Conor Lyons (N00182816)

N00182816@iadt.ie

Dissertation submitted in partial fulfilment for the degree  
of BSc (Hons) in Creative Computing   
April 2022  
  
Supervisor: Catherine Noonan

Festivly- A web application COMPARATIVE study between recommender systems

# Declaration of Authorship

The incorporation of material without formal and proper acknowledgement (even with no deliberate intent to cheat) can constitute plagiarism.

If you have received significant help with a solution from one or more colleagues, you should document this in your submitted work and if you have any doubt as to what level of discussion/collaboration is acceptable, you should consult your lecturer or the Course Director.

WARNING: Take care when discarding program listings lest they be copied by someone else, which may well bring you under suspicion. Do not to leave copies of your own files on a hard disk where they can be accessed by other. Be aware that removable media, used to transfer work, may also be removed and/or copied by others if left unattended.

Plagiarism is considered to be an act of fraudulence and an offence against Institute discipline.

Alleged plagiarism will be investigated and dealt with appropriately by the Institute. Please refer to the Institute Handbook for further details of penalties.

The following is an extract from the B.Sc. in Computing (Hons) course handbook. Please read carefully and sign the declaration below.

Collusion may be defined as more than one person working on an individual assessment. This would include jointly developed solutions as well as one individual giving a solution to another who then makes some changes and hands it up as their own work.

DECLARATION

I am aware of the Institute’s policy on plagiarism and certify that this thesis is my own work. Student:

Signed: Conor Lyons

# Acknowledgements

I would like to thank my Supervisor Catherine Noonan for her support and feedback over the academic term as well as my second reader Louise Glynn for her advice on how I should move forward with the project. I would also like to acknowledge the rest of the lecturers for their on-hand support as well as my close friends and family for their never-ending assistance throughout the Major Project.

# Abstract

A recommender system, or a recommendation system, is an information filtering system that attempts to predict a particular ‘rating’ or ‘preference’ a user would have for an item. For the purpose of studying the intricate workings of Recommender Systems in further detail and to observe their role in presenting a user with similar items to their own preferences, it was decided to design and develop a web application based around a music recommender system. The purpose of this application is to observe the outcome a recommender system has with providing users with similar content whilst also personally honing my own skills in machine learning and new technologies.

Table of Contents

[Declaration of Authorship 1](#_Toc97805747)

[Acknowledgements 2](#_Toc97805748)

[Abstract 3](#_Toc97805749)

[Feasibility study & Requirements 6](#_Toc97805750)

[Introduction 6](#_Toc97805751)

[Requirements gathering/Research 6](#_Toc97805752)

[Similar applications 6](#_Toc97805753)

[User Profile 8](#_Toc97805754)

[Survey 8](#_Toc97805755)

[Personas 8](#_Toc97805756)

[Requirements Modelling 10](#_Toc97805757)

[Functional requirements 10](#_Toc97805758)

[Non-functional requirements 10](#_Toc97805759)

[System Model & Requirements 10](#_Toc97805760)

[Feasibility 10](#_Toc97805761)

[Test Plan 11](#_Toc97805762)

[Project Plan 11](#_Toc97805763)

[Conclusion 11](#_Toc97805764)

[Research 11](#_Toc97805765)

[Collaborative vs Content-Based Filtering 11](#_Toc97805766)

[Collaborative filtering methods 11](#_Toc97805767)

[Memory-based collaborative filtering 13](#_Toc97805768)

[Model-based collaborative filtering 13](#_Toc97805769)

[Content based methods 14](#_Toc97805770)

[Measuring Recommender System Accuracy 15](#_Toc97805771)

[User Experience and Recommender Systems 15](#_Toc97805772)

[Research Conclusion 15](#_Toc97805773)

[Design 17](#_Toc97805774)

[Introduction 17](#_Toc97805775)

[UI/UX – Application Flow 17](#_Toc97805776)

[System Architecture 18](#_Toc97805777)

[Recommender Engine 18](#_Toc97805778)

[Recommender Prototypes 19](#_Toc97805779)

[Web Backend 19](#_Toc97805780)

[Web Frontend 19](#_Toc97805781)

[Database Design 20](#_Toc97805782)

[Style Guide 20](#_Toc97805783)

[Design Summary 21](#_Toc97805784)

[Implementation 22](#_Toc97805785)

[Introduction 22](#_Toc97805786)

[Scrum Methodology 22](#_Toc97805787)

[Sprints 22](#_Toc97805788)

[Sprint Planning 22](#_Toc97805789)

[Daily Scrum 22](#_Toc97805790)

[Sprint Review 23](#_Toc97805791)

[Sprint Retrospective 23](#_Toc97805792)

[Bibliography 24](#_Toc97805793)

# Feasibility study & Requirements

## Introduction

The following document will highlight the requirements that will be necessary for the application to function. Over the course of this chapter, several requirements will be considered, and the study will go into detail about how the requirements were gathered such as similar applications and surveys. Throughout the course of this chapter, it will also highlight example personas for users that might want to avail of the web application as well as also detailing the functional and non-functional requirements. Finally, the various technologies that will be implemented into the application will be discussed.

