter”, and “Music Corner” but has never been executed in the correct way. Most of these websites are basic forums that allow for anybody to say anything, with little moderation. My goal is to create a website for musicians that allows them to have a safe and welcoming environment in which they can discuss their interests. My site would have key features such as: liking, posting text, posting audio, posting video, post searching, and temporary accounts. This is a very large undertaking as is, so I plan to implement only the essential features as an HU student, and continue development in my professional years. Now, all of this seems great, but actions speak louder than words. As someone who knew next to nothing about web development before researching, I had a lot of work to do. My literature review will discuss how each of my sources will affect my website and give me the knowledge to develop the site that I want.

To start off, I needed to learn how to even start printing HTML to my browser. Through my research, I chose the framework “Flask” to create my website using the Python Programming language. There were a few sources that I pulled about Flask that I have used, and will be using in the future. For starters, *Gaspar, Daniel, and Jack Stouffer (2018)* discuss Flask from a corporate point of view. This is interesting because when you perform a basic google search for

Flask, most people are doing small projects for school assignments. Their deep dive into using Flask as a business tool is a much needed breath of fresh air. I plan to deploy my app to the public, so the help from this source will allow me to prepare for the scenario that my app gets a large user base. This is contrary to the analysis done by *Grinberg (2018)*. This source is more aimed towards the single developer app that I will be creating. Although these two sources do not necessarily focus on the same things, that is not a bad thing. They will both provide a unique point of view on the use of Flask, along with their specific tips and tricks. I do think that the work by *Grinberg (2018)* will be more useful, just because it is more to my scale of development.

Flask has been a very important aspect and asset to this research so far. To use Flask, you must also know some basic HTML. This was something that I learned from a few Youtube videos and eventually got a hold of pretty quickly. On the other hand, CSS and Javascript have given me a little more trouble overall. After reading some online forums, I learned about a set of CSS plug-ins called “Bootstrap”. I soon realized that the implementation of Bootstrap into my project would be extremely useful. The first source that I looked over was some work done by *Spurlock (2013)*. His work outlined the basic yet important aspects of Bootstrap. For example, an explanation of template implementation. This book is more of a tutorial and explanation than anything, which was nice for first starting out. The nice thing about Bootstrap is that the code is already written, so it is entirely up to your implementation for its uses, and there are many. I drew from *Krause*’s (2016) book as well, but his work is very similar to that of *Spurlock (2013)*. These sources are just a very basic level of what is possible with Bootstrap, but I plan to learn a lot of the details during the implementation phase itself.

Websites get a lot of traffic, more than some can manage at times. All of this traffic will accumulate data over time, but where is that data stored? I had this exact same question when I started looking in web development. Interestingly enough, all of this data is either stored on a physical disc or a database. Now, technically I could store user data on a physical disc, since my user base would be really small, but databases are the smart solution to this problem. I did some digging on this topic, and found a few good sources that will be very useful to me. I learned that I will use SQL as my database interface, but am not entirely sure how I want to interact with that database in Python yet. *Welling, Luke, and Laura Thomson (2003)* wrote a book on web database and web backend connection. The source is somewhat useful, but not entirely. I plan to use it mostly for the SQL explanations, along with some of the organization tips. But the fact that this source is written without any web framework in mind, makes it a little less ideal. To explain this further, Flask allows you to write Pythonic code to manage your backend server, while still using HTML. This source is not conducive to that form of web development. The bright side is that my SQL database will not be super involved, so I don't have to worry about it too much. I can always find more sources and add them to my Bibliography if needed.

After learning all of the tools necessary for creating a site, I needed to start actually considering how to make the UI. Almost all of us use some sort of social media on a daily basis, and some sort of eye-catching, yet easy to use UI is very important. I made sure to take from multiple sources for this topic, as it is probably the most important one mentioned in this paper. To start off, *Zarrella (2009)* describes how large companies market and use social media to their advantage. Although this is irrelevant to my current situation as a developer, if my app grew, it could be very useful. Reading *Zarrella*'s (2009) work can also help me to plan certain UI and functionality features towards a large-scale usage. One of these features could be something like

a “Professional Account”, much like Instagram/Facebook. This source differs a lot from the work of *Evans (2010)*. Her work focuses on using social media to engage users and sell a product, but on a different scale than *Zarrella (2009). Evans' (2010)* work will be the most useful to me in the short run. The basic user engagement is what I want to focus on for this project. For example, I plan to implement some sort of liking system, maybe even ranking. I want my users to gain some sort of satisfaction from using my site. The issue with ranking is that users can have negative feelings if their ranking isn’t good, this is less likely with the basic liking system. This instant feedback will make the user want to return and continue using the site, which will increase traffic and the site’s user base.

