article

University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Mezzanine: A Recommender System for the Hospitality Sector

Marty O'Neill

08/04/2022

Abstract

The tools available for musicians and venue owners for finding work alike are cumbersome and offer more issues than solutions. A booking system for the hospitality sector is needed in order to speed up this job matching process and create. Mezzanine aims to be this solution, by utilising Machine Learning and Information Retrieval techniques to make informed job recommendations to musicians. Mezzanine aims to use these tools to create an effective user experience while also being easy to use and operate without any training needed.

Acknowledgements

Thank you to my advisor, Christos Anagnostopoulos, who provided me with guidance from start to finish.

Education Use Consent

Contents

1 Introduction

This chapter will examine the motivation and aim behind the development of Mezzanine, an interactive web-app for Venues and Musical Talent. Additional chapters in this paper will discuss the projects development life cycle, including the background, design, implementation and evaluation.

1.1 Motivation

In the spring of 2020, the United Kingdom introduced its first lockdown in order to combat the spread of Covid-19. Nearly every industry was affected. As traditional office jobs were being transitioned to allow employees to work from home, other industries that relied on face-to-face interactions had no choice but to furlough employees or let them go. By the end of 2020, the effect of the pandemic was realised. Within the Hospitality sector an estimated 640,000 jobs were lost, with over 10,000 licensed venues - clubs, restaurants, bars - closing their doors permanently. The music industry, an industry that is often partnered with hospitality, also saw a 35% increase in unemployment, resulting in 69,000 people losing their job. Each of these sectors have been severely affected and neither of them have fully recovered.

The UK government has introduced schemes designed to lift these industries back up to pre-pandemic levels. However, many independent artists within these sectors, referred to within this report as *Talent*, have also been affected and remain without compensation or consistent work.

Due to the nature of the job, individual artists face unique, complex challenges when finding work. They often rely on personal branding via social media, word of mouth, and directly contacting *venues* to enquire about potential openings. Conversely, venues face similar problems when trying to fill live music slots - managing multiple channels of communication can impede the selection process, leading to suitable fits being missed. Producing software that is tailored to this process involves analysing the needs of both the Talent and the Venue. Talent want to be shown relevant, open jobs and be able to apply for them. Venues must be able to create openings, receive applications and accept or deny applicants. But many factors must be included when constituting what is a 'relevant' job. For example, a well rated venue that is providing a potential job, but is very far away may not be relevant to the talent and therefore should not appear before other jobs that are better suited.

1.2 Aims

This paper proposes *Mezzanine*, an easy-to-use interactive web-app which aims to utilise information retrieval and Machine Learning practices to streamline the job-hunting process faced by musicians, and the recruiting process faced by venues.

Mezzanine aims to understand the business logic associated with creating, applying, and offering work to Talent, and create a flow of data that streamlines the process for each type of user. Venues must be able to create upcoming events, and Talent must be able to apply for these events. This

relationship between the users therefore means that the Venue and Talent profiles must be able to communicate with each other in order to have the job opening filled.

A Talent user must be able to search for work by location and, if events are available, be recommended the ones best suited to their profile. Due to this location service, some locations may have more open events than others. For example, a small village may only have 3 venues versus a well populated city with hundreds or thousands. This must be accounted for when building the system, as different methods of recommendation rely on different sizes of dataset. The application aims to analyse the amount of data that is available when recommending potential jobs to Talent, and use suitable Machine Learning and information retrieval techniques to make the correct recommendation.

In order to recommend the correct Venue to Talent, the data provided in the profile of each user must be analysed for similarities through the use of natural language processing techniques and then quantified so that mathematical concepts, such as similarity equations, can be applied.

Mezzanine aims to be modern and responsive with an emphasis on ease of use. All users should have clear visual aids that improve the experience when registering for, and using the application, while still extracting the necessary data needed to make relevant recommendations. Talent and Venues should have easy to understand profiles, allowing ease of access to information that will be required when making a decision on whether or not to apply to an event, or accept an application. Venues should have their address available, with an accompanying map to allow proper visualisation of the location.

1.3 Structure

The remainder of this dissertation is structured as follows:

- Chapter 2: *Background*. This chapter highlights the research that was undertaken and how it is relevant to the development of the system.
- Chapter 3: *Requirements Analysis*. This chapter discusses the functional and non-functional requirements and analyses how they were drawn from the aims.
- Chapter 4: *Design*. This section discusses the design decisions that were made with regards to the front-end and back-end of the system.
- Chapter 5: *Implementation*. In this section the development process is discussed; including the software engineering practices, the Recommender Algorithm, the Talent-Venue handshake and the front-end implementation.
- Chapter 6: *Evaluation*. Mezzanine was tested throughout development, and also at the end via a pilot study and user evaluation surveys. The findings are correlated and analysed in this section.
- Chapter 7: *Conclusion*. In this chapter the development life cycle is reflected upon, data is analysed and conclusions are drawn about the effectiveness and limitations of *Mezzanine*.