## Requirements gathering/Research

### Similar applications

#### Amazon

Amazon.com has a fantastic recommender system that incorporates what’s known as ‘item-to-item’ collaborative filtering recommendations. In an article submitted by ‘McKinsey & Company’, 35% of sales on Amazon are down to their recommendation system. Amazon will recommend a wide array of products from a plethora of categories based on what the user is browsing through and display those recommendations to the user based on what they are most likely to purchase. One example is the ‘Frequently bought together’ section that is displayed at the bottom of the item page that the user is viewing in an attempt to convince the user to buy more than one item.

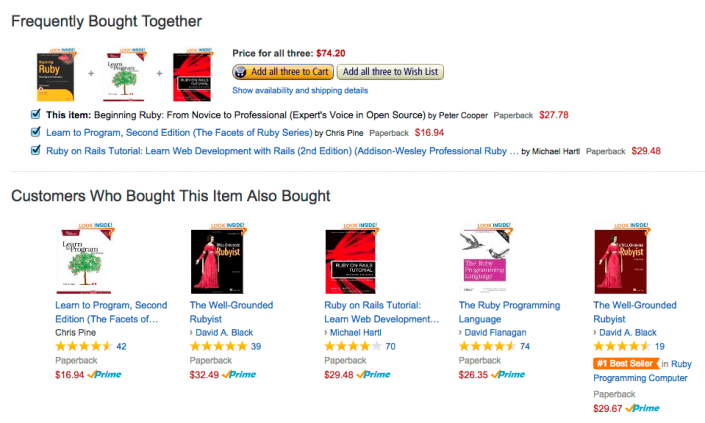


Figure 1: Example of Amazons Recommender System

While the Festivly application will not allow users to purchase music, the above can be seen as a concrete example of how recommender systems can offer multiple choice to the user and allow for more traffic and user-interaction on a website.

#### Netflix

The Netflix Recommendation Engine (NRE) has proven to be an incredibly popular and successful algorithm over the last decade. The NRE is comprised of several different algorithms that are designed to filter content based on each individual user profile. The recommender engine is capable of filtering over 3000 titles at any given time and incorporates a massive 1300 recommendation clusters based on a user’s preference. The NRE tracks a number of data points such as:

* Time and date a user watched a title
* User profile information such as age, gender, location, and selected favourite content upon sign up
* The device used to stream
* If the show was paused, rewound, or fast-forwarded
* If the viewer resumed watching after pausing
* Whether an entire TV series or movie was completed

According to Netflix themselves, the recommender system is so accurate that 80% of Netflix viewer activity is driven by personalized recommendations.

#### YouTube

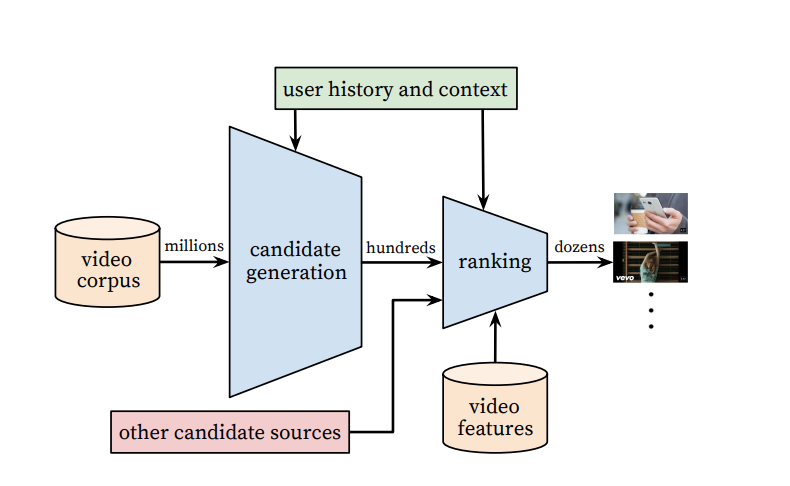
Contrary to Amazons recommendation strategy of item-to-item collaborative filtering, YouTube’s recommendation algorithm works by implementing user-user collaborative filtering into their recommendations. In short, the user will be recommended videos that similar users found interest in. This is measured by the length of time a particular user spent watching a video. If it is found that the user watched the entire video, then YouTube’s algorithm will recommend more videos based on other users who were receptive to the original video. 