Lastly, I want to talk about data storage and the ethical nature of storing somebody’s information. I am sure you have seen this topic pop up in headlines consistently, and that is one of the main reasons that I want to take this into heavy consideration during the planning phase. *Verma (2016)* wrote one of the most interesting sources that I could find for this project. The author discusses the storage and management of all the data that would get processed through a social media website. This data includes text, audio, video, and user traffic. Although the source does not speak on the ethics of the situation, it gives a full rundown to social media data storage. For my ethical knowledge, I chose to look towards *Kosinski*’s *(2015)* work on the ethical nature of social media sites. This author’s work is specific to Facebook, but really could be applied to any social media situation. This topic is actually really hard to find scholarly information on, so a lot of my decision making will be made through common sense and critical thinking.

In conclusion, creating a website is no simple undertaking, so gathering as much information as possible is key. As you can tell from my paper, I tried to get sources that include as many angles on the topic as I possibly could. These different takes and perspectives can allow

me to make decisions on my implementation that can work both now, and in the future. After collecting all of this information, I now feel that I am ready to start planning out the creation of this project.

Project 2

Introduction

This half of the essay will cover the implementation phase of my graduation project. As of writing this paper, I have a music social media website where users can post, like, follow, create accounts, edit accounts, and delete posts. All of these features are crucial parts of any social media platform, but are especially important for my vision of the "MusiChat" social media website.

My goal with creating this website is that musicians (like myself) would have a place to network and find work, all in one place. As a musician, a platform like this has never existed, and that very fact is what inspired me to make it. So, not only did I have to learn alot about web development and databases to create this website, but I needed to study the music scene and its needs as well.

For the prototype of this implementation, I used "Flask" in Python to create the website. But, for my full implementation, I chose to use "Django" due to its enhanced database capabilities. Learning these technologies will surely help me advance in my academic and professional lives. Now, I would like to break my project experience down into multiple sections, each explaining a key aspect of the project.

Technologies Used

Django and Flask

In Project 1, my prototype was created with the Flask Python module. This module is extremely useful and user friendly, but I quickly discovered some issues with it that would be problematic. As I started to implement data flow, I realized that Flask did not have any built-in

database content, which meant that I would have to do everything myself. I attempted to use the Firebase database that I had set up from my XCode endeavors (more on this later), but could not get this to work correctly. So, I then started to watch educational videos on the Django module, which fit my project's needs much better. Django has its own built-in database functions that allow for easy data transfer from HTML forms to your server. This became the base foundation for my project.

Bootstrap

For CSS and Javascript, I stuck with using Bootstrap templates, just like in my prototype. Bootstrap creates CSS and Javascript files that can be placed in your project, or can also be easily customized to fit a specific look. These templates fit in with some of the FreeCodeCamp CSS templates that were also used in implementation.

MySQL/SQLite

I used MySQL for all of my account credential management. This would allow me to go into "SQLiteStudio" and manually edit all of my data, as well as organize it as needed. My database was set up so that you had a password, username, first name, last name, and an email. This data storage was more than enough for my project. The posts were not stored in SQL, but instead are stored locally on the host's computer. This can be easily changed if I were to deploy this website. To do this, I would just need to link a Database, instead of listing the file location in the function variable field.

Python

Django and Flask are packages that are written in Python, so my entire server side of "MusiChat" is written in the Python coding language. This is what HU has shown me to be the most comfortable with, and is a great language for website creation.