1.4 Summary

This chapter began by discussing the the motivation and aims behind Mezzanine. This highlighted the objectives of the system, and how it plans to aid in the post-pandemic recovery of the Hospitality and Music sectors.

2 Background

This chapter will highlight the background research that was undertaken with regards to the project. As mentioned in the introductory chapter to this report, the system revolves around *Talent*, *Venues* and the booking of Talent into *Events*. Each one of these objects is a source of data relevant to itself, but may also be related to instances of the other objects. This section will discuss the technologies used in the retrieval and processing of the data to derive similarities between each of the three sources.

2.1 Information Retrieval and Recommender Systems

Information Retrieval and Recommender Systems are branches of Computer Science that are relevant to the design, implementation and evaluation of Mezzanine. At a high level, the system must analyse qualitative input by a user and extract meaningful data (Information Retrieval) in order to make relevant item recommendations (Recommender Systems). Services such as Amazon, Netflix and YouTube use a combination of user-item interactions and user-user profile similarities to make recommendations. In Mezzanine, a similarity based approach is taken to make recommendations.

2.1.1 Precision, Recall, and the F1 Score

Precision, Recall and F1 scores are metrics used to measure the effectiveness of information retrieval systems (Log (2009)) In information retrieval, Precision refers to the ratio of correctly classified documents within the total returned set of documents. For example: a system returns 5 Events that it believes the user will find relevant, it is then realised that only 3 of the 5 are relevant to the user. This results in a precision of 3/5 or 60% precise for this recommendation.

Similarly, Recall refers to the ratio of relevant items retrieved from a set divided by the total relevant items contained within the set. For example, if a corpus contained 10 relevant Venues, and the system classified and returned 7 of these venues - then the precision is 7/10 or 70% precise.

Precision therefore can be described as the number of items retrieved that are meaningful. Whereas recall can be described as how many total meaningful items are returned.

Information retrieval systems have different needs and some may require more precision or more recall. (Shung (2018)) For example, medical scanning apparatus may focus on finding every occurrence of cancer cells (high recall) as false positives can be discarded upon further analysis without harm, while still finding the majority of true positives. When a system does not require one or the other explicitly, a balance of high recall and high precision is aimed for. This balance can be represented using the F1 Score. The F1 score falls between 0 and 1, with scores approaching one indicating high recall and high precision. The score is seen as a metric to analyse the completeness of the prediction.

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (2.1)$$

It is calculated by finding the harmonic mean of the combined precision and recall.

2.2 Natural Language Processing

Natural Language Processing (NLP) is a branch of artificial intelligence and linguistics that deals with how a computer interacts with naturally spoken or typed words and phrases - and how it makes sense of it. (SAS (2022)) There are many use cases for NLP, such as: Spam Detection for email inboxes, translation services like Google Translate, Apple's Siri virtual assistant and sentiment analysis used in social media. NLP research has provided techniques to aid in the extraction of data from raw language sources. NLP involves processing qualitative data in ways that it can then be used in quantitative ways, like mathematical concepts and Machine Learning.

In Mezzanine, NLP is used to process the information inputted by the users as they create their account or event. Each user, Talent and Venue, are asked to provide a description as they register for the site. This description represents who they believe they are as an individual artist or venue, and contains no bounds. This means the information entered may be descriptive or nondescript, may contain numbers and symbols, and may provide necessary data to match one user-type to the other. The system must extract useful information while ignoring the noise. This is done through pre-processing the descriptions of each object.

2.2.1 Pre-Processing Data

When analysing descriptive text - articles, biographies, profile descriptions - there are many factors to consider. For example, some words are more important than others, and others can be made redundant. In NLP, pre-processing is the first step in deriving meaningful information from a raw dataset. This is done using a number of techniques:

- **Tokenization** - Tokenization is the most important sub-step in pre-processing. This step involves taking raw and unstructured natural language and splitting it into relevant pieces - words, phrases, terms, sentences etc, referred to as tokens. (Menzli (2020)) For example: "I am a great guitarist!". After tokenization is applied, the string is represented as 5 tokens: 'I', 'am', 'a', 'great', 'guitarist!'. These tokens can be counted then used in comparison methods and also be represented as vectors in order to be used in Machine Learning techniques. These methods will be described in upcoming subsections.
- **Lower-casing** - When text is being processed, most programming languages will identify grammatically identical strings differently if they contain different letter casings. For example, 'hello', 'Hello' and 'hELLo'. These three strings mean the same thing, however are not seen that way by the compiler. Reducing every string in the corpus to the same casing removes this problem. In Mezzanine, this is very important. For example, if an Event description contains the phrase 'Guitarist needed' and a Talent's description contains the phrase 'I am a guitarist' - this must be identified as being similar, without pre-processing the letters case, this similarity may not be realised.
- **Stop-Words and Punctuation** - Certain words provide little descriptive value to a corpus. Words like: 'the', 'and', 'for' etc are all conjoining words that are crucial to the structure of sentences, but not to the meaning behind it. By removing these words from the dataset that is being analysed, the noise is greatly reduced. For the same reasons, punctuation holds little value when retrieving information from a dataset. Using the same example as above: "I am a GREAT guitarist!" becomes: "GREAT guitarist".
- **Lemmatization** - Lemmatization is the process of breaking down a verbose word into its meaningful base word by analysing the vocabulary. (Beri (2020)) For example: "I like performing guitar" and "We need someone who performs the guitar for this event". Both sentences contain the same meaning, but are phrased in different ways. In this scenario,

lemmatization can be used to convert the word 'performing' and 'performs' into 'perform'. This process is important when comparing descriptions of Venues and Talent as full words may not directly match, but the meaning behind the words do.

Consider two strings, X and Y, the first from a Talent profile description: "I am a Performer who enjoys playing the Guitar. I also DJ part time and I am actively looking to play for work.". The second from an Event description: "We are a Nightclub who are actively looking for a DJ to perform for us on Friday". After pre-processing, with each token being placed into an array, both strings are reduced to:

X = ["perform", "enjoy", "play", "guitar", "dj", "part", "time", "active", "look", "play", "work"]

Y = ["nightclub", "active", "look", "dj", "perform", "friday"]

Pre-processing reduces the noise from each description and reduces it down to key words that can be easily compared.

2.2.2 Bag of Words Model

The Bag of Words (BoW) Model refers to the process of analyzing a corpus - a collection of documents - and quantifying each document into a numerical vector. These vectors then represent that document, and can be compared easier against other documents in the corpus that have also been vectorized. In Mezzanine, the corpus represents the set of all Events and a single Talent object, where each Event and Talent profile is a document.

In order to create the document vector, the BoW Model combines all of the words found in the corpus into one dataset and then counts the occurrences of each word in each document. (Brownlee (2017)) Applying BoW to the document example seen in **2.2.1**, the count of occurrences and vectors results as:

Table 2.1: *Bag of Words Example*

Words in Corpus	Document X	Document Y	Combined Count
perform	1	1	2
enjoy	1	0	1
play	2	0	2
guitar	1	0	1
dj	1	1	2
part	1	0	1
time	1	0	1
active	1	1	2
look	1	1	2
work	1	0	2
nightclub	0	1	1
friday	0	1	1

The resultant vectors of X and Y are:

X = [1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 0, 0]

Y = [1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1]

These vectors contain data with regards to the documents, however they are not very descriptive and have limitations - A large corpus will contain hundreds, thousands or potentially more documents, each with its own vocabulary. This would result in each documents corresponding vector being very long, potentially containing many 0's.

Bag of Words acts as the foundation for more complicated methods of Natural Language Processing, in which more meaning can be derived from the data, such as TF-IDF.

2.3 Term Frequency - Inverse Document Frequency

Term Frequency - Inverse Document Frequency (TF-IDF) expands on the BoW model seen in **2.2.2**. It is a "text based statistical weighting technique(s)", often used in Information Retrieval (Alfirna Rizqi Lahitani (2020)). It incorporates the same concept - whereby each document in the corpus is vectorized. Except in TF-IDF, a *weight* is calculated for each document which is used to identify which words hold more importance. The greater the weight given to a word - the greater the importance that word has on the document.

Weights are calculated by multiplying two components, the term-frequency and the inverse document frequency.

$$\text{tf-idf} = tf(t,d) \cdot idf(t,D) \tag{2.2}$$

In a dataset containing thousands of restaurant descriptions, each description may contain the word 'restaurant'. For example: "A local Italian restaurant", or "Traditional Greek restaurant". This results in restaurant being less important than other words, so it should hold less weight compared to words like 'Italian' and 'Greek' which help identify the type of food that is sold there. This is applicable to Mezzanine for the same reasons - the TF-IDF weight is calculated through multiplying two values:

2.3.1 Term Frequency

Term Frequency is the first part of the calculation and refers to the number of times a term or word occurs in a document. It is calculated by counting the number of occurrences of a term, *t*, in a document, *d*, and then dividing it by the total number of words in the same document. As each document is of different length, and a word can be present in a document many times, by dividing by the total length a more accurate representation of the word's frequency is provided.

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \epsilon d} f_{t',d}} \tag{2.3}$$

2.3.2 Inverse Document Frequency

Inverse Document Frequency is used to counterbalance the value given to terms that appear in many documents - the value of TF may increase disproportionately for certain words. IDF weighs down words that occur very frequently and increases the weight of words that occur rarely.

For example, words that are relevant to certain scenarios could be considered stop-words within their niche. In Mezzanine, the word "open" may appear more frequently in venue descriptions as venues will usually state when they are *open* or what their *opening* times are (NLP preprocessing would change *opening* into *open*). Without IDF, these frequent words will be placed at a higher importance than words that are more descriptive like the genre or type of Venue. therefore, Inverse Document Frequency can be used as a metric to calculate which words hold the most importance. This is calculated using:

$$idf(t,D) = \log \frac{N}{|d \epsilon D : t \epsilon d|} \tag{2.4}$$

Where parameters *t,d* represent the term and the document, *N* is the total number of documents and the denominator is the *number of documents in which term, t, is contained*. The log is taken to soften the scores of words which are frequently seen, instead of the score being proportional to the total count.

2.3.3 TF-IDF Result

The calculated weight for each term in the document is then used in the vector representation of the document. Where each component is a different words TF-IDF score. This process results in a quantitative representation of a qualitative subject - with each document of a corpus having its own point in the same vector space.

In Mezzanine, the descriptions of Venues, Talent, and Events can be represented using vectors, which allows the objects to be used further in Machine Learning concepts and similarity measurements.

2.4 Cosine Similarity

Cosine Similarity is a similarity metric between two vectors. It can be derived using the Euclidean dot product formula *(2.5)* and compares the angle between a pair of vectors in vector space. (Alodadi and Janeja (2015)) As mentioned previously, when calculating the TF-IDF of a document, a vector representation is created with each component being a terms related score. This is the vector that is used in cosine similarity, and thus compares two documents. Normally, this formula on its own returns a result between -1 and 1, but for information retrieval, the result is returned in positive space, as the term frequencies will not be negative. Therefore, results that approach 1 are more similar (comparing a vector to itself results in 1). Conversely, pairs of vectors with a cosine similarity score approaching 0 are dissimilar.

The score is calculated using:

$$\cos(\theta) = \frac{A \cdot B}{\| A \| \| B \|} \tag{2.5}$$

Where *A and B* represent two document vectors in the same vector space. The numerator is the dot-product of the vectors, and the denominator is the multiplied magnitude of the vectors.

In Mezzanine, the comparison is being made between a Talent profile, and a subset of Events. The system uses cosine similarity to recursively compare the Talent's vector representation to an Event, receives the score and adds the score to a data frame along with the Event ID. Once all the scores have been calculated, the data frame can be sorted by cosine similarity score, showing the highest scoring (most similar) first.

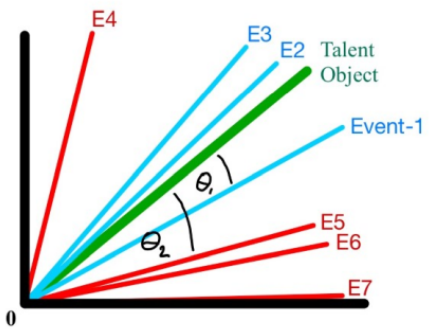


Figure 2.1: *Measuring similarity of Talent and Events*

As seen in *Fig 2.1*, the Talent object is the vector that each Event, *En*, is compared to. With regards to the Cosine Similarity Formula, in each iteration the Talent object is *A*, and each Event is an instance of B. The lines protruding from the origin in red indicates events that are dissimilar to the Talent, where the blue lines indicate events that are more similar.

Using cosine similarity for calculating similarity is a suitable metric for Mezzanine due to the nature of user-defined input. Some users may input relevant, but fewer words in their user

profile. Cosine similarity only uses non-zero vectors, meaning objects that are more sparse are not disadvantaged: meaning the profiles are matched due to the *quality* of the content, not the quantity. When plotted on a multi-dimensional plane, the cosine similarity measures the angle between the objects, not the magnitude - meaning profiles with larger descriptions don't necessarily mean a better match. θ_1 and θ_2 represent the angle between two objects and a talent profile. In this case, θ_1 is more similar than θ_2.

2.5 Clustering

Clustering is a Machine Learning principle that involves analysing a dataset and grouping each item into the same group or *cluster* - where each item in the cluster shares similar properties. The cluster as a whole can then be described, with each item in the cluster being related to this description in some way. therefore, clustering can be used to analyse large datasets and derive 'meaning' behind groups of items. Similarly as mentioned in sections *2.2 and 2.3*, this can be done by quantifying the data seen in qualitative objects, such as text fields, through vectorization. Once each object has a corresponding vector, the objects (and therefore the clusters) can be represented in vector space. Vectors that are closer together share similar attributes, whereas vectors that are far apart share less attributes. therefore, clustering can be used as a method of defining similarity between multiple items within a dataset.

Some objects may contain properties that make them suited to more than one cluster. These objects can be seen on the outskirts of a cluster, close to another cluster. The 'meaning' behind an object can then be approximated, depending on what clusters they are near. For example, a Venue may play rock music and also hip-hop music - cluster analysis would predict the best suited cluster to place the venue in. The venue may be in between two clusters, and may also have nearest neighbours belonging to a different cluster. The phrase *meaning* simply refers to the overall characteristics of a cluster.

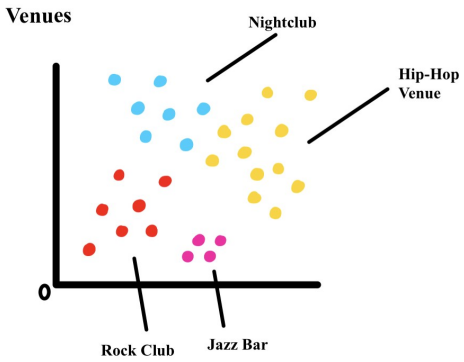


Figure 2.2: *High level representation of Cluster Analysis, where each colour represents a different cluster*

2.5.1 Uses

Clustering has many uses. These include semantic analysis for social media, identifying groups of films or TV-shows for video platforms and creating balanced portfolios for businesses in the finance sector. In this project, cluster analysis techniques are used to analyse Venues and place them close to other Venues that share similar attributes and 'meaning'. The efficiency of clustering increased proportionally to the amount of data available. This is why it is used in social media - thousands or more posts must be analysed and grouped together, while also providing useful information for the needs of the system. Similarly, certain locations like cities, may contain thousands of Venues, making clustering more efficient than iterative Cosine-Similarity checks.

2.5.2 The K-Means Clustering Algorithm

There are multiple clustering algorithms used within Machine Learning, each with their own advantages and disadvantages. After background research was conducted into each type, the naive K-Means clustering Algorithm was chosen. The K-Means clustering algorithm is an unsupervised learning algorithm and aims to separate n items in a dataset into k clusters (Madhulatha (2012)). The value of K is predefined, and represents the number of clusters that will be present. Each cluster is non-overlapping, meaning items can only be assigned to one cluster. The algorithm aims to minimise the sum of distance between each item and the cluster center, known as the clusters *centroid*, therefore keeping the clusters compact and separate from each other. The algorithm is iterative, meaning it will be performed x times, changing the centroid location each time until a best fit is achieved, this process is referred to as Expectation Maximization.

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \parallel x^i - \mu_k \parallel^2 \tag{2.6}$$

The equation used within the k-means clustering algorithm is seen in *Eqn (2.6)* (Likas et al. (2003)) . First J is minimised with respect to w_{ik} and μ_k is fixed. Then J is minimised again with respect to μ_k, with the value of w_{ik} being fixed with the minimum value from step one.

2.5.3 Pros and Cons

K-Means Clustering offers many advantages which makes it a suitable choice for moving forward with development. It is relatively simple to implement compared to other clustering algorithms, with many supported libraries and packages that streamline the development process. K-Means scales to large datasets, but can also be efficient with smaller data-sets - this caters to the nature of Mezzanine, in which one location may have a small number of Venues and another may have a large amount.

While it has many advantages, K-Means also has disadvantages. The algorithm requires the value of K to be predefined. This is the largest downfall of K-Means as larger datasets may conform better to a greater number of clusters - but choosing a larger number for K may mean smaller data-sets are spread too thin between centroids. K-Means clustering may also be affected by the 'Curse of Dimensionality', where the distance between points in the vector space converges to a constant as the dimensionality of the vectors increase. This results in inaccurate predictions of neighbours and therefore inaccurate recommendations. This problem can be mitigated by lowering the dimensionality of the feature data by using Principle Component Analysis (PCA).

2.6 The Cold Start Problem

All recommender systems face a similar problem when new users sign up to a site or service without ever having used it before. Recommender's use the data provided by the user and their previous actions on the site to make predictions on what similar items (products, films, venues etc.) they may like, while simultaneously deciding which items they may not like. However, if a user has never interacted with a site before then the recommender does not have the necessary data to provide accurate predictions. This is referred to as the cold start problem (Lam et al. (2008)).

Recommender systems such as Netflix or Amazon Prime prompt a new user to select some films and TV shows on registering with the site. Each of these films creates a background for the users profile, allowing the system to recommend items that match. If a user refuses to select some items, then the recommender promotes shows and films of various categories and analyses the interactions, with each interaction being categorised into positive or negative class.

In Mezzanine, a tag-based recommender is deployed, where a user adds tags to their profile on creation. A similar process as Netflix is used, where a user is given a selection of descriptive keywords, relevant to both Talent and Venues, and asked to select the ones most relevant to their venue or personality. Behind each keyword is a dataset containing multiple tags for each keyword. Each user is not allowed to create an account without selecting at least one keyword. This is not a complete solution to the cold start problem, but encouraging a user to select multiple keywords results in a more descriptive TF-IDF Vector, that then results in better recommendations between Talent and Venue objects.

2.7 Summary

This chapter reviewed the technological concepts that are used to recommend Venues to Talent, with reference to the needs of the system. Linguistic concepts like stemming and stop-words were also discussed to understand the pre-processing methods and why they are required to increase the efficiency of the recommender that was built.

3 Requirements Analysis

This chapter will discuss the functional and non-functional requirements of the project, the methods taken to extract these requirements and the different iterations of Mezzanine: from the Minimal Viable Product (MVP) to the current version that is implemented. The majority of requirements were gathered at the beginning of the planning phase, however as development proceeded, more requirements were added.

3.1 Requirement Elicitation

When discovering the requirements of Mezzanine, user scenarios were created to understand how different types of users would react with the system, and what features would be needed in order to create the best experience. Each scenario was then turned into user stories, which allowed for a high-level view of the functional and non-functional requirements (Lucassen et al. (2016)).

3.1.1 User Scenarios

Various user scenarios were created to depict how each user type interacted with the system. These users were a venue owner, a musician and a recreational performer. Each scenario was then reduced into multiple user stories and then into functional and non-functional requirements.

- *Richard* is the owner of a small pub. Due to the recent coronavirus pandemic, he has had to close his doors on-and-off for 2 years. The pub operates as a restaurant up until 8pm, and he would like a musician to come that can play family-friendly music from 6pm until 7pm. Mark would also like a band to come and play between 9:30pm and 11pm, that will play up-beat music to encourage dancing and socialising. However, he is not sure how to advertise the position. The pub operated a single social media account, but is only used to promote new menu items and does not have a large following.
- *Amanda* is the owner of a nightclub and is looking for a new DJ to fill a spot every Wednesday. The nightclub has a large following over several social media accounts where they receive many messages each day. In order to advertise the role, she put a post on each social media page - asking suitable DJs to get in touch with a message containing their name, age and examples of their work. Within a few hours, she has over 100 applicants and cannot decide which DJ is the best suited.
- *Jackie* is a full time flight-attendant for a large airline and has a passion for music. She used to perform as a singer in a band, however had to stop as her career involved her moving around a lot. When she operates on long-haul trips, herself and her crew mates receive a few days off before departing the location again - this allows her to spend time in new places all over the world. When back in her home city she performs at her local bar and receives a wage for doing so. She would like to do this more often and has seen many potential places to perform on her lay-over stops. However due to not being from the area, Jackie does not know what venues are offering work, and if any venues are requiring an artist that performs what she does.

3.1.2 User Stories

User types and user stories can be derived from the user scenarios described in *3.1.1*. The results were as follows:

User

- *As a User* of the system, I want to be able to register for an account, so that I can interact with other Users of the system.
- *As a User* of the system, I want to be able to login and logout of the system.
- *As a User* of the system, I want to be able to update my account details, so that my profile is the most relevant.
- *As a User* of the system, I want to be able to securely delete my account, so that my details are not on the site for longer than I require.
- *As a User* of the system, I want to be able to view other Users, to find out information about them.
- *As a User* of the system, I want to be able to view Events, to find out more information and if I want to attend them or not.
- *As a User* of the system, I want to be able to create a descriptive profile, so that other users can find information about me.

Venue

- *As a Venue Owner* on the system, I want to be able to create an Event, so that I can advertise the event to users.
- *As a Venue Owner* on the system, I want to be able to receive applications for Events, so that I can fill open positions.
- *As a Venue Owner* on the system, I want my upcoming Events to be present on my profile, so that users can see all upcoming Events.

Talent

- *As Talent* on the system, I want to be able to search different locations for work, so that I apply to jobs that I am able to travel too.
- *As Talent* on the system, I want to be recommended jobs, so that I do not apply to Venues that are not looking for what I offer.
- *As Talent* on the system, I want to be given the chance to respond to job offers, so that I am not double-booked.
- *As Talent* on the system, I want my profile to show the events that I am working on, so that other users can see when I am next performing.

Admin

- *As an Administrator* of the system, I want to be able to view all Venues, Talent, and Event objects, so that I can have a full overview of the site.

3.2 Functional Requirements

Once the requirements were gathered, functional requirements for each user type could be identified and non-functional requirements for the system could also be identified. When creating the list of requirements, the *MoSCoW Analysis* approach was taken (Kuhn (2009)). This approach involves categorising each requirement into four types:

- *M - Must Have.* These are requirements that are crucial for the successful functionality of the system and have to be implemented to meet the requirements of the Minimum Viable Product (MVP).

- *S - Should Have.* These are requirements that should be developed if time constraints allow, but are not crucial to delivery of the MVP.
- *C - Could Have.* These are requirements that the system could potentially offer to users, however are not a priority for successful development and deployment of the system.
- *W - Won't Have.* These are requirements that were conceptualised, but were decided against and will not be developed for Mezzanine.

As in *3.1.1*, the functional requirements of Mezzanine have been categorised based on which user type will be most affected by the requirement. As there is profile functionality that is valid for both Venue and Talent objects, this has been placed under the *User Profile* subheading. There is also functionality that is carried out internally by the system that is interacted with by users, but not directly affected as a result of their actions - these requirements are listed under the *System* subheading.

User

1. **MH** Users must be able to register for a new account on the system.
2. **MH** Users must be able to login to the system.
3. **MH** Users must be able to update and delete their account.
4. **MH** Users must be able to view Venues, Event and Talent information
5. **SH** Users must be able to search for Venues without signing into their account
6. **CH** Users could upload images to their accounts.
7. **WH** Users were originally going to be able to make posts and comments, however this requirement was disbanded as it is not crucial to the development of the proof-of-concept of Mezzanine.

Talent

8. **MH** Talent must be able to search for jobs by location.
9. **MH** Talent must be able to apply for open jobs.
10. **MH** Talent must be able to receive and view offers to work on jobs that they have applied for.
11. **MH** Talent must be able to accept or decline offers to work on jobs.

Venue

12. **MH** Venues must be able to create Events.
13. **MH** Venues must be able to view applications made by Talent to work at Events.
14. **MH** Venues must be able to accept or deny applications made by Talent.

System

15. **MH** The system must analyse Talent profiles and make Event recommendations based on information gathered from each object.
16. **MH** The system must facilitate the acceptance of job offers between Talent and Venues.
17. **CH** Users could receive email confirmations when certain functionality is complete. E.g. Creating an account, applying for events, accepting offers.
18. **WH** The system was potentially going to offer payment methods between the Talent and Venue objects, however this requirement was disbanded as it is not crucial to the development of the proof-of-concept of Mezzanine.

3.3 Non-Functional Requirements

As well as functional requirements, a list of non-functional requirements (NFR's) was also gathered. A non-functional requirement refers to criteria that is used to judge the performance and operation of the system, however are not critical for the system to work as expected: "(Non functional requirements) Describe the non behavioral aspects of a system, capturing the properties and constraints under." (Glinz (2007)). which a system must operate By defining the NFR's, the bounds are set as to what is a needed to deliver a system that provides a suitable user experience. Due to Mezzanine revolving around lots of user interaction, the non-functional requirements are needed to ensure users find the site easy to use which encourages them to come back.

19. **- MH** The system must offer visualisation of where Event locations are using a map.
20. **- MH** The system must be natural to use, with minimum training needed to operate the systems functionality.
21. **- SH** The system should be able to work on different display sizes.
22. **- SH** The system should have a low latency and deliver new page requests to the user quickly.
23. **- CH** The front-end of the system could only use colour-blind-safe colours to improve accessibility.

3.4 Summary

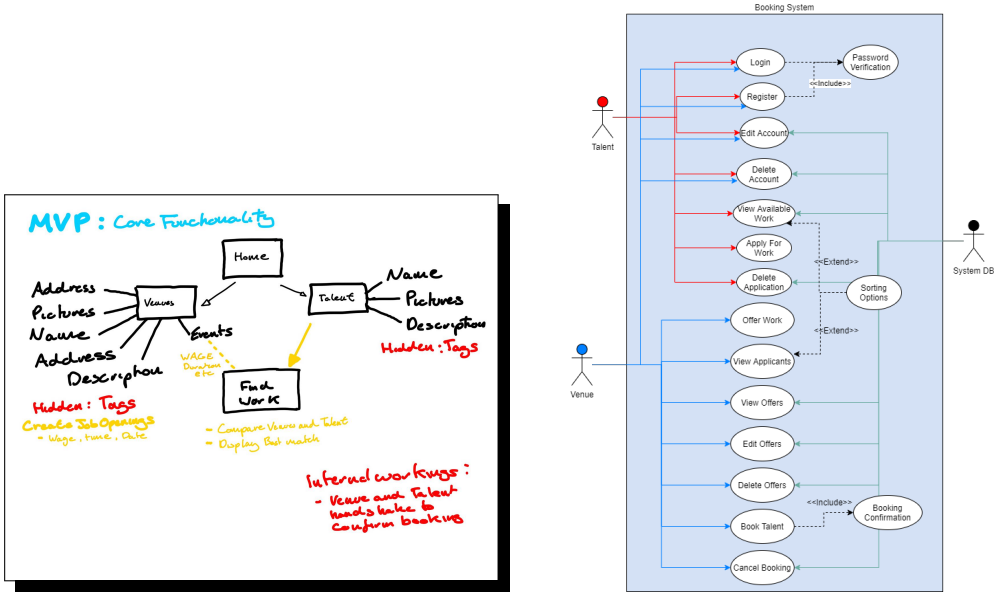
This chapter began by discussing the ways the requirements were gathered for Mezzanine. This highlighted the use of user scenarios and the process of turning these scenarios into functional and non-functional requirements of the system.

4 Design

This chapter will explain the design of Mezzanine, beginning with the idea generation process, system architecture and then the user interface. This chapter will also describe the 'handshake' between the Venue and Talent - a core feature of the system with regards to the business logic. Continuing from this, the user-interface design process will be discussed. The chapter will then conclude with an overview of the core tools and technologies used within the front and back ends of the system. In each section, the decisions made will be correlated with the requirements that they satisfy.

4.1 Initial Design Process

The solution generation phase for the requirements involved continuous rough prototyping for each requirement before a final decision was made on how to integrate the solution into the system. The process mainly happened in 3 stages: first the problem was addressed through the use of 'issues' derived from the requirements, then sketches were drawn out on note-taking software of each idea before finally selecting a final solution and creating a UML diagram based on that. Prototypes included small sketches to visualise ideas, back-end flow of the system and many different layouts for the front-end. This was due to the number of connecting pages within Mezzanine, the number of interactions between the user types and the requirements for an easy to use system. These factors resulted in many ways to take the site from A to B.



(a) Minimum Viable Product sketch

(b) Initial UML Use Case Diagram of the site

Figure 4.1: *Two iterations of the system development, including the initial MVP and the next site with additional features. See appendix [x] for more iterations.*

In the initial design stage, UML diagrams were created including Use Case, Activity, Entity Relationship, Sequence and Class diagrams. These diagrams allowed for a clear understanding of the back-end flow of the system before development had begun. This made finding the right technology to use for implementation easier, as the research of the system had been completed, meaning the needs of the system were understood and the right framework could be selected based on these needs.

The MVP requirements of the system were visualised with the use of drawing software, highlighting the relationship between user types at a high level before analysing the internal workings of the site. The MVP drawing created offered a similar functionality to a Use Case diagram, except allowed the visualisation of how user types worked together, and how each iteration of the system expands in size due to additional features being added.

4.2 System Architecture

The back-end functionality of Mezzanine can be broken up into three main parts: The interactivity between the Venue and Talent objects, the analysis of similarity between Venue and Talent objects to derive a list of recommended Venues and the external API calls used to build the Maps used throughout the site.

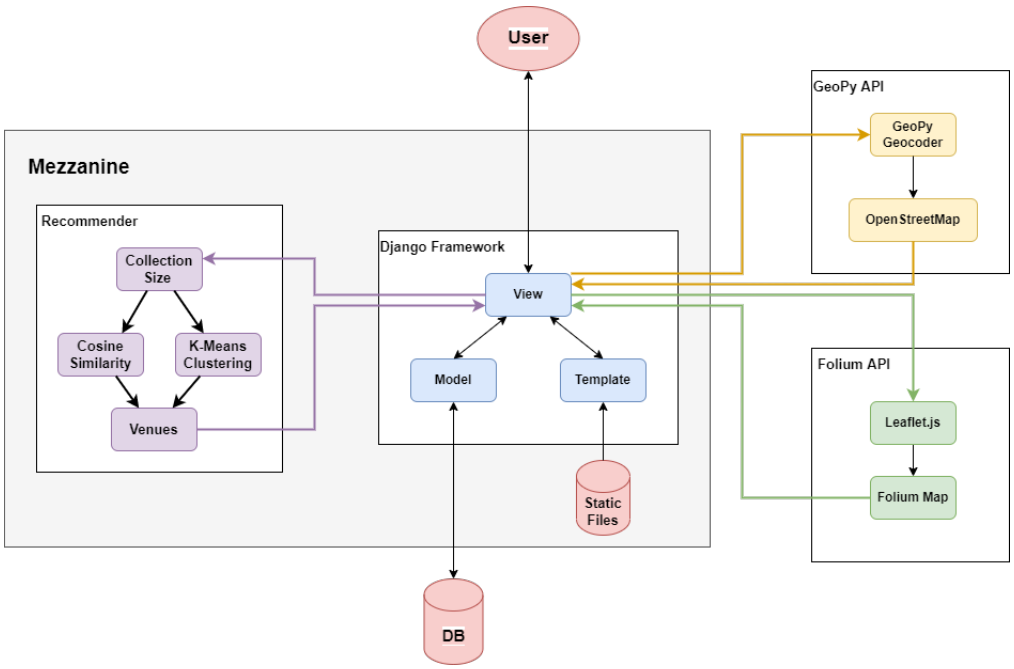


Figure 4.2: *System Architecture diagram of Mezzanine. Showing the interactivity of the Django framework, external API's and the recommender code at a high-level.*

Mezzanine's core is built using the python web development framework Django. Django utilises a Model-View-Template (MVT) approach to web development that aims to dynamically create web pages based on the data stored in the database. The DB is represented in the system by the Model. Each model represents a table in the DB, with each attribute of the model being a field of the table. A tables rows can be sorted, searched, added to, updated and deleted from through interactions with the model - without the need to provide any SQL. The model is interacted in this way through the use of the View. The view accesses the model, and if needed, uses data read

to build the Template - the final web page that is delivered back to the user.

As seen in *Fig. 4.2*, the View is