

**LocalGigs**

**Final Year Project Report**

**DT211c**

**BSc in Computer Science (Infrastructure)**

**James Ward**

**Mark Foley**

School of Computing

Dublin Institute of Technology



# Abstract

The objective of this project was to develop a mobile application and Web application powered by a recommendation model that provides personalized live music event recommendations to users. The live music events are recommended using an item based collaborative filtering model that utilizes the Singular Value Decomposition (SVD) algorithm. The model is trained on a large corpus of data that details the playlist contents and listening habits of tens of thousands of Spotify users. The model also incorporates the individual taste and location of the user to provide event recommendations that are highly personalized.

As well as providing users with personalized event recommendations users have access to all events occurring in their location and the ability to search for events which they can then save to their profile. The venues for saved events can be located, and directions provided using an embedded mapping and routing feature.

Playlist information is gathered on each user via Spotify’s APIs, and events are attained using Ticketmaster’s APIs. The Web application and recommendation system are deployed as separate standalone applications that are accessible via API. They are hosted with a cloud service provider behind a load balancer for scalability and the Web application is served over HTTPS to ensure data transfer security for users.

The project was developed and deployed using the latest frameworks and tools. Django for the recommendation model, Django and PostgreSQL for the backend, server-side technologies, Django and Ionic for the front end, client-side technologies and Docker, Amazon Web Services and GitHub for deployment and version control.

Table of Contents

[Abstract 2](#_Toc5279004)

[1. Introduction 6](#_Toc5279005)

[1.1. Project Background 6](#_Toc5279006)

[1.2. Project Description 7](#_Toc5279007)

[1.3. Project Aim and Objectives 8](#_Toc5279008)

[1.4. Project Challenges 11](#_Toc5279009)

[1.4.1 Data Acquisition and Preparation 11](#_Toc5279010)

[1.4.2 Researching and Implementing the Recommendation System 11](#_Toc5279011)

[1.4.3 Programming Language and Framework Selection 11](#_Toc5279012)

[1.4.4 Separation of Concerns 12](#_Toc5279013)

[1.4.5 API Request Limitations 12](#_Toc5279014)

[1.5. Thesis Roadmap 12](#_Toc5279015)

[2. Literature Review 12](#_Toc5279016)

[2.1. Introduction 12](#_Toc5279017)

[2.2. Research Papers and Articles 13](#_Toc5279018)

[2.2.1. Basic Approaches in Recommendation Systems 13](#_Toc5279019)

[2.2.2. Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems 13](#_Toc5279020)

[2.2.3. Building and Testing Recommender Systems with Surprise 14](#_Toc5279021)

[2.3. Recommendation System Approaches 14](#_Toc5279022)

[2.3.1. Collaborative Filtering 14](#_Toc5279023)

[2.3.2. Content Based 15](#_Toc5279024)

[2.4. Machine Learning Algorithms 16](#_Toc5279025)

[2.4.1. K-Nearest Neighbour 16](#_Toc5279026)

[2.4.2. Bayesian Clustering 17](#_Toc5279027)

[2.4.3. Matrix Factorization 17](#_Toc5279028)

[2.4.4. Singular Value Decomposition 18](#_Toc5279029)

[2.5. Development Technologies 19](#_Toc5279030)

[2.5.1. Python 19](#_Toc5279031)

[2.5.2. Java 19](#_Toc5279032)

[2.5.3. R 19](#_Toc5279033)

[2.5.4. Spring Boot 20](#_Toc5279034)

[2.5.5. Django 20](#_Toc5279035)

[2.5.6. Angular 20](#_Toc5279036)

[2.5.7. Ionic 21](#_Toc5279037)

[2.5.8. PostgreSQL 21](#_Toc5279038)

[2.6. Libraries 21](#_Toc5279039)

[2.6.2. Pandas 21](#_Toc5279040)

[2.6.3. NumPy 21](#_Toc5279041)

[2.6.4. Scikit-Learn 22](#_Toc5279042)

[2.6.5. Surprise 22](#_Toc5279043)

[2.7. Web Hosting Platforms and Deployment 22](#_Toc5279044)

[2.7.2. Amazon Web Services 22](#_Toc5279045)

[2.7.3. Docker 22](#_Toc5279046)

[2.4. Apps, Software Tools, and Other Inspirations 23](#_Toc5279047)

[2.4. Alternative Existing Solutions 23](#_Toc5279048)

[2.4.1. Bandsintown 23](#_Toc5279049)

[2.4.2. Ticketmaster 24](#_Toc5279050)

[2.6. Conclusions 25](#_Toc5279051)

[3. System Design 26](#_Toc5279052)

[3.1 Introduction 26](#_Toc5279053)

[3.2 Software Methodology 26](#_Toc5279054)

[3.2.1. Agile 26](#_Toc5279055)

[2.3.1. Agile Scrum 26](#_Toc5279056)

[2.3.2. Agile Kanban 27](#_Toc5279057)

[2.3.3. ScrumBan 28](#_Toc5279058)

[2.4. Predictive Data Analytics Software Lifecycle 28](#_Toc5279059)

[2.4.1. CRISP-DM 28](#_Toc5279060)

[2.4.1.1. Business Understanding 29](#_Toc5279061)

[2.4.1.2. Data Understanding 30](#_Toc5279062)

[2.4.1.3. Data Preparation 30](#_Toc5279063)

[2.4.1.4. Modelling 30](#_Toc5279064)

[2.4.1.5. Evaluation 31](#_Toc5279065)

[2.4.1.6 Deployment 31](#_Toc5279066)

[3.3. Technical Architecture 31](#_Toc5279067)

[3.3.1. Front-End Design 32](#_Toc5279068)

[3.3.2. Middle-Tier Design 32](#_Toc5279069)

[3.3.3. Back-End Design 32](#_Toc5279070)

[3.4. Software Test plan 32](#_Toc5279071)

[3.5. Conclusions 32](#_Toc5279072)

[4. System Development 33](#_Toc5279073)

[4.1. Introduction 33](#_Toc5279074)

[4.2. Software Methodology 33](#_Toc5279075)

[4.3. Technical Architecture 33](#_Toc5279076)

[4.3.1. Front-End Development 33](#_Toc5279077)

[4.3.2. Middle-Tier Development 33](#_Toc5279078)

[4.3.3. Back-End Development 33](#_Toc5279079)

[4.4. Software Test plan 33](#_Toc5279080)

[4.5. Conclusions 33](#_Toc5279081)

[5. Testing and Evaluation 34](#_Toc5279082)

[5.1. Introduction 34](#_Toc5279083)

[5.2. System Testing 34](#_Toc5279084)

[5.3. System Evaluation 34](#_Toc5279085)

[5.4. Usability Evaluation 34](#_Toc5279086)

[5.5. Conclusions 34](#_Toc5279087)

[6. Conclusions and Future Work 35](#_Toc5279088)

[6.1. Introduction 35](#_Toc5279089)

[6.2. Conclusions 35](#_Toc5279090)

[6.3. Future Work 35](#_Toc5279091)

[References 36](#_Toc5279092)

[Appendix A: Moving to the Passive Voice 37](#_Toc5279093)

[Appendix B: Help with Writing 38](#_Toc5279094)

# 1. Introduction

## Project Background

Media distributions platforms such as Spotify and YouTube have drastically changed the music listening habits of people worldwide. For the first time in history companies can see what music you are listening to and when you listen to it. What music you listen to sometimes, what music you listen to all the time and everything in between. In the past music was spread by word of mouth or by wearing wristbands of concerts or festivals, this meant word travelled slowly and discovering new artists was a lengthy process. The radio stations knew when and what songs were being played on the radio and the record companies knew the number of CD, cassette or vinyl sales and so had they had a reasonable insight to the listening habits of people, but nothing on the scale of today’s media giants. Thanks to the rapid growth of these media distributers, the music analytics business today has an estimated worth of over two billion euro. At first glance it may appear these companies are taking business away from the record companies by giving artists the means to self-distribute and promote, but the data they are generating about artists is making it easier for record companies to pick out artists who are generating the right “buzz” in the music scene and sign them to contracts. Metrics such as Twitter followers likes on Facebook, sales via Ticketmaster or BandCamp and listens on Spotify can give record companies fairly accurate predictions of future prospects. Spotify in particular has a large effect on the new artists listened to by its users, the artists Spotify recommends are provided by its powerful recommendation systems.

Recommending the right music to users is a large part of what makes Spotify such a popular and successful platform. Both YouTube and SoundCloud provide users with platforms to listen to music, however, YouTubes recommendation system is not solely intended to recommend music and so is cluttered with other non-related material and SoundCloud does have a recommendation system, but neither are as popular as Spotify for listening to music. This in no small part is due to the extremely large corpus of data Spotify has gathered from its millions of users. The effective implementation of the right data analytic techniques provides Spotify with invaluable insight to its customers listening behaviours which enables them to recommend the right songs to its users. Spotify’s recommendation combines the data from all users into one huge dataset that predictions are made against. The model examines the listening behaviour of users and if their listening behaviours are largely similar their tastes are considered to be similar. Conversely, if two songs are listened to by the same group of users, they probably sound similar. This kind of information can be exploited to make very accurate song and artists recommendations.