Implementation Problems

Before I get into the details of my project, I'd like to discuss some problems that I had with the creation of "MusiChat". Over winter break, I researched XCode, as well as writing "Swift" code. When the Spring Semester started, I actually began implementing my project in XCode. The project was going well for a couple of weeks, until it eventually made a turn for the worst. About 4 weeks into my project, I updated my Mac, which also happens to update XCode. To my surprise this update actually made my code completely outdated and unusable, which set me very far behind. After hours of bug fixing, I couldn't get my code to run; so that is when I started developing the website in DJango through online tutorials.

Although this setback was significant, I learned a lot of lessons that I can use in the future. For starters, I need to make sure that my computers are always up-to-date, because that was irresponsible on my part. Proper maintenance of my technology is something that I shouldn't have ignored, as it would've saved me a lot of extra time. Secondly, having something like this occur has reminded me that I am not invincible. Ever since this occurrence, I have been making sure to save my work every couple of minutes, as well as create cloud backups consistently. I cannot risk losing that much progress on something ever again.

Data Flow

This project (like many social media platforms) has many different types of data flow avenues. To start off, the log-in page stores credentials in an SQLite database that is built into Django. This specific function of Django is specifically designed for log-in pages. Using Django to manage my SQL was useful as it allowed me to keep all of my data separated.

Next, my posts and captions are passed from the given HTML document into the Python server. The server then gives them an ordered name, and then stores them locally in my machine.

I chose to store the posts locally due to pictures files taking up far more space than text; This could easily be changed if I ever deployed the website. If I were to deploy, I would create a Firebase database, and send the given files there instead.

Lastly, post interaction (ie. likes) is stored as a tally that is directly connected to every post. This is another feature that would need to be connected to Firebase during deployment. Storing these things locally makes more sense while I am developing on my local server.

Use Cases

Find Musicians

- User logs into website
- User posts about their available position
- Someone looking for work can search their feed for the user's post
- User finds a suitable candidate based upon listing interactions
- User contacts the candidate by their listed profile information

Post Audio Work Offers

- User logs into website
- A job listing is made via the post feature
- Candidates can reach out the user by looking at their contact information
- The original user and the job candidates can follow each other to stay in contact

Follow Artists

- A fan logs into the website
- Fan searches for the desired artist via the chatbox
- The user can then search for the desire account search result
- Fan will then navigate to that artist's profile page
- Fan can then interact with the follow button, which will allow all of that artist's post to appear in their timeline

Discover New Artists

- User logs into the website
- Hashtags can be used to find specific genres, vibes, cultures
- User then searches through hashtag results
- User can then pick whatever post interests them the most

Network With Fellow Artists

- User logs into website
- User can search for know musicians, or discover unknown ones
- Musicians can add each other's information in their bios
- Musicians can create collaborative posts *(future feature)*

What About The Future?

Although MusiChat contains all of the basic/main features of a social media website, I don't want to stop there. There are plenty of other features that I would like to implement into MusiChat after College.

Chatbox

Most importantly, I would like to create a "Facebook Messenger" type of addition to MusiChat. As it stands, a MusiChat user can go to an artist's profile and contact them via listed information; but, I would like to make this into a feature that is accessible in-app. A feature like this would allow a much more safe and moderated environment for new professionals to interact. This type of feature would also include file sharing, as sending music/video privately would be an important part of the app's community.

Notifications

Notifications are important to any app, and is currently one thing that MuisChat does not have. If I were to deploy this app, an addition like this would not be that hard. During my work with XCode, I learned that Firebase has its own notification system that is connected to the Apple OS. So, Whenever I would pass data to Firebase, I could easily have it send home screen notifications to IPhone users. Now for Android, this process is much harder, but can be done using the "Mynotifier" Python Module.

Stories

Although oversaturated in the industry, stories would make for a great addition to MusiChat. Every app has their own version of a story, and there is a good reason for this. Stories allow for instantaneous information, and they keep the user returning to the website. As for MusiChat, I would like to implement collaborative stories for song features, multi-artist writing, and multi-worker production. This way people can get the idea of what is happening with one project, but all of the people involved are credited and getting exposure.