Figure 2: YouTube recommendation system

From the above (Fig.2), you can see the recommendation system is comprised of 2 main components: Candidate Generation & Ranking System. The candidate generation system works by shrinking down a video corpus in the order of millions to a fine-tuned array of videos depending on the users watch and search history. The Ranking system will then take into account the candidate generation systems output along with a number of different features such as the user’s primary language, the language of the video etc. By doing this, it will then ranks the same in order of likeliness a given video will be the next a video a user watches.

## User Profile

The primary goal of the User Profile section is to highlight what the user expects to achieve with the software they intend to use during the Major Project. It is also designed to assist in understand the main requirements of users that will be interacting with the application and aim at satisfying these requirements.

### Survey

### Personas

In relation to designing and developing an application, a Persona is a fictional character that is created to gain a basic understanding of the average user that will be availing of the web application. Below, you can see a persona that was created using ‘Xtensio’.

Graphical user interface, website

Description automatically generated

Figure 3: Persona

## Requirements Modelling

In the Requirements Modelling section, the requirements of the application will be discussed, these requirements are based on survey feedback and previous research.

### Functional requirements

For Festivly, the following are considered functional requirements for the user to experience the application ideally:

* Users can Register and Login
* Users can search for a particular artist and see their top songs
* Users can search for a song and be given similar songs.
* User can save these songs

### Non-functional requirements

The following were considered nonfunctional when designing the application and aren’t perceived as detrimental to the final outcome of the application if they fail to be factored in.

* Form Validation when registering
* API for Music rather than a Dataset
* User can have custom profile picture
* Link to Artists Spotify/Soundcloud

## System Model & Requirements

The system model is split into three components, the frontend, the backend, and the database. The backend consists of Python and the necessary libraries that will be required to generate recommendations of the various songs/artists. The frontend will be the display that the user sees upon visiting the website, it will consist of a signup/login form followed by a homepage. From here the User may navigate to the recommender page and search for different artists/songs. This will be developed in Python Flask which will allow simple interaction with the python backend. The design aspect will be displayed using Jinja2, a templating engine which is packaged with Python Flask. The database, designed using SQLAlchemy, will hold all of the user’s data.

## Feasibility

|  |  |  |
| --- | --- | --- |
| **Challenge** | **Description** | **Solution** |
| Time Constraint | The application will start development in early mid-January and finish by late April. Given the complexity of the application it may be a struggle to implement the desired features on time. | With careful planning and setting bi-weekly deadlines it should be within reason that the main functional requirements of the application be implemented. |
| Lack of knowledge in required technologies | The recommender application will feature a set of never before used technologies and require a confident amount of knowledge in using them. | Working with the outset project plan and timeframes it should be possible to learn the necessities of each technology and how to apply them to the project. |

## Test Plan

To test my project, I will need an environment in which I can host and record user interaction. For last year’s project my partner and I used Loop11 to host and test our application for users to interact with. It lets the host set out and provide tasks that the user must complete to highlight whether a website can be navigated smoothly. This method worked very well last year with myself and my partner able to obtain very useful information that comes with a fresh pair of eyes testing your application.

## Project Plan

The project is being supervised by a mentor and will rotate around meetings where a discussion is had over progress and goals made over the previous weeks. The project management protocol being followed is Agile: which uses sprint cycles, splitting up the development process into smaller chunks which are easier to track and complete the goals set out.

## Conclusion

In Conclusion, the research obtained will come very useful over the course of this Major Project. Not only has it provided a foundation to build from, but it has also generated key resources that can be referenced over the various sprints that will no doubt benefit the team in the long term. Researching has also sprouted new ideas for the project with a recommender system now on the cards for the Festivly application.

# Research

The purpose of the Research document is to examine what technologies and documentation are available in assisting with the creation of a web-application music recommender system. Within this research document there will be an array of information on what recommender systems are, the different types of recommender systems and the algorithms that are contained within these systems.

Over the last decade, recommender systems have become instrumental in one’s day to day life. With the rise of YouTube, Netflix, Amazon and many other web services, recommender systems are unavoidable from e-commerce to online advertisement. In short, recommender systems are algorithms aimed at suggesting relevant items to users e.g what movies to watch, books to read, music to listen to etc. Recommender systems are critical in most industries as they possess the ability to generate a substantial amount of income when they perform efficiently.

## Collaborative vs Content-Based Filtering

### Collaborative filtering methods

Collaborative methods for recommender systems are methods that are based solely on the past interactions recorded between users and items to produce new recommendations. These interactions are stored in the so-called “user-item interactions matrix”.