Spurred on by the success of companies such as Spotify and YouTube recommendation systems have become one of if not the most successful and widespread implementations of machine learning technologies in business today. They have become the backbone of some of the largest companies in the world. According to a study by McKinsey, 35 percent of what consumers purchase on Amazon and 75 percent of what they watch on Netflix come from product recommendations based on strong algorithms. [1] One of the most popular implementations of a recommendation system is a collaborative filtering model. This is due to its accuracy, efficiency and scalability some of the most successful companies, such as Amazon and Spotify implement either a purely collaborative filtering approach or hybrid approach that includes collaborative filtering and another recommendation method. The other method is generally supplemental and intended to mitigate against the cold start problem associated with the approach.

The value of a well-designed recommendation system cannot be understated, in the age of huge data influx it is vital companies can efficiently analyse this data and utilize it to accurately predict the behaviours of their customers so they can continue to offer them products or services they are interested in. With the plethora of digital stores, media vendors and social platforms in the marketplace it is critical that companies differentiate themselves from their competitors. The best way for this to be achieved is to provide the user exactly what they want when they want it, failure to do so may result in loss of sales or user disinterest.

A primary use case of a robust recommendation system is to produce increased customer engagement. This is accomplished by providing users with the right recommendations. Customers can become overwhelmed when provided with too many recommendations, a robust recommendation system not only recommends the right products but the right quantity and variety of those products. This concept of increased customer engagement can be witnessed in full effect on the world’s largest video streaming platform YouTube. Its algorithms are specifically designed to keep customers engaged by recommending new videos tailored to pique the customers interest and keep them on the website generating ad revenue for the business.

## Project Description

This project will implement a collaborative filtering recommendation system that produces personalized live music event recommendations to users. In order to build a robust live music event recommendation system a large corpus of historical data and data specific to each user is needed to train the model. This involves constantly gathering data from the user about the artists they are listening to and their location and incorporating these key pieces of information into the recommendations that are produced. The finished recommendation system should be able to gather information on the user’s musical tastes and their location and produce recommendations of live music events that are of interest to the user.

The interests of each user are learned by retrieving playlist information from their Spotify account. All recommendation systems require that items are rated by users, these ratings are used to gauge a person interest in an item. Ratings can be found in many forms depending on what type of item the system is trying to recommend. For example, purchases or star ratings might be particularly useful when recommending new products to a user while likes and clicks might be indicative of what content should be rendered to a user on a social media platform. As this system will be providing live music event recommendations for artists the rating metric used is the number of tracks the user has for each artist in their playlists. The more tracks a user has for a given artist the higher the user has rated that artist.

Collaborative filtering recommendation systems use scores of data and powerful algorithms to find latent similarities between users and items and so the ability of the system to produce accurate recommendation relies heavily on the quality and volume of the data it is trained with. A large corpus of authentic data will be used in the training of this system it will be the genuine Spotify playlist contents of tens of thousands of users equating to an initial data set of nearly thirteen million data points.

The Web and mobile applications will be constructed in a clean uncluttered style and ensure adherence to the six fundamental principles of design balance, proximity, alignment, repetition, contrast and space. This will provide the user with an enjoyable user experience while still implementing all the desired functionality. The Web and mobile applications will be the main acquisition points for new user data, the constant acquisition of data is inherent to the success of the recommendation model. The data acquisition process will be efficient and reliable ensuring accurate up to date information is attained for all users. The Web and mobile applications will also provide up to date details on all live music events taking place in the user’s location. In order to provide maximum usability, the applications will provide user with a means to save events they are interested in for later reference and the ability to purchase tickets from Ticketmaster, listen to the artists using Spotify, or watch their music videos using YouTube. For any events the user has saved they will have access to a routing feature that will direct the user to the location of the event venue.

## Project Aim and Objectives

This aim of this project is to bring the benefits of the recommendation model to the live music arena in a clean, familiar and well-designed interface. The lack of promotion and therefore awareness of live music events is prevalent throughout the industry with only the highest earning artists receiving the necessary amount of promotion to make fans aware of their shows. A common theme that has been witnessed is for fans to only become informed of a live music event after it has occurred. This leads to frustration with the effectiveness of companies to promote artist and their shows, most of whom rely heavily on the income from their live shows due to the current state of affairs in the digital music industry. A platform that provides up to date recommendation on live music events would benefit the fans, the artists and in turn the businesses associated with the artists. This project attempts to address these problems by provide such a platform.

Several objectives were outlined at the beginning of this project that needed to be met for the project to be successful. The initial objective involved finding a sufficient source of historical data pertaining to the listening habits of Spotify users and their playlist contents. A method of retrieving the playlist contents of new users was also needed. After those objectives were achieved it was then necessary to prepare and combine these data sources in a format that is employable by the recommendation model. When the data acquisition had been completed and the data had been prepared it was essential that development of the recommendation model took precedence. Several techniques and machine learning algorithms needed to be tested and researched in order to find the approach most suitable for this problem. Upon completion of the machine leaning model the final objectives were to build the Web application and the mobile application. The goal was to put the focus on the Web application with the mobile application providing supporting functionality. The construction of the Web application needed to be separated into distinct sections the backend and frontend. The backend would implement the logic of the application, orchestrating the various components and providing connectivity to the recommendation model. The frontend would encompass the entirety of the client facing aspects, including any of the tools or features accessed by the users. The mobile application was intended to provide all the feature of the Web application by accessing the backend functionality of the application via API and then rendering this content in a mobile friendly manner.

The first objective is gathering the historical music listening habits and playlist contents data. The data will be used to train the model to predict artists a user may like based on the values attained for other users and artists in this dataset. Such the dataset that will be adopted for this project is the #nowplaying dataset. This is a Twitter based project that leverages social media for the creation of a diverse dataset which describes the music listening behaviour of users. [2] Twitter users regularly post information about the music they are currently listening to which was the basis for construction of this dataset. A subset of this massive thirty-six gigabyte dataset, the playlist dataset will be used for this project. The playlist dataset is based on a subset of users that publish their #nowplaying tweets via Spotify. The users in this subset publish the songs they are listening to as an embedded link to Twitter the meta data from which is used to build up playlist information from the user. The playlist dataset is a csv file that contains just shy of twelve million data entries in the following format, a hashed user name followed by an artist name a track name and a playlist name.

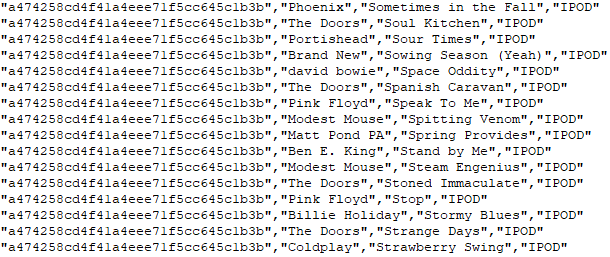


Figure : Excerpt from #nowplaying dataset

The second requirement that needed to be met is the ability to gather information on the listening habits and playlist contents of new users. This data needs to contain information on the quantity of artists and tracks the user has saved in their playlists. In this project that data was attained by utilizing Spotify’s APIs to gain access to the users Spotify account and retrieve their playlist data.

After the data had been attained it was necessary for it to be pre-processed and cleaned. The data in the #nowplayling dataset needed to be aggregated based on users and artists in order to get it into the required format of user, item, rating for the recommendation model. The dataset was aggregated down to the format of a hashed user name followed by and artist name and a track count signifying the number of tracks a user had for any given artist. This process alone reduced the dataset from close to thirteen million entries to just under three and a half million entries. Further cleaning and processing were performed on the dataset to remove any malformed entries and to set the rating scale from one to ten thus limiting the maximum score a user can give to an artist. Such is another pre-requisite of a collaborative filtering model which requires a fixed scoring metric. Enforcing this type of rating system helps to deal with any outliers in the dataset and ensure there are no entries for artists that are unusually high which could impair the ability of the model to make accurate predictions. The playlist data was retrieved from the new users is in the form of an API response and so pre-processing similar to the historical dataset was required before it was utilized in the recommendation model.

The next objective in this project is to build the recommendation system. The first step to achieving this is researching the type of recommendation system to implement and then deciding which machine learning algorithm to use. The algorithm must produce an effective model that can efficiently deal with very large volumes of data and perform fast enough to operate in real time. The decision process for choosing the recommendation model and algorithm followed a CRISP-DM approach by first focusing on the data and the problem that needed to be solved before choosing the model or algorithm.

The final objective of this project was the development of the Web and mobile applications. This portion of the project needs to orchestrate the entire functionality of the project incorporating the recommendation system into an interactive front end that allows users to retrieve their recommendations. The first step in achieving this is the development of the backend API endpoints, business logic and database connectivity. Once the backend functionality has been implemented the next objective is constructing the frontend of the Web application and mobile applications to provide users with the live music events that are generated by the recommendation model.

## Project Challenges

### 1.4.1 Data Acquisition and Preparation

One of the challenges faced in this project was the acquisition of a sufficiently dense dataset and the subsequent cleaning and processing of that data. A plethora of different dataset were examined before the #nowplaying was decided upon, many of the datasets that were analysed prior to the #nowplaying dataset were too small and would not have produced a diverse and realistic recommendation model. Cleaning the data was necessary to ensure the model could be accurately trained, the data preparation included several steps as mentioned in the project objectives. Firstly, removing or correcting any malformed data. Many of the artists names were empty or populated with quotation marks or surrounded by special characters these entries were removed or edited. Choosing the scale of the rating metric to optimize it for the recommendation system was also tricky as there is no specific scale required for the algorithm the decision and came down to an educated decision based on analysing the data and sentiment analysis from peers.