Concert Live Streams *(Key Idea)*

Recently, there have been a lot of talks about NFT's and Cryptocurrencies. These talks tend to come from people who have strong convictions on the subject. During the end of implementation, I thought about using these topics for the good. My idea is that not only would an artist have a concert at a physical venue, but they could also stream the event privately. Now, this already occurs, but on platforms such as Twitch and YouTube. My idea would be to create an environment that is specific to this sort of event, and would allow for digital ticket (NFT) sales. These tickets would be sold in the form of NFTs and would allow you access to the show, the ability to chat in the concert chatbox, and access to expedited merchandise sales. This idea would almost need to be an entirely separate website, but I wanted to mention it along with MusiChat.

What Did I Learn?

When I started this project, I had never taken a Web Development course before. Throughout my learning process, I learned how to use Python (better), HTML, Flask, Django, XCode, Bootstrap, and MySQL. These technologies are all key components of many modern day tech, and will be the stepping stones upon which I will continue my learning.

At the beginning of the semester, I chose to start writing my app in Swift (XCode). This went well until I made the mistake mentioned earlier, which then led me to take a deep dive into Django. Through this journey, I learned that Django (in my opinion) is the best Web Development package for Python. Flask comes up short in a few areas, namely in the database side of things. The biggest upside to Django for my project was that you can interact with SQLite databases within Django, instead of accessing third-party packages; as someone who never had much experience with databases, this was significant.

This project also taught me about the usefulness of templates. The base of the website, as well as the stock images were acquired from a template provided by FreeCodeCamp.org. This template, along with Bootstrap's CSS templates, allowed me to expedite the development process. These templates saved me a lot of time due to the fact that CSS and Javascript are entire other languages. It was not realistic for me to learn all of the languages in such a short amount of time.

Other than these new technologies, I learned some skills that I will be able to use in everyday life. First of all, this project taught me how to manage my time accordingly. Normally, a project is overseen by somebody who gives you deadlines; in this case, I set all of my own deadlines. This type of independent work showed me that I am fully capable of completing professional work without intense oversight. I am certain that this trait will be useful in future endeavors.

Lastly, I learned to not stress myself out over the little things. When I lost my entire XCode project, I was pretty discouraged. This event made me worry that my project would not turn out well; this same feeling ended up giving me the motivation to move forward with the project. I then started to watch tutorials on Django and Django social media implementation, which then started the project that I have now. Although scary, this occurrence was good for me; as it reminded me that anything can happen in the workplace, as well as in life.

Conclusion

After studying web development and teaching myself these skills for months, I have the pride that I have completed something great this semester. I am excited to be able to enter the workforce with my given knowledge of a plethora of modern technologies. This project has taught me countless skills that I will be able to implement into my everyday life, as well as be

able to share with future co-workers/bosses. I am so happy to be able to share my idea of MusiChat, as well as its accompanying Powerpoint/Prototype and Project 2 implementation. I am so excited to see what the future holds. The Project 1-2 experience has not only helped me prepare for the workforce, but also for all of my future endeavors.

References (Project 1)

Zarrella, Dan. *The social media marketing book*. " O'Reilly Media, Inc.", 2009.

Welling, Luke, and Laura Thomson. *PHP and MySQL Web development*. Sams Publishing, 2003.

Evans, Liana. *Social media marketing: strategies for engaging in Facebook, Twitter & other social media*. Pearson Education, 2010.

Grinberg, Miguel. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

Lokhande, P. S., et al. "Efficient way of web development using python and flask." (2015).

Gaspar, Daniel, and Jack Stouffer. *Mastering Flask Web Development: Build Enterprise-grade, Scalable Python Web Applications*. Packt Publishing Ltd, 2018.

Spurlock, Jake. *Bootstrap: responsive web development*. " O'Reilly Media, Inc.", 2013.

Krause, Jörg. "Introduction to bootstrap." *Introducing Bootstrap 4*. Apress, Berkeley, CA, 2016. 23-32.

Verma, Jai Prakash, et al. "Big data analytics: Challenges and applications for text, audio, video, and social media data." *International Journal on Soft Computing, Artificial Intelligence and Applications (IJSCAI)* 5.1 (2016): 41-51.

Kosinski, Michal, et al. "Facebook as a research tool for the social sciences: Opportunities, challenges, ethical considerations, and practical guidelines." *American psychologist* 70.6 (2015): 543.