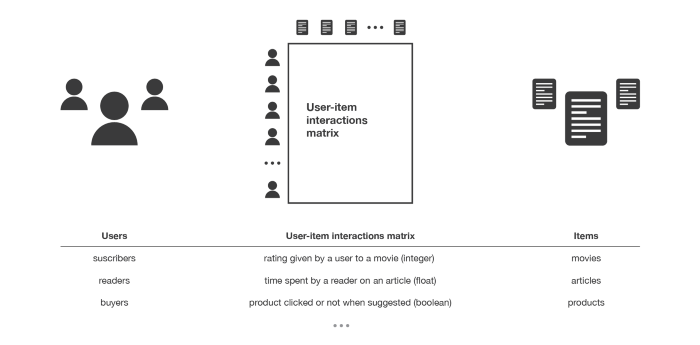


Figure 4: user-item interactions matrix.

The main aim of these collaborative based methods is that these past user-item interactions are sufficient in generating new recommendations for similar users/items and make predictions based on these proximities. Collaborative filtering is divided into sub-categories known as memory and model-based approaches. Memory based approaches works directly with the knowledge of previous interactions and rely on the ‘Nearest Neighbour’ search. An example of this would be to find the closest users from a user of interest and suggest the most popular items among these neighbours. Model-based approaches assume an underlying generated model that can explain the various user-item interactions and try to generate new recommendations based on the model itself.



Figure 5: Collaborative filtering methods

The key advantage of collaborative filtering approaches is that they require no prior information about users or items, and this allows them to be applied in multiple scenarios. Furthermore, the more users that interact with items on a website, the more recommendations become accurate. Within a fixed set of users and items, new interactions that are recorded bring new information over time and make the recommender system more effective. However, as collaborative filtering only considers past interactions to make recommendations it suffers from the ‘cold start’ problem. “It is impossible to recommend anything to new users or to recommend a new item to any users and many users or items have too few interactions to be efficiently handled”. This issue can be resolved with a number of temporary solutions whilst the website is in its infancy: Recommending random items to new users/ new items to random users, recommending popular items or new items to most active users (known as the ‘maximum expectation strategy’) or even using non-collaborative methods for the early life of the user.

### Memory-based collaborative filtering

As discussed above; memory-based is the standard in collaborative and commonly features neighbourhood based algorithms. These algorithms produce similarity scores between two items/users which then generate an array of predictions based on these. The main cited disadvantage of this type of filtering is its performance with large datasets. Another issue is that this area of filtering can be sensitive to sparse data – data with a large number of null cells. Datasets that include books or music can be considered sparse data as users may not have rated every index in the set or songs from the 1920’s-1950’s may have missing information.



Figure 6: Visual demonstrating how K-Nearest neighbour uses Euclidean distance to locate similar items in vector space.

### Model-based collaborative filtering

Model based collaborative rely on user-item interactions and a latent model to explain these interactions. Matrix factorisation algorithms consists in decomposing the large user-items interaction matrix into a product of two small matrices: a user-factor matrix (contains users’ representations) that multiplies a factor-item matrix (contains items representations).

Diagram

Description automatically generated

Figure 7: Matrix factorisation method

### Content based methods

Unlike collaborative approaches that require user-item interactions, content-based approaches use additional information about users/items. If we take a music recommender system as an example, this additional information can be the artist, the music genre, the duration of the song etc. The main goal of content-based methods is to attempt to construct a model that are based around the available ‘features’, that explain the observed user-item interactions.

Diagram

Description automatically generated

Figure 8: Content based methods

One major advantage that content-based methods have over collaborative is that is suffers far less from the cold start problem. New users/items can be described by their characteristics (content) and so accurate suggestions can be done for these new entities. The main type of content-based filtering happens in the form of ‘Cosine Similarity’ – Which measures the similarity of two vectors (items) by retrieving the cosine of the angle between the two items to determine whether or not they’re alike. The formula for calculating cosine similarity is below with A and B representing the respective vectors. The final answer of the formula will produce a figure between 1 and 1; -1 signifies that the items are not at all similar with 1 stating a strong similarity prediction.

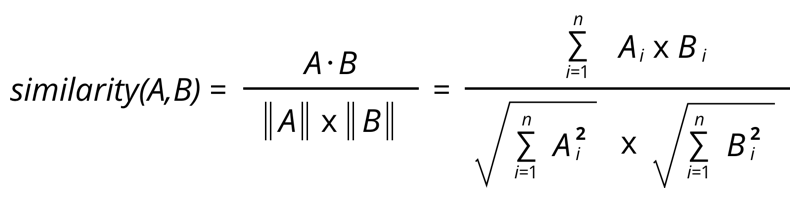


Figure 9: Formula for calculating cosine similarity

The major drawback of content-based filtering is that there is rarely adequate information to determine what items a user prefers and doesn’t prefer – leading to a very broad final output.

### Measuring Recommender System Accuracy

When creating a recommender system, it is common to implement several versions of experimentation to find the ideal technique for the intended dataset. A crucial area of choosing the right technique is to measure the accuracy of your recommender system. This is achieved using two measures: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). A few years ago, Netflix organised a challenge (the “Netflix prize”) where the goal was to produce a recommender system that performs better than its own algorithm with a prize of 1 million dollars to win. The winner was chosen on the basis that a team could achieve the lowest RMSE, influencing those competing to tweak their recommender systems towards that common objective. This ideology of creating recommender systems to achieve low RMSE scores has carried through to present day. However, researchers claim that user testing may be a better approach in developing more successful recommender systems.

## User Experience and Recommender Systems

The main reason we see recommender systems in high demand in today’s market is down to the significant increase they cause in sales. Algorithms that focus on user activity are implemented to generate accurate and relevant recommendations, which in turn causes increased sales figures as the user is offered items they are expected to enjoy and avail of. There are, of course, alternate factors that can alter the recommendation process and influence the user experience. It isn’t true that using a recommender system is only essential for increasing product views; good and efficient recommendation engines will offer a more satisfying experience for the user throughout their visit to a particular website and this also needs to be taken into account when picking the ideal algorithm for a user.

## 

## Research Conclusion

In Conclusion, the aim of the Research document was to assist the design and development of the music-based recommender system. By following the available resources and conducting in-depth research in to how the common recommender system functions, it has now become clear what kind of drawbacks may feature during the development. Firstly, it has become apparent that choosing the correct recommendation system will be difficult, whilst Collaborative based filtering seems to be a more favored recommender engine for more accurate results, it may prove difficult to implement such an engine into the application, content-based filtering has proven to be slightly favorable during initial prototypes. This could prove to be less effective if taken forward as recommendations may not be as fine-tuned and as a result leave the user feeling unsatisfied with what they have been presented with. Secondly, the question of what kind of data to use has also been raised. Whilst initially, the project will feature a dataset, the question of implementing an API has been discussed. Whether this can/will be added to the application has yet to be seen but will be a feature that will be looked further in to for definite. Nevertheless, understanding the expectations the user wishes to see will be critical in the implementation and design process.

# Design

## Introduction

This section will examine, in detail, both the UI/UX of the web application recommender system as well as the various technologies, architectural structure and the design patterns that were constructed. The technologies being used to create and develop this application are:

* Python
* Flask
* Jinja2
* SQLAlchemy
* GitHub
* Figma
* Miro
* VSCode

These technologies were chosen because as things stand with the level of knowledge regarding the first 3 technologies, I know little about them. They were introduced to the group at the start of 4th year and as a result something I want to further increase my level of understanding of. The Python language was briefly introduced at the start of final year in the Artificial Intelligence module and was something the team felt could be looked further in to in regard to a recommender system. GitHub is also being used as there needs to be a way of keeping a log of updates to the project as well as being able to share this project with the respective supervisors. Figma and Miro are technologies being used to wireframe and design the look of the web application and act as a drawing board for the project. Finally, VSCode will act as the project IDE and this will be discussed more in the Implementation chapter.

These technologies complement each other well so there should be no compatibility issues between them.

## UI/UX – Application Flow

After concluding research and receiving feedback from the surveys handed out, it was agreed that the UI would remain simplistic, allowing the user to avail of the website’s key features; mainly to generate recommendations based on the user’s inputted decision. The emphasis on design simplicity also transferred to the User Experience of the Festivly Application as the user is only required to fill out a single login/signup page to access the recommender system and not be pestered with providing a plethora of personal information to be presented with recommendations.

The next discussion was how the data was going to be inputted and presented back to the user. It was decided that using a simple text input and submission button that included a title and instructions would suffice. The user will be asked to enter their preferred music artist/song and upon submission be returned with a list of recommendations that are similar to their initial preference.

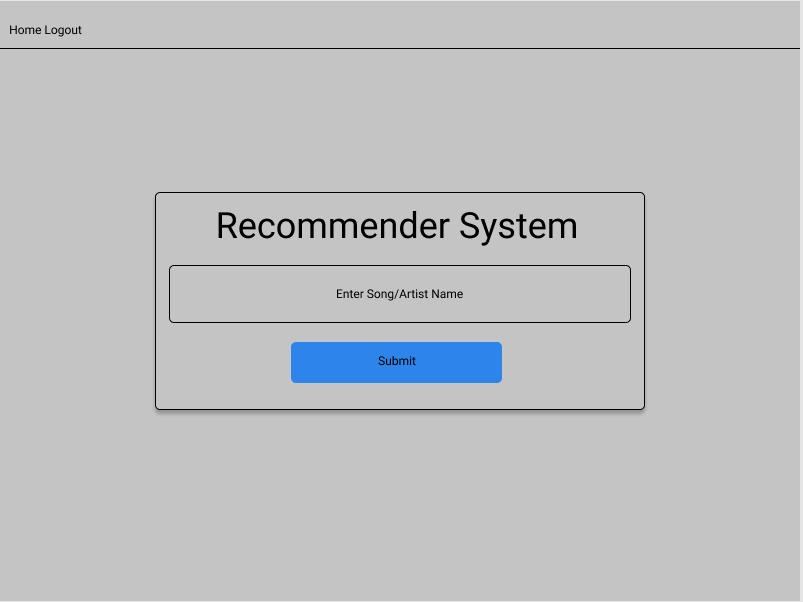


Figure 10: Wireframe of Recommender Input Page

After deciding the idea of how to generate predictions for the user, a basic wireframe was drawn up using Figma (Fig. 10). Next it was decided what dataset to use to provide recommendations to Users. There are several different datasets available on Kaggle, MillionSongSubset etc. So far, the team have yet to settle on one different dataset as each one contains different information, they find useful in different areas.

## System Architecture

There are three main components to the music recommender web application: the recommender engine, the web backend, and the web frontend. All three components are dependent on each other to operate and display the front-end web page which is shown to the user through templating provided by Jinja2. Each component will be explained in detail below.

### Recommender Engine

The recommender engine is the area of the application which will return recommendations for the user. Based on the research collected in previous sections, the algorithm being used is the Cosine Similarity algorithm. Given the complexity and the teams lack of knowledge on writing algorithms, it was decided to use existing algorithms from researched tutorials for the purpose of the application rather than attempt to manually write them. These tutorials are referenced within the Bibliography section of the paper. Other important libraries such as pandas and numpy were imported for use within the recommender engine. Pandas is a software library written for Python programming and specialises in data manipulation and analysis while numpy supports large, multi-dimensional arrays and matrices which are required to create recommendation systems. Once these libraries and algorithms had been chosen, the next part of development was to create an initial prototype.

### Recommender Prototypes

To effectively test the design of the recommender engine and to ensure the libraries implemented worked appropriately, a simple prototype was created. Using Jupyter Notebook, a server-client application that allows editing and running Python notebook documents via a web browser, a number of recommender prototypes were developed. The purpose of these prototypes were to gain further knowledge into how content-based recommender systems worked as well as ensuring the technologies being used had no compatibility issues between them.

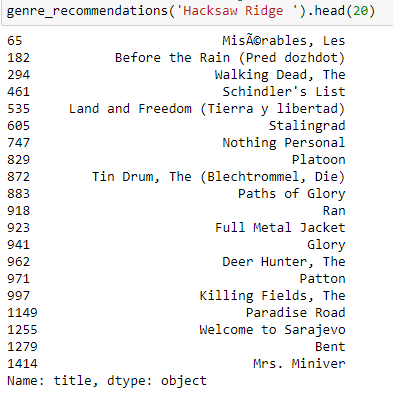


Figure 11: Initial Prototype developed

### Web Backend

The Web Backend is the component of the application which deals with making calls to the recommender engine; and then returning predictions to the user. The technology used for this is Python Flask; a lightweight web application framework which prioritizes development of simple web applications in contrast with the other popular python web framework Django; which is less flexible and more of a full-stack technology. The recommender engine will be kept outside of the websites folder which the backend will then be able to make calls to through one primary main.py file. Flask also offers routing functionality, so the main.py file will also be used to set up endpoints for the user.

### Web Frontend

The frontend of the application will be designed using HTML5, a basic CSS3 framework (Bootstrap) and Jinja2; a templating engine that comes packaged with Flask. Jinja2 works by allowing HTML templates to be set up that can be passed Python code when the route is called (such as variables or arrays) and process these into a required structure, before loading the page for the user. These technologies will be used to build the design shown and discussed in the wireframes previously.

### Database Design

For Users to be able to see the content on the Festivly website, a database is required to be able to store users’ information such as their email address, password, and name. The technology chosen for this database was SQLAlchemy, a library that facilitates the communication between Python programs and databases. SQLAlchemy provides a standard interface that allows developers to create database-agnostic code to communicate with a wide variety of database engines. The reason for choosing SQLAlchemy over another database software such as MySQL was that this technology is more suitable for basic testing and development, it doesn’t require the use of a web server like XAMPP to implement and fits into the overall development of the project quickly and easily.

### Style Guide

#### Colours

The following are a palette of colours that will be displayed on most of, if not all, the applications pages:

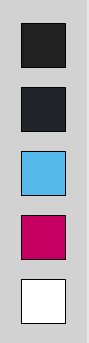
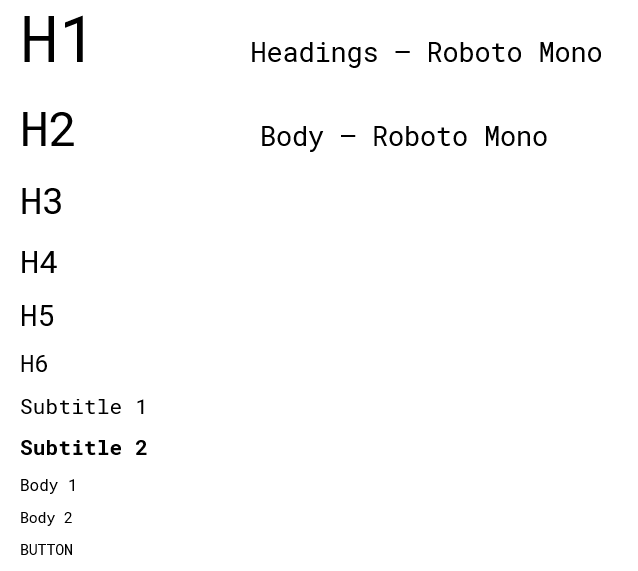
The first 2 dark colours will be the main theme of the website, opting for a darker presentation as it’s easier on the eyes and more appealing to a user which in turn will keep them on the website. The blues and pinks will be the button colours as they will be easy to see against the dark backdrop with the white being used for text as this is the clearest colour for the user to read against a charcoal black.

Figure 12: Colour Palette

#### Typography



This is the type-scale used in the application. This was created using material.io’s type scale generator. The Roboto Mono font family will be used for both headings and body text. The inspiration for this font came after viewing the Ticketmaster website as both sets of fonts look quite similar.

## Design Summary

In conclusion, these are the design features that will be seen in the Festivly application. The colour system and font types complement each other well with the first iterations of the recommender prototypes coming together nicely. There have been a few hiccups in development and trying to get things working but progress has been steady. The technologies being used are being studied more and more each day and the knowledge that is being obtained from further research is proving very useful in creating a more ideal recommender application. Overall, this design stage has proven fortuitous in setting the foundations for future development.

# Implementation

Introduction

Within the Implementation area of the project, the technologies being implemented, and the code structure will be discussed in detail. The purpose of this chapter is to give the reader an insight into how the project was developed and what steps were taken to achieve the final product. As discussed in the design chapter, the recommendation application was developed using three primary technologies. Python, Flask and Jinja. The main design technology will be Bootstrap.

* **Python**: Python has become one of the most popular programming languages in the world in recent years. It's used in everything from machine learning to building websites and software testing. It can be used by developers and non-developers alike. One of the most popular programming languages in the world, Python has created everything from Netflix’s recommendation algorithm to the software that controls self-driving cars. Python is a general-purpose language, which means it’s designed to be used in a range of applications, including data science, software, and web development.
* **Flask**: Flask is a web framework, it’s a Python module that lets you develop web applications easily. It has a small and easy-to-extend core: it’s a microframework that doesn’t include an ORM (Object Relational Manager) or such features.
* **Jinja**: Jinja2 is a modern-day templating language for Python developers. It was made after Django’s template. It is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.
* **Bootstrap**: Bootstrap is an open-source CSS framework that specialises in front-end web design and development. With Bootstrap it’s possible to create clean and concise looking webpages that follow a continuous theme. Bootstrap is vital in creating a tidy grid system that allows content to flow and fall into place neatly. The framework also allows for their content to be overwritten; this allows for changes to be made easily to match the design work that had been constructed before coding the application. All in all, Bootstrap was another critical element that allowed the project to be shaped the way the team intended.

## Scrum Methodology

The SCRUM methodology is a framework that allows for teams to work together more easily. The scrum consists of 5 different events. Sprint, Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective.

### Sprints

Sprints are the core values of the Scrum; this is the stage where ideas are turned into value. Sprints are like mini projects where there is a short goal to achieve by the end of the sprint itself. No longer than a calendar month, sprints are implemented to generate quicker learning cycles and avoid risk and cost due to its smaller time frame. Overall, they are designed to not put the main project at risk.

### Sprint Planning

Sprint Planning is the stage where the work needed to be performed in the Sprint is laid out. This plan is created by the combined work of the scrum team. Here, the project manager ensures that the scrum team talk about potential sprint drawbacks and how they can ultimately achieve the sprints’ main goal.

### Daily Scrum

The Daily Scrum is a critical process in the work and production of each sprint. The main purpose is to inspect progress towards the sprint goal and to adapt and adjust upcoming work if there is a backlog or problem. It is usually a small 15 min meeting and improves overall communication, decision-making and eliminates the need for larger meetings.

### Sprint Review

The Sprint Review is designed to inspect the outcome of a completed Sprint and to decide future adaptations. During the Review, the Scrum team review what has and hasn’t been achieved during the Sprint and decide on a plan of action for the upcoming, future sprints. The end result means that the Project backlog is altered and adjusted to achieve new goals

### Sprint Retrospective

The sprint retrospective is the time after a Sprint has been completed, where the scrum team meet to discuss potential ways to increase efficiency and overall quality for future sprints. They discuss areas of success and failure and where they will improve in the next sprint.

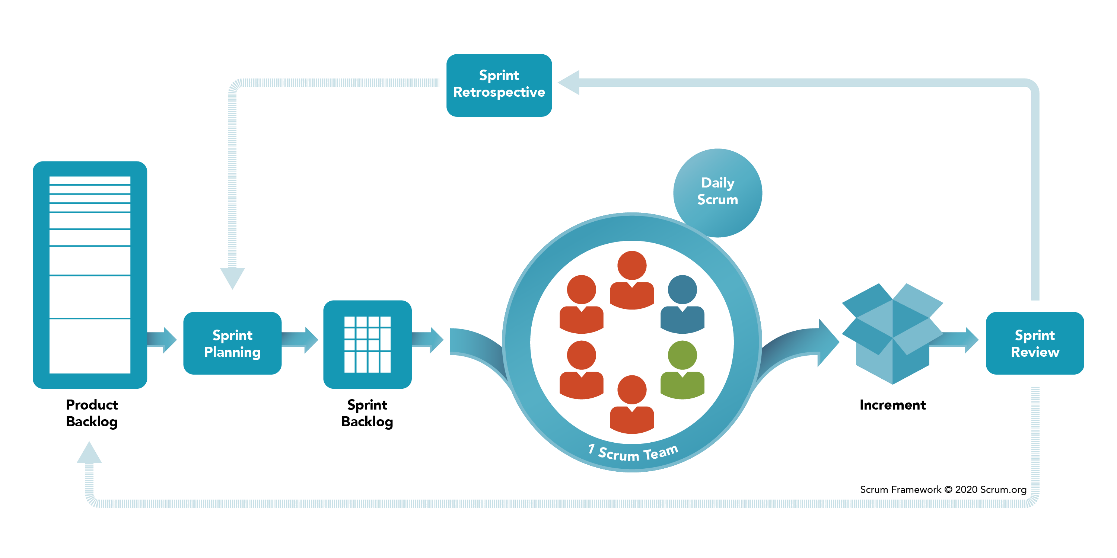


Figure 13: SCRUM Framework

Fig.13 describes the steps taken by the Scrum team to efficiently produce new innovations to their current product/service whilst constantly keeping in communication with themselves about how they can improve their overall work rate.

Development Environment

Throughout the duration of the project, Visual Studio Code was the primary Integrated Development Environment used to code the application. As an application itself, VSCode checks every box with support for multiple languages, syntax highlighting, bracket matching, auto-indentation and much more. Overall, this software made navigating the code structure much easier and proved a massive benefit for the project.

During the development of the Festivly application, Git Bash was also implemented and frequently used as a separate IDE to run and install the necessary packages for the python project. An easy to use Linux framework, Bash allowed VSCode projects to be launched quickly from its terminal as well as launching and running the server for the project whilst giving clear and concise error messages during overall development.

# Bibliography

Coursera. (2022). What Is Python Used For? A Beginner’s Guide. *Coursera*.

Dilmegani, C. (2017). Recommendation Systems: Applications and Examples in 2022. *AIMultiple*.

Dugar, D. (2018). Jinja2 Explained in 5 Minutes! *codeburst.io*.

Ian MacKenzie, C. M. (2013). How retailers can keep up with consumers. *McKinsey & Co.*

Kharwal, A. (2021). Spotify Recommendation System with Machine Learning. *TheCleverProgrammer*.

Krebs, B. (2017). SQLAlchemy ORM Tutorial for Python Developers. *auth0*.

Kumar, P. (2020). Movie Recommendation System | Python & Flask | Web Application | Heroku Deployment. *Medium*.

Makadia, M. (2020). Explained – Working and Advantages of a Recommendation Engine. *Business2Community*.

Meltzer, R. (2020). HOW NETFLIX UTILIZES DATA SCIENCE. *LighthouseLabs*.

Prakash, R. (2020). Implementing the YouTube Recommendations Paper in TensorFlow. *The Iconic*.

pythonbasics. (2022). what-is-flask-python. *pythonbasics.org*.

Rocca, B. (2019). Introduction to Recommender systems. *TowardsDataScience*.