### 1.4.2 Researching and Implementing the Recommendation System

Researching the multitude of different recommendation systems and choosing the correct one was one of the most challenging aspects of this project. There are a wide variety of implementations available when building a recommendation system each having their own advantages and disadvantages. The final decision on what approach to use was based on the analysis of industry standard approaches to similar problems, information discovered through reading research papers and the implementation that won the Netflix prize. After the type of recommendation system had been chosen the best suited algorithm for the problem had to be chosen. This was chosen much in the same way the type of system was. However, some comparison testing was also implemented to verify the cases made in the researched papers. The algorithm that would be chosen needed to work very well with large datasets and sparse matrices due to the nature of the data and the model chosen.

### 1.4.3 Programming Language and Framework Selection

The research and selection of what programming language and frameworks should be used in solving the problem of building a Web application, mobile application and recommendation model was challenging. This research was performed with the aim of finding a versatile language that could handle as many aspects of the project as possible to mitigate against compatibility problems but that also provided a familiar development environment. The frameworks chosen needed to be robust and scalable and provide languages for developing backend and frontend features that followed a model view controller (MVC) approach to their architectural design. They also needed to be able to efficiently communicate with each other and the database which will be hosted separately in the cloud and store all user related data. The language and frameworks chosen played a pivotal role in choosing the integrated development environment best suited to this project.

### 1.4.4 Separation of Concerns

Another aspect of this project that posed a challenge was ensuring a separation of concerns was maintained between the different aspect of the application. It was essential the recommendation model be developed as a stand-alone application as it is operationally and computationally costly to run. Incorporating the model into the main applications backend would have drastically reduced the performance of the system and made the application far less scalable. This posed the problem of what aspects of data gathering and preparation should be handled by the recommendation model and what should be handled by the Web application backend. Deliberate reasoning and design choices govern the location of all pieces of functionality throughout the codebase to ensure the MVC architecture is adhered to and the application remains as modular and efficient as possible.

### 1.4.5 API Request Limitations

This project relies heavily on its ability to access data via APIs provided by Spotify and Ticketmaster. It was challenging to develop and application that stayed within the minimalistic request limit provided for the free tier use of these APIs. Ticketmaster not only put a limit on the number of requests an application can make per day but also the number of requests it can make every second. This request per second limitation has the most noticeable effect on performance on the entire system. Mitigating the effect of this policy meant compromises had to be made in the features and functionality of the Web application.

## Thesis Roadmap

One sentence summary each of the following chapters

# Literature Review

## Introduction

In this chapter some of the key areas of research that are important to this project will be presented. The development of this project involved continual research which led to ideas, plans and decisions mutating from what they were at their initial point of conception. The key areas of research that were investigated for this project are recommendation systems, machine learning algorithms, Web application technologies, mobile application technologies, deployment tools and cloud services. Any decisions made regarding the design or development of this project that are influenced by this research will be emphasized.

## Research Papers and Articles

### 2.2.1. Basic Approaches in Recommendation Systems

In their 2014 paper *“Basic Approaches in Recommendation Systems”,* Alexander Felfernig, Michael Jeran, Gerald Ninaus, Florian Reinfrank, Stefan Reiterer and Martin Stettinger detailed the basic approaches to a recommendation system including collaborative filtering, content-based filtering and knowledge-based filtering. The paper began by discussing the principles of the underlying algorithms used in recommender systems. It then provided insights into what recommendation technology to use in a certain application context. Thereafter, the paper provided an overview of hybrid recommendation approaches which combine basic variants. [3] The paper concluded with a discussion of newer algorithmic trends, especially critiquing-based and group recommendation approaches. The paper overall detailed the wide variety of possible applications of recommendation systems and provided very useful insight as to when a specific recommendation technology should be chosen and what algorithms work best with each approach.

### 2.2.2. Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems

In their 2002 paper *“Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems”,* Badrul Salwar, George Karypis, Joseph Konstan and John Riedl investigated the use of dimensionality reduction to improve the performance of collaborative filtering-based recommendation systems which have historically been difficult to scale due to the sparsity of the data in their databases. In the paper they mention testing multiple dimensionality reduction techniques before trying Latent Sematic Indexing (LSI) which implements Singular Value Decomposition (SVD) as its dimensionality reduction algorithm. They found that SVD which is a matrix factorization technique used for producing low-rank approximations of large sparse matrices out performed all other dimensionality reduction algorithms by producing the best low-rank linear approximation of the original data matrix. The paper further went onto describe in detail the process and mathematics behind SVD which involves the representation of each user and item by their corresponding eigenvectors and how to process of dimensionality reduction can be used the map users who have rated similar products but not necessarily the same product into the same space spanned by the same or similar eigenvectors. [4]

### 2.2.3. Building and Testing Recommender Systems with Surprise

In her 2018 article *“Building and Testing Recommender Systems with Surprise”,* Susan Li outlined the two most popular approaches to building recommender systems, content-based filtering and collaborative filtering. The article details the development and testing of a collaborative filtering recommendation model that utilizes the Python library Surprise which was built by Nicholas Hug. In this example she is working with a small dataset that details users rating of books. A perplexing result produced by the article was the algorithm that produced the lowest RMSE in her use case was a base line algorithm that had no specific tailoring. The article provides a widely applicable demonstration of the effectiveness of this powerful library. [5]

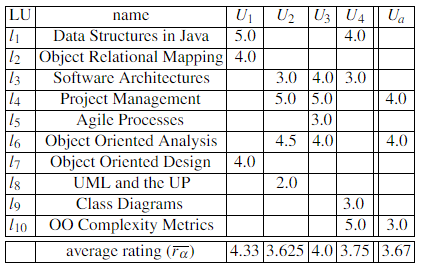
## Recommendation System Approaches

### Collaborative Filtering

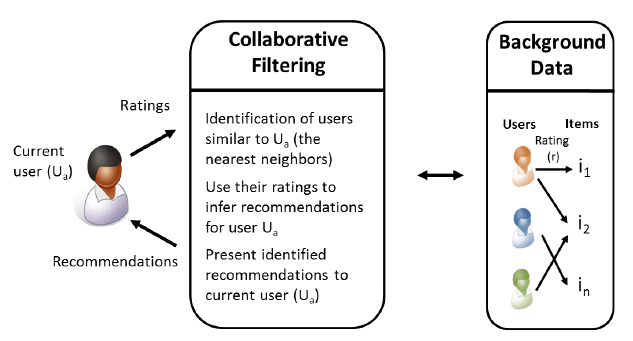
Collaborative filtering is based on the idea of word-of-mouth-promotion which highlights that the opinion of friends and family plays a major role in personal decision making. In a business context family members and friends are replaced by so-called nearest neighbours, who are users that have a similar preference pattern as the current user. [3] Collaborative filtering is a general technique for exploiting the preference patterns of these group of users to predict the utility of items for a particular user. Three different components need to be modelled in a collaborative filtering problem: users, items, and ratings. Most collaborative filtering methods fall into two categories: Memory-based algorithms and Model-based algorithms.

* Memory Based Algorithms – Store the training data that has already been rated in a database and from this data the rating of a user for a specific item is predicted based on corresponding users from the training data that have similar tastes.
* Model Based Algorithms – Statistical models are learned from the ratings of the users in the training data and predictions are made for test users against the trained model.

There are many approaches that can be taken when implementing a collaborative filtering system two of which are user-based and item-based. User-based models find users who have similar tastes based on the items they have rated, or in the case of this project their Spotify playlist behaviour chiefly, the artists and the number of songs they have stored for them. Item-based, items or artists in this case are recommended to the user based on the similarity of the items to other items.

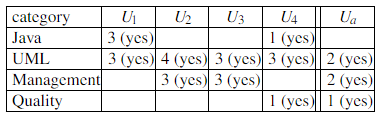


In general, all collaborative filtering approaches assume that users with similar tastes would rate an item similarly. This means that in all approaches some form of clustering is being exploited, either explicitly or implicitly. Compared with memory-based approaches, model-based approaches provide a more principled way of performing clustering and is also often much more efficient in terms of the computation cost at the prediction time. As one of the concerns for this project was the computational power required to make predictions in real time when such a large dataset is involved, it was thought to be a judicious decision to choose a model-based approach to building the collaborative filtering recommendation system. This would reduce the cost of deploying the model and reduce time required to produce recommendations. There is one significant drawback to the collaborative filtering recommendation system and that is the cold-start problem.[6] The cold-start problem describes the difficulty in making recommendations when the users or items are new or not very popular. The traditional approach to tackling this problem is to establish the user item profile via some interview process before generating any recommendations. A similar approach that has been taken with this project except the user profile is generated automatically by gathering the users Spotify playlist data before making any predictions thus mitigating against the cold start problem and eliminating the much-loathed interview process.



### Content Based

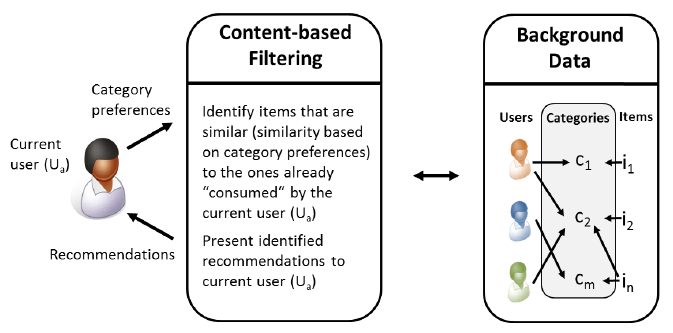
Content-based recommendation systems are based on the assumption of monotonic personal interests for example persons interested in Stock Exchange are typically not changing their interest profile from one day to another and will still be interested in this topic in the near future. [3] In a business context a user that reads an article or purchases a newspaper with information on the stock exchange will likely be interested in other articles or documentation about the stock exchange. A content-based recommendation system recommends items to users based upon a description of the item and a profile of the user’s interests. The profile of user interests in a content-based recommender system is built from data provided by the user either explicitly, from analysing a set of documents or descriptions of objects previously rated by a user or implicitly by clicking on a link. [3]



The profile is a structured representation of the user interests. This profile is adopted to recommend new and interesting items. Items that can be recommended to the user are often stored in a database table. A variety of learning algorithms have been adapted to learning user profiles, and the choice of learning algorithm depends upon the representation of the content. There are two key concepts at work in all effective content-based recommendation systems

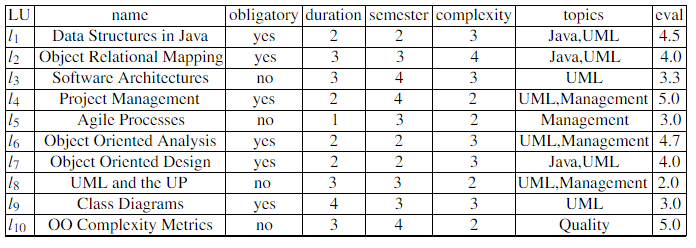
* Term Frequency – This is simply the frequency that a term or word appears in a document
* Inverse Document Frequency – This is a process that diminished the weight of a term or word that appears very frequently in a document and increases the weight of a term or word that rarely occurs.

The effective implementation of these two concepts helps to ensure the most important words are extracted from any pieces of data gathered about the users interests rather than the most frequent. The initial approach for this project was to implement a content-based recommendation system. The rationale behind this initial decision was mainly due to an overall lack of knowledge on recommendation systems after further research it was discovered that content -based recommendation systems typically function best when dealing with text documents. However, creating a hybrid recommendation system by combining a content based and collaborative filtering approach has been shown to produce very good results and is the method that Spotify utilizes. The content-based recommender helps mitigate against the cold-start that collaborative filtering models can be affected by.

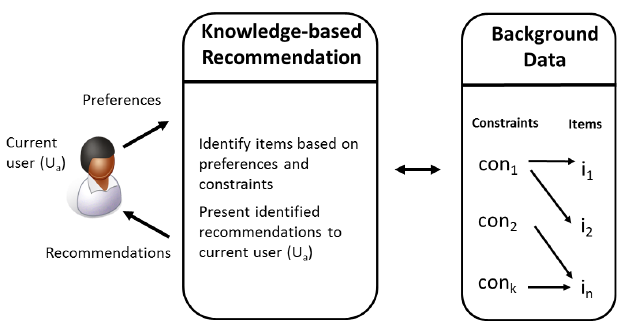


### Knowledge Based

Knowledge-based recommendation systems like the other approaches use knowledge about users and products to produce recommendations that meet a specific users’ requirements. However, they differ from content-based and collaborative filtering systems in that they do not recommend items based on a rating system or the textual description of an item. Rather they utilize deep semantic knowledge of the items, such knowledge describes each item in far greater detail than the other approaches and thus allows for a different recommendation approach. [3]



Knowledge based systems use this descriptive data combined with a set of constraints or rules and or similarity metrics provided by each user to produce its recommendations. The current user receives recommendations by specifying interest in particular item properties. An example of this would be topics = Java, this shows the user has a specific interest in java related topics and so any java related topics are recommended to the user. As the user’s interest profile builds into a set of interests the rules outlined by the user determine what new unsolicited items will be recommended. This recommendation approach is particularity useful for recommending items such as restaurants where the establishment, its location, the menu and its opening hours could be viewed as one large document of information. There is a variety of semantic knowledge required to accurately describe a restaurant and so using such data and acknowledge-based recommendation system a user could provide constraints such as distance and price and provide an interest in a particular food for example pasta and receive recommendation for a all restaurant that satisfy the restraints and the users interest in pasta. This approach to solving the music recommendation problem for this project was investigated and it was thought this could be useful as Spotify’s API provides vast amounts of meta data for each song. Every song has have a variety of semantic knowledge such as duration, genre, artist and much more, this approach would have worked better if the problem was recommending new songs to the user but as a rating system was being used to determine a user’s interest in an artist and not the description of their songs this approach was deemed unfit for purpose and so was abandoned.



## Machine Learning Algorithms

### 

### K-Nearest Neighbour

The K Nearest Neighbour (KNN) algorithm is the standard approach to similarity-based learning. Similarity based learning uses a distance metric such as Euclidian or Manhattan Distance to determine the distance between two points in a feature space. This distance between two points determines how similar they are, the closer they are the greater the similarity. KNN is a non-parametric lazy learning algorithm, the default distance metric used in the algorithm is Euclidian Distance. This is one of the simplest classification and regression algorithms that still provides good results and is commonly used. KNN uses a database of data points that are separated into classes, these classes are used to predict the classification of a new data points based on its distance from the K nearest points within the feature space. K is a number that the developer sets when implementing the algorithm.

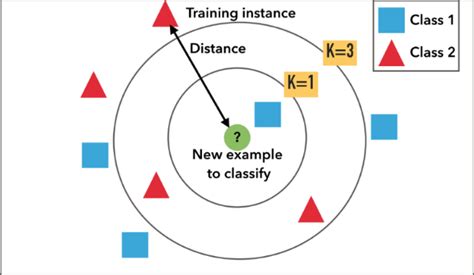


Figure 1: K Nearest Neighbour

### 

### Bayesian Clustering

Bayesian Clustering is a type of hierarchical clustering algorithm, these algorithms are one of the most commonly used approaches to an unsupervised machine learning model. These algorithms output binary trees whose leaves are data points and whose internal nodes represent the nested clusters of items. Bayesian Clustering assumes that similar users will rate the same type of item in a similar way. This allows for users to be grouped together into clusters according to the ratings they give for these items. Given a user class ‘z’, the preferences for different items expressed as ratings are independent, and the joint probability of user class ‘z’ and the ratings of items can be written as the standard naïve Bayes formulation. [6]

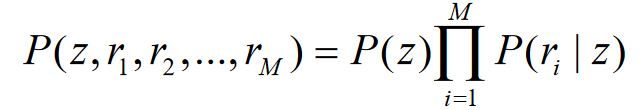


Figure 2: Naive Bayesian Formula

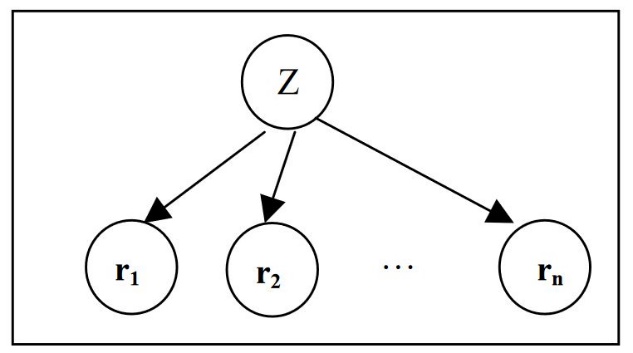


Figure 3: Bayesian Clustering

### Matrix Factorization

Matrix factorization also known as Matrix Decomposition is a discipline of linear algebra that is used to factorize a matrix into a product of matrices.[7] In other words it factorizes a large complex matrix into sub matrices which can be resolved to the original matrix using the dot product of two or more of the sub-matrices. There are two common methods when implementing matrix factorization *LU* and QR factorization. *LU* is used to factorize a square matrix into *L* a lower triangle matrix and *U* and upper triangle matrix. *QR* is used to factorize an *n x p* matrix into *Q* a square matrix and *R* an upper triangle matrix. Matrix factorization is extremely useful in recommendation systems when dealing with a very large sparse matrix like the type required by a collaborative filtering system, it helps to greatly reduce computation time and reveal latent information about the relationships between the users and items in the matrix.

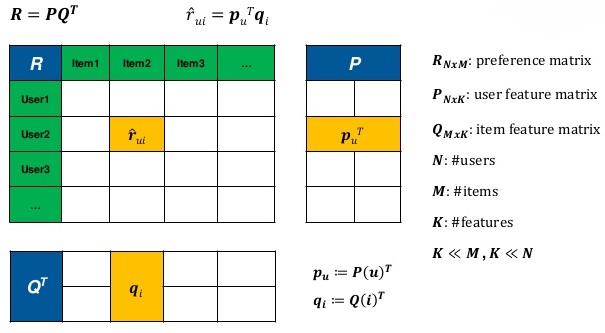
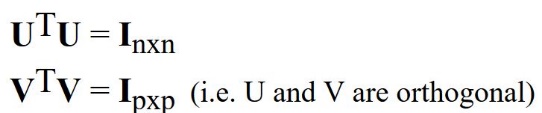


Figure 4: Matrix Factorization

### Singular Value Decomposition (SVD) and SVD++

Singular value decomposition is a form of matrix factorization, it takes a rectangular matrix of gene expression data (defined as A, where A is a *n* x *p* matrix) in which the *n* rows represents the genes, and the *p* columns represents the experimental conditions. The SVD theorem states:

 **Where** 

Where the columns of U are the left singular vectors (*gene coefficient vectors*); S (the same dimensions as *A*) has singular values and is diagonal (*mode amplitudes*); and VT has rows that are the right singular vectors (*expression level vectors*). The SVD represents an expansion of the original data in a coordinate system where the covariance matrix is diagonal. [8] Singular value decomposition is a complex approach to collaborative filtering, but it can produce very good results. It helps to identify latent features from the dataset in order to represent users and items as vectors in an N dimensional space. It helps to extract the most useful information form the dataset and quantify it so that accurate predictions can be made from it.

## Languages and Frameworks

### Python

Released in 1991 by Guido van Rossum, Python is one of the most popular and fastest growing programming languages today. It is an open-source, high-level, object-oriented, interpreted programming language that enforces indentation to ensure code readability. It is ideal for rapid development, it takes a high-level approach to data structures and it has an abundance of useful open-source libraries. Thanks to the wide variety and abundance of libraries Python is one of the top languages of choice for data science and machine learning. It has a great built in debugger which is also written in Python. Python also has a number of fantastic frameworks that provide developers well designed and tested platforms to rapidly develop applications because of all of these benefits Python was chosen as the programming language for this project

### Java

Released in 1996 by Sun Microsystems which was acquired by Oracle in 2010, Java is a very popular and widely applicable computer programming language, it is concurrent, object-oriented, class based and designed to have a minimal amount of implementation dependencies. Syntactically Java is very similar to C and C++, but it lacks some of the low-level facilities provided by these languages. Java was designed to allow developers to develop once and run anywhere because of this compiled Java code is capable of running on all platforms that support Java without requiring recompilation. Java is generally compiled to byte code and run on a JVM (Java Virtual Machine). Java also has several great frameworks for rapid application development and was the initial choice of programming language for this project. This choice was mainly based on familiarity however, a decision was made to invest the time in learning a new language (Python) and explore some of the benefits that it could offer in the realm of machine learning and Web application development.

### R

Released in 1993 by Robert Gentleman and Ross Ihaka, R was written primarily in C, Fortran and R, it is an implementation of the S programming language with the addition of lexical scoping semantics. It is an open source programming language that is mainly used for statistical computing and graphics. It is a very popular language among data scientists, statistician and data miners. R was initially determined to be the language of choice for the predictive analytics portion of this project but due to the accessibility of Python and the realization that time could be better spent researching solutions to the problem rather than learning a new programming language the idea was dropped.

### Spring Boot

Released in 2013, Spring Boot is a free open source project that has been developed on top of the Java Spring framework. It enables developers to create a stand-alone MVC (Model View Controller) application with minimal configurations. It provides the developer with default code annotations and configurations to allow for rapid development of applications. It provides the choice of either Maven or Gradle to manage the dependencies and libraries utilized in the project. Spring Boot generates much of the boilerplate code involved with developing Spring applications and it has an embedded HTTP Tomcat or Jetty server to allow testing of the application during development, it also has a wide variety of available plugins. Spring Boot was the initial choice of framework for this project but as it is a Java based framework it was no longer in contention when the decision was made to use Python as the main programming language

### Django

Released in 2013, Django is a Python based not for profit open-source framework. Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability) and "pluggability" of components, less code, low coupling, rapid development, and the principle of don’t repeat yourself.[9] Django implements its own version of MVC (Model View Controller) called MTV (Model Template View) where the view is the controller and the template is the view, it also eliminates the development of boiler plate code. It has a built-in webserver for testing during development and an object relational mapping system for dealing with database design that provides the developer with out of the box compatibility for some of the most popular database management systems such MySQL, Postgres and MongoDB. It is very easy to configure and has a very large community. Thanks to its large community it has a wealth of fantastic libraries that can aid in the development of services such as Representational State Transfer (REST), geolocation and social authentication to name but a few that were implemented in this project. Django is the framework of choice for this project it satisfied all the requirements outlined in the initial plan, to use a frame work that enabled rapid development with an MVC architecture that could be deployed and scaled with relative ease.

### HTML, CSS and JavaScript

Hyper Text Markup Language (HTML) defines the meaning and structure of all content on the Web. Hypertext refers to links that connect Web pages to one another, every HTML page is accessible through some link. It uses markup to annotate text, images and other content for display in a Web browser.[10]

Cascading Style Sheets (CSS) is a stylesheet language that is used to describe the presentation of a document written in any markup language (HTML, XML). It uses style declarations to detail how elements should appear when rendered on the screen, in speech, on paper or in other media. [11] Style declarations are made by creating key value pairs of styling properties.

JavaScript is a lightweight, interpreted, object-oriented scripting language for Web pages. [12] Although, it is found in many non-browser environments. It runs on the client side of the Web and it can be used to design and control how a Web page operates. It generally orchestrates the functionality of a Web page and is essential to building reactive feature rich Web pages.

### Angular

Angular is a JavaScript framework that uses TypeScript as its language for front end web applications, it combines declarative templates, dependency injection end to end tooling and integrated best practises to produce high quality applications. Angular is open-source and is developed and maintained by Google and several other organizations and individuals. It is used to build dashboard like applications that are geared towards optimizing user experience. Angular is feature rich with support for almost all JavaScript based libraries, it follows a single page application type design. Angular is utilized in the mobile development framework Ionic enabling it to produce native feeling applications for mobiles while using a more familiar Web application development environment.

### Ionic

Ionic is the world’s most popular cross platform mobile development framework. It is an open-source SDK (Software Development Kit) for developing hybrid mobile applications that is built on top of Angular and Apache Cordova. It has a plethora of native feeling elements for IOS and Android such as buttons, icons, themes and much more. It has a built-in webserver that allows developers to utilize the browser as a test bench for applications. The major benefit to Ionic is its cross-platform compatibility thus removing the need for platform specific frameworks such as Android Studio (Android) or Swift (Apple). This framework was used for the development of the mobile application for this project, having Angular as its frontend framework made accessing and rendering the content served from the backend via API relatively straight forward.

### PostgreSQL

PostgreSQL in an open-source object-relational database system that implements the SQL language with many other features that provide safe storage and scalability. PostgreSQL enables the development of scalable database architectures and fault tolerant environments, it is highly extensible allowing developers to define their own data types making it very flexible. [13] It provides many of the good features of SQL such as data integrity without the harsh rigidity associated with it. PostgreSQL is the database management system of choice for this project, it provides fantastic compatibility with Django with the ability to create and store complex data structures such as JSON (Java Script Object Notation) and Arrays and provides a well-designed graphical user interface called PGAdmin4.

## Libraries

### Pandas

Pandas is an open-source, BSD-licensed library for the Python programming language. It provides high performance and easy to use data structures as well as useful data analytics tools. The flexible data structures that Pandas offers allow developers to quickly manipulate large datasets into forms they can use to perform data analysis against. Some of the most useful data structures Pandas provides are the DataFrame and Series which represent data in a matrix format. Pandas was used extensively throughout this project to process and analyse the large volumes of data, its aggregation, sorting and cleaning capabilities were essential in preparing the data.

### NumPy

NumPy is an open-source, BSD-licensed library for Python. It is the fundamental package required for scientific computing in Python. [14]Among many others it provides developers with tools such as N-dimensional array objects, linear algebra operations, Fourier analysis, and random number capabilities. The N-dimensional arrays and linear algebra capabilities were particularly useful for the machine learning aspect of this project and are especially useful in the case of matrix factorization problems.

### Scikit-Learn

Scikit-Learn is an open-source, BSD-licensed library for Python. It is built on top of NumPy, SciPy and Matplotlib and is aimed at providing tools for data mining and data analysis problems. Scikit-Learn has tools for dealing with Classification, Clustering, Dimensionality Reduction, Regression and more. Scikit-Learn was used to analyse the data in this project and is a requirement for the Surprise library which uses it extensively in its recommendation system models.

### Surprise

Surprise is an open-source, BSD-licensed library for Python. Surprise is built on top of the NumPy and Scikit-Learn libraries, it aims to give users perfect control over their experiments by providing various ready-to-use prediction algorithms such as baseline algorithms, neighbourhood methods, matrix factorization-based and the ability to create custom recommendation algorithms. It also has built in tools for working with datasets and for evaluating, analysing and comparing the performance of the various algorithms. The matrix factorization algorithms mainly Singular Value Decomposition (SVD) and evaluation tools were used throughout this project. The SVD algorithm provided by library was developed based on the SVD algorithm used by the winning team in the Netflix prize which demonstrates both the effectiveness of this algorithm and the libraries applicability to this project.

### Django REST Framework

Django Rest Framework is a powerful and flexible toolkit for building Web APIs. [15] Among many other useful features, it provides multiple authentication policies including packages for Token Authentication, Oauth1a and Oauth2. A serialization system that supports Object Relational Mapping (ORM) and non-ORM data sources and view sets for combining the logic of a set of related views into a single class to increase code modularity. It also provides generic API templates for views to ease the development and testing process. In this project Django REST Framework was used to build the API endpoints that are used to serve most of the backend functionality of the Web application. Its serialization classes were used extensively in performing CRUD (Create Read Update Delete) operations on the database and the Token Authentication system was used to log in users accessing the system via mobile device.

## Cloud Service Providers and Deployment

### Amazon Web Services

Amazon Web Services (AWS) is a subsidiary of Amazon that provides on demand cloud computing services on a pay as you go basis. AWS is the largest cloud service provider in the world, it comprises of more than 90 services spanning a wide range including computing, storage, networking, database, analytics, application services, deployment and more. The services that are of most use for this project are the Elastic Compute Cloud units (EC2) and the Relational Database Service (RDS). EC2 is a virtual machine that can be configured to the developers required specification including the number of CPUs, operating system, storage capacity and networking capabilities. RDS is a service provided to host relational databases such as SQL, PostgreSQL and many more. For this project the EC2 service is being used to host the Django Web application and recommender system on separate instances and the PostgreSQL database is hosted using the RDS service.

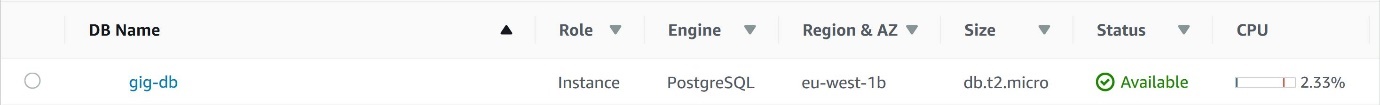


Figure 2: RDS Dashboard PostgreSQL DB

### Docker

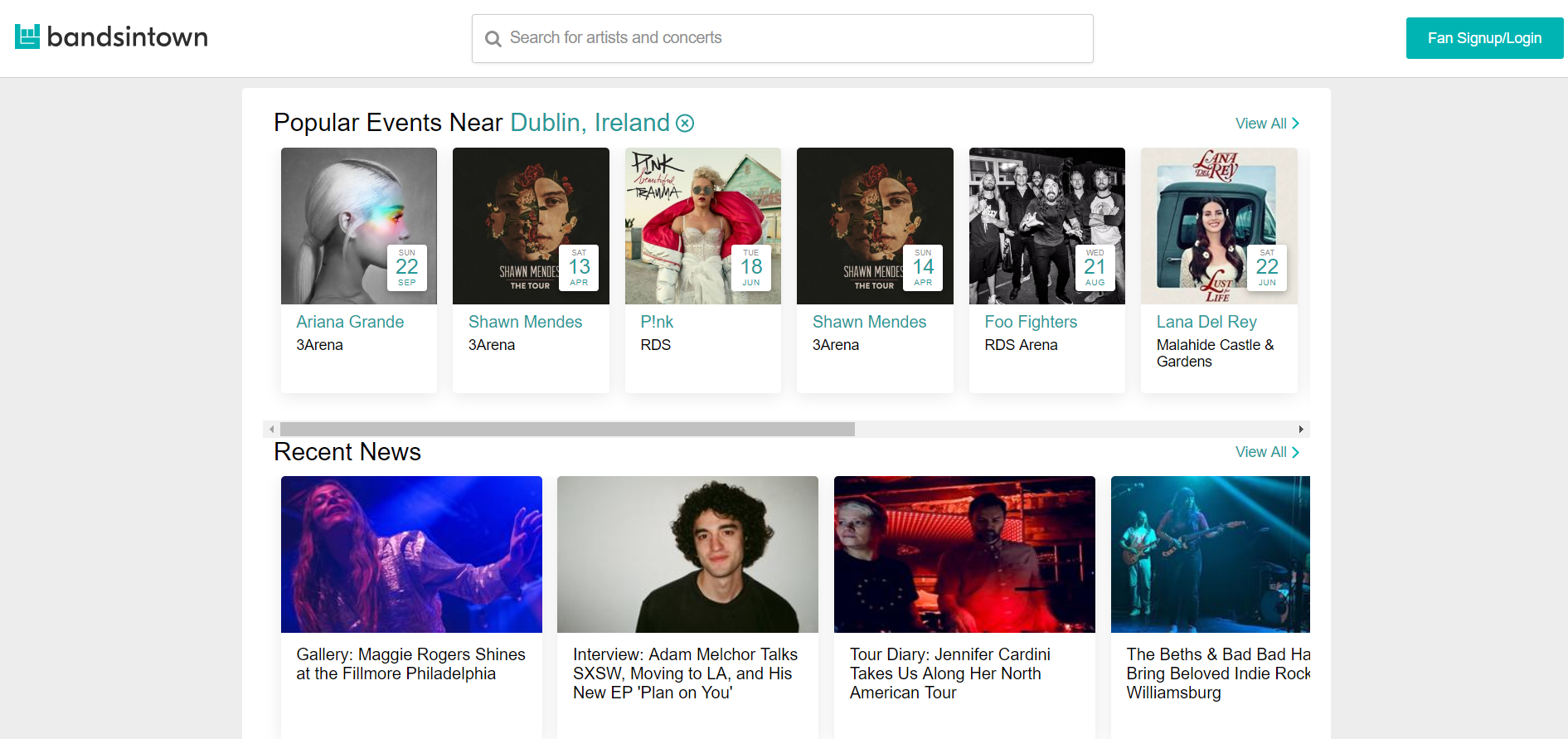
Docker is an open-source tool that is used to make it easier to create, deploy and run applications using containers. Containers are essentially packages that allow developers to wrap up their application with all its required dependencies and libraries and ship it out as one unit. This ensures that the application will work on any Linux machine so long as it has Docker installed on it. Docker works like an operating system for containers, containers work by virtualizing the operating system of a server and then run the application code on top. This allows developers to produce applications without worrying about the system the application will eventually run on it also benefits operational staff by giving greater flexibility and potentially reducing the number of systems needed because of its small footprint and lower overhead. Docker was used in this project to containerize the Web application and the recommendation system and deploy them to the EC2 instances. This was all accomplished with a Dockerfile and a handful of commands. A Dockerfile is essentially a specification file that tells the Docker kernel what packages and libraries to install in the container, in what directory the application should be placed, what scripts and commands to run and the entry point for the container when it is run.

## Alternative Existing Solutions

Several products and services exist in the field of music recommendation however no products or services could be found that recommended live music events. The closest substitute that could be found were products that sell and advertise tickets to events. There is a plethora of products that provide this service so two will be examined in further detail.

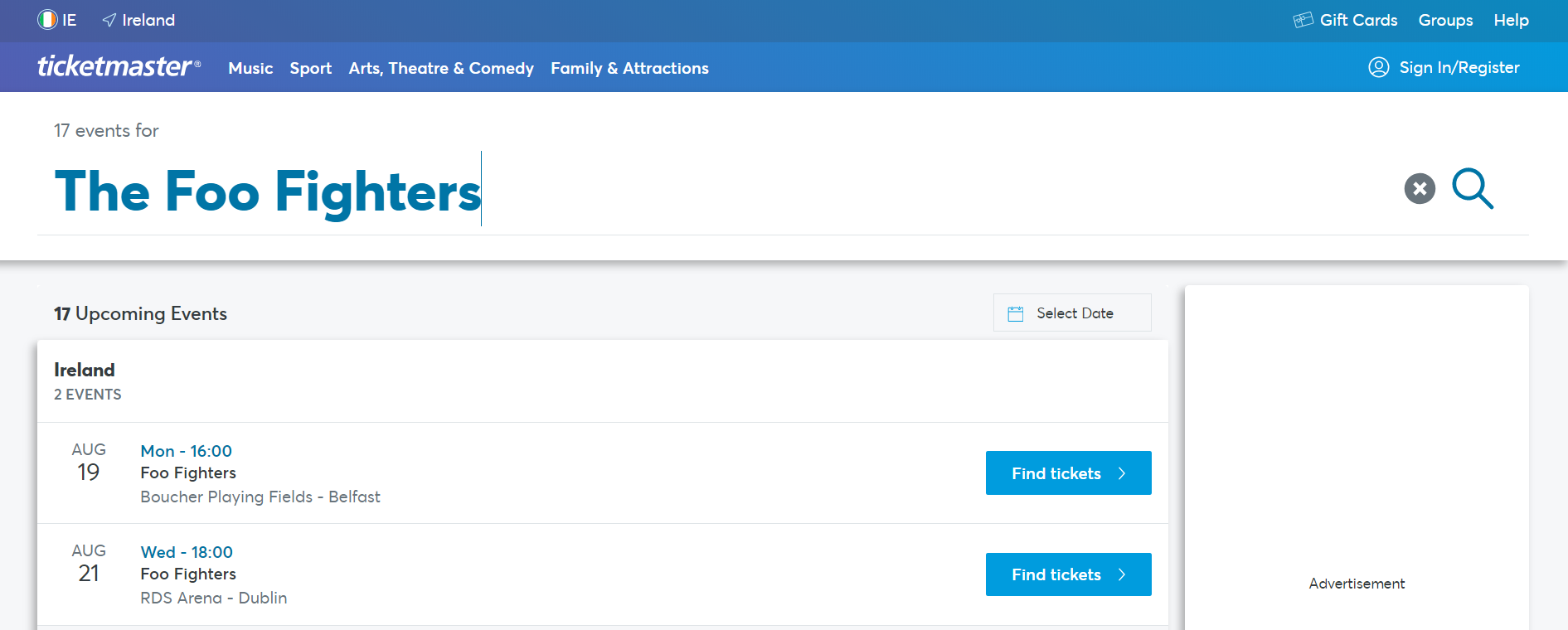
### Bandsintown

Bandsintown is a Web and mobile application that provides users with a platform to find and purchase tickets for upcoming live music events, it also provides a facility to book hotels or hostels in the city where the event is occurring. The website provides a brief description of each event including the name of the artists, venue, door time and information on what area of the venue each ticket grants the holder access to. Design inspirations were taken from this website. It has a clean and simplistic approach to its design, the colour scheme used throughout the platform was similar to what had already been developed when this product was discovered and so provided a depiction of what a finished product might look like. <https://www.bandsintown.com/>



### Ticketmaster

Ticketmaster is the largest and most popular ticket sales and distribution company in the world. It provides a secure platform for the sale and distribution of tickets to a wide variety of events. It provides information on the type of event, its location, the different show times and dates and in some cases provides a brief description of the event and critic reviews. Ticketmaster also provides access to much of the information about these events via its APIs. Ticketmaster played a pivotal role in this project and without its APIs it would not have been possible, the project also borrowed some design queues from its website. <https://www.ticketmaster.ie/>



## Conclusions

This chapter looked at some of the key background inspirations of this project, it first presented for relevant academic research, including … Following that we looked at … Finally a review of some existing Final Year Projects, include …

# 3. System Design

## 3.1 Introduction

Following on from the previous chapter, where some of the key background research was presented, these themes will be continued in this chapter, where the design of the system will be presented. The first section will look at the software methodology employed in this project which describes … The next section outlines the technical architecture of the system …

## 3.2 Software Methodology

The development of this project consisted of two distinct and separate processes, the development of the Web and mobile applications and the construction of the recommendation system. As such, it was prudent to take separate approaches to the development lifecycles of these aspects and implement methodologies that suited the different tasks. The following sections will outline some of the methodologies that were researched for this project, highlight the ones that were implemented and provide justification for their selection.

### 3.2.1. Agile

Agile software development is based on an incremental, iterative approach. It favours incremental design and development over approaches such as Waterfall which focus on in depth planning at the begging of the project. Cross-functional teams consisting of developers, systems administrators and management work together on iterations of a project over a long time in small periods called sprints which are typically between one and two weeks. The work is organized into a backlog that prioritizes tasks based on the business requirements. Agile methodologies are open to the requirements of a project changing over time and it encourages constant feedback from the end users. [16] This allows teams working in an agile lifecycle to deliver high-quality features faster and be more flexible and adaptable to change.

### 2.3.1. Agile Scrum

Scrum is a subset of Agile and one of the most popular process frameworks for implementing Agile. It is an iterative software development model used to manage complex software and product development. Fixed-length iterations, called sprints lasting one to two weeks, allow the team to ship software at a regular cadence. At the end of each sprint, stakeholders and team members meet to plan the next steps. Scrum follows a set of roles, responsibilities, and meetings that never change. For example, Scrum calls for four ceremonies that provide structure to each sprint: sprint planning, daily stand-up, sprint demo, and sprint retrospective. [16] During each sprint, the team will use visual artifacts like task boards or burndown charts to show progress and receive incremental feedback. Scrums daily stand up meetings provide more transparency and visibility to projects this helps to eliminate any misunderstandings within the team. Due to the lack of a project manager the team has increased accountability, the team works as a collective to decide on the best solution to the current problem. This decreases expenditure by removing a layer of management and allows the team to be more flexible and accommodate changes with greater ease.

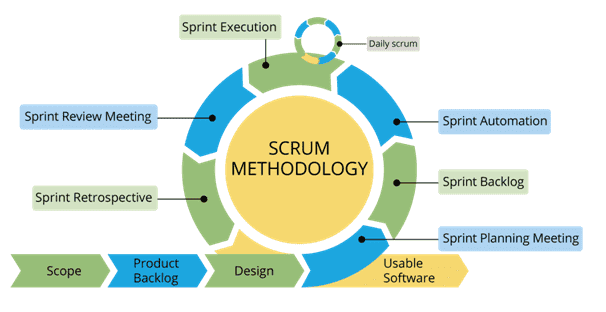


Figure 5: Agile Scrum Lifecycle

### 2.3.2. Agile Kanban

Kanban is Japanese for “visual sign” or “card.” It is a visual framework used to implement Agile that shows what to produce, when to produce it, and how much to produce. It encourages small, incremental changes to the current system and does not require a certain set up or procedure. This means, it is possible to overlay Kanban on top of other existing workflows. A Kanban board is a tool to implement the Kanban method for projects. A Kanban board, whether it is physical or online, is made up of different swim lanes or columns. The simplest boards have three columns: to do, in progress, and done. [16] The main benefits of Kanban include its ability to enable the visualization of the workflow which can provide a broader more abstracted view of the project as a whole. This feature allows for the projects backlog to be prioritized more effectively.

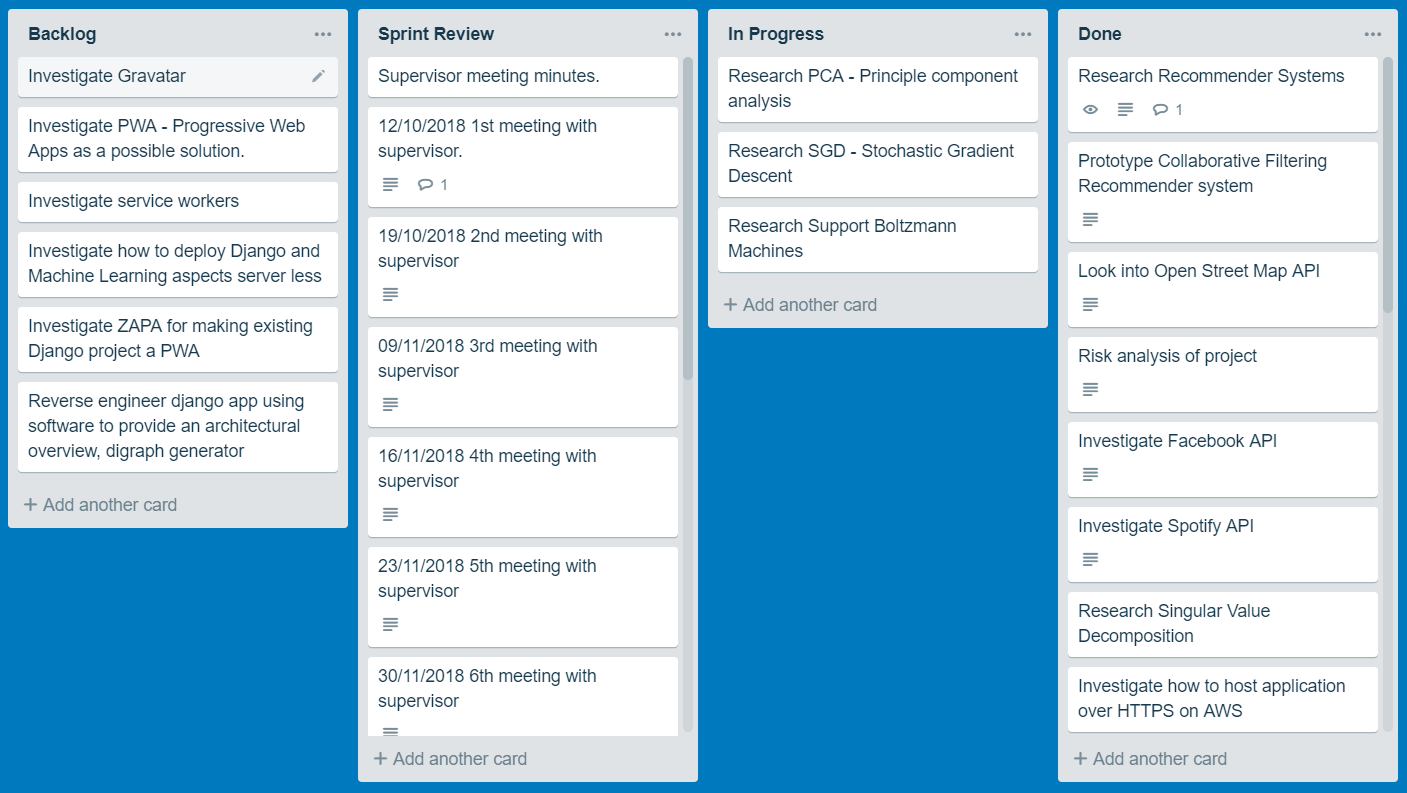


Figure 6: Agile Kanban Board (Trello)

### 2.3.3. ScrumBan

ScrumBan is a name coined for the hybrid software methodology used throughout the Web and mobile application development in this project. It is an implementation of Agile Scrums cyclic sprint development pattern that also incorporates the organizational benefits of an Agile Kanban board. The sprint cycle for this project is one week. This was decided upon so any additional features or problems could be demonstrated to the project supervisor who acted as a sudo scrum master. In order to keep track of the objectives and tasks to be completed a Kanban board was implemented using the popular free platform [www.Trello.com](http://www.Trello.com). This effectively acted as the backlog for the project and was updated constantly throughout the project’s entirety. I was used to detail the weekly meeting minutes with the project supervisor and to keep track of any ongoing work or research, potential features, and any completed work.

## 2.4. Predictive Data Analytics Software Lifecycle

### 2.4.1. CRISP-DM

Building predictive data analytics solutions for applications involves a lot more than just choosing the right machine learning algorithm. Like any other significant project, the chances of a predictive analytics project being successful are greatly increased if a standard process is being used to manage the project through the project lifecycle. One of the most commonly used processes for predictive analytics is the Cross Industry Standard for Data Mining (CRSIP-DM). [17] While there are several other development and design methodologies for data mining and machine learning, one such example being Sample Explore Modify Model and Assess (SEMMA) developed by the SAS Institute, CRSIP-DM has been chosen as the methodology that will be adhered to throughout the development of the predictive analytics portion of this project. The reasoning behind this is it outlines very clear and well-defined stages that are highly applicable to this project. The iterative cycle it proposes aligns perfectly with the development cycle for the rest of the project, it allows for constant improvements and changes as the scope of the project evolves. The recommendation model is a key component to this project and thus the adherence to a tried and tested development methodology is a prudent move in ensuring the production of a system that meets the requirements of this project.

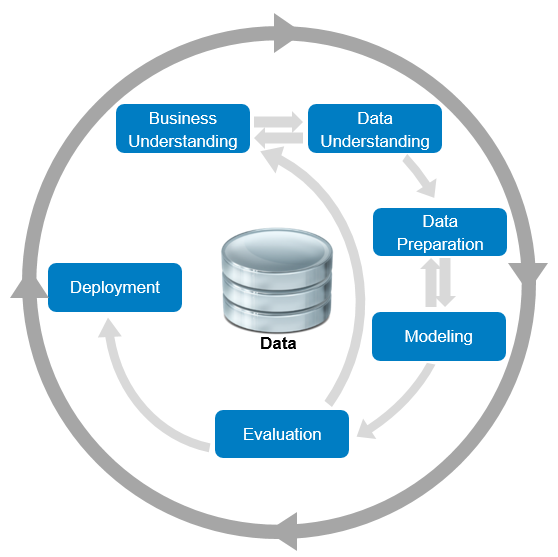


Figure 3: CRISP-DM Lifecycle

### 2.4.1.1. Business Understanding

The first stage in the CRISP-DM process is business understanding. Predictive analytics projects never start out with the goal of building a prediction model. Instead their focus is to gain more customers or sell more products all while improving the efficiency of their business. So, the goal in this phase of the development cycle is to fully understand the business needs so the right model can be produced. The goal at this stage of the process is to uncover important factors that could influence the outcome of the project. This can be achieved by outlining the primary objective from a business perspective, producing a plan that details how data mining and predictive analytics can achieve that objective and clearly defining the business success criteria.

### 2.4.1.2. Data Understanding

Once the objectives have been accessed from a business perspective, it is important that the data analyst then fully understands the different data sources available within the organization and the different types of data within those sources. If the data is gathered from multiple sources, it is important at this stage to consider how and when these data sources should be integrated. There are a few key aspects to this stage, the first is exploring the data, this involves utilizing data visualization and reporting techniques such as simple statistical analysis or producing graphs based on simple aggregations. The next stage is verifying the data quality, this entails examining the data and answering questions such as, is the data complete, are there any missing values? Is it correct, does it contain any errors and how prevalent are they? The final stage is to produce a data quality report, this report should list the results of the data quality verification and suggest solutions to data quality problems.

### 2.4.1.3. Data Preparation

Building predictive data analytic models requires specific kinds of data, organized in a specific kind of structure known as an Analytic Base Table (ABT). This phase of CRIPS-DM includes all the activities required to convert the disparate data sources that are available into a well-formed ABT. This involves deciding which data from which sources will be used for analysis, the criteria for which should be based on the relevance of the data to achieving the business objective. The main concepts at this point are, cleaning the data which includes raising the overall quality of the data and possibly implementing techniques such as imputation for handling missing values by substituting a plausible estimate or clamp thresholding for setting the upper and lower bounds of specific fields. The second step is constructing the data, this includes constructive data preparation operations such as the production of derived attributes or entire new records or transformed values for existing attributes. The final step is integrating the data, this involves merging the different data sources and any aggregations that are performed on the data.

### 2.4.1.4. Modelling

The modelling phase of the CRISP-DM process includes utilizing several machine learning algorithms to build a range of predictive models from which the best model will be selected. This is accomplished by following several processes the first of which being the generation of the test design. Generating a test design encompasses generating the procedure or mechanism by which the different models will be evaluated, for criteria such as speed, effectiveness and validity. The next step building the model included the construction of the different models, adjusting the parameters to attain the best results and describing the results of the model. The final step is assessing the model in which results of the model are noted and any revision of the parameter settings are implemented.

### 2.4.1.5. Evaluation

This phase of CRISP-DM covers all the relevant evaluation tasks required to prove the effectiveness of the tested models and determine which of the models is best suited to the business needs and to asses any other information gathered throughout the process. This may be accomplished by assessing the degree to which these aspects meet the business objectives or possibly by testing the models on test applications. After a model has been selected at this time it is important to perform a more thorough review of the data mining engagement to ensure no details have somehow been overlooked. This process should also include quality assurance measure to determine if the model was correctly built and tuned. The final step in the evaluation process is determining the next step. Depending on the successfulness of the evaluation process it should be determined whether another iteration of the lifecycle is necessary or if the model is fit for deployment.

### 2.4.1.6 Deployment

The final section of the CRSIP-DM lifecycle covers all the work that is necessary to successfully integrate the predictive data analytics model into the process within the organization. Achieving this includes determining a strategy for deploying the model, where in the business the model should be deployed or whether the system should be deployed on site or with some other service provider. Also included in this stage is the planning of monitoring and maintenance, having a robust monitoring and maintenance plan is one of the best strategies to mitigate against unnecessarily long periods of incorrect usage of data mining results. Finally, it is necessary to produce a report that give a detailed overview and review of the process in its entirety that also outlines any areas for future work and potential risk factors. [18]

## 3.3. Technical Architecture

. Each artist the user has in their playlists is stored along with a count of the number of tracks the user has for that artist. This dictionary of artists and track counts acts as the rating metric upon which the recommendation system is built.

This section discusses the technical architecture of the system, it first discusses the front-end design of the system, and presents some proto

### 3.3.1. Front-End Design

Screen prototypes – paper and computer-based

### 3.3.2. Middle-Tier Design

The connection type, security considerations

### 3.3.3. Back-End Design

ERDs and description of data

## 3.4. Software Test plan

## 3.5. Conclusions

In this chapter we looked at the design of the system, first exploring the methodology that will be used in the development process, next the technical architecture was outlined …

Based on the key themes discussed in this chapter, the next chapter will cover the development process and will be revisiting many of the same issues covered here.

# 4. System Development

## 4.1. Introduction

This chapter continues the issues explored in the previous chapter, and will outline the development process undertaken in this project …

## 4.2. Software Methodology

## 4.3. Technical Architecture

### 4.3.1. Front-End Development

Screen development, user interface design

### 4.3.2. Middle-Tier Development

JDBC or ODBC

### 4.3.3. Back-End Development

Creating the database, configuring the environment

## 4.4. Software Test plan

## 4.5. Conclusions

This chapter discussed the development process involved in this project, it started by outlining the software methodology, which is .. it then looked at the technical architecture …

# 5. Testing and Evaluation

## 5.1. Introduction

This chapter discusses the testing and evaluation of the system. In this chapter we will using the term *Testing* to refer to our own appraisal of the system, and *Evaluation* to refer to the appraisal of the system by other people and other pre-existing metrics. In this chapters the Testing will include … and the Evaluation will include …

## 5.2. System Testing

Not just Black Box and White Box, look at Unit Testing, Module Testing, Subsystem Testing, Integration Testing, and Acceptance Testing.

Automated Testing Tools.

Also the Test Plan again.

## 5.3. System Evaluation

Including some of the following:

* By users
* With automated tools
* Using rubrics
* Using benchmarks

## 5.4. Usability Evaluation

## 5.5. Conclusions

This chapter reviewed the testing and evaluation of the system. The testing included … The evaluation included.

# 6. Conclusions and Future Work

## 6.1. Introduction

In this chapter the key lessons learned from this project will be discussed. First a series of conclusions will be presented, starting with … Following this, a discussion of some further areas of research based on the work in this project will be discussing, including …

## 6.2. Conclusions

About 6-10 key conclusions

* 2-3 from Chapter 2
* 2-3 from Chapter 3
* 3-4 from Chapter 4
* 3-4 from Chapter 5

Each one 200-300 words

Conclusions = Summary + Justification (“because”)

Include diagrams in about 33% of them

## 6.3. Future Work

About 6-10 future work ideas

Each one 200-300 words

Include diagrams in about 50% of them

# References

|  |
| --- |
| **To cite in text** |
| Cite the name of the author and the date published or last revised:  (International Narcotics Control Board, 2017) |

|  |
| --- |
| **To cite a website:** |
| **In the References**  Author/Article title, Year, Publisher, Date of Publication, <Retrieved from URL>  International Narcotics Control Board 2017, United Nations, accessed 1 October 2017, <http://www.incb.org> |

|  |
| --- |
| **To cite a book:** |
| **In the References**  Last, First, Year, *Title*. City: Publisher.  Smith, Henry, 2017, *The Ambassadors*. Rockville: Serenity. |

|  |
| --- |
| **To cite a research paper:** |
| **In the References**  Last name, First initial, Year published, Title of Paper or Proceedings, *Title of Conference or Journal*, Location, Place of publication: Publisher.  Cloyd, A.M., 2017, “Surveying Students: A Look at Citation Habits of College Students”, *EasyBib Info Lit Conference*, New York City, 2014. New York, NY: EasyBib Publishing. |

# Appendix A: Moving to the Passive Voice

|  |
| --- |
| **To get rid of "I” statements:** |
| I will explore several technologies  Several technologies will be explored |
| I will research academic papers  Academic papers will be researched |
| There will be many benefits to my system.  There will be many benefits to the system. |
| There will be many benefits to the system I will create.  There will be many benefits to the system. |
| To achieve these goals I will be implementing a series of technologies  To achieve these goals a series of technologies will be implemented |
| As I will discuss later in Section 4.5  As will be discussed later in Section 4.5 |
| In this project I will be using java to develop the front-end  In this project java will be used to develop the front-end |

# Appendix B: Help with Writing

1. Just write down key words and phrases, don’t worry about writing full sentences, you can get to it later.
2. If you feel like you are repeating the same words or phrases, colour them in red to remind yourself to change them later, and move on.
3. If you are worried about stating things in the passive voice, just write everything in the active voice, colour it in red to remind yourself to change it later.
4. Using a speech-to-text tool like Speechnotes to speak your content
5. Look at other FYPs and papers for verbs and nouns, but don’t copy whole sentences.
6. Look for books in the library on technical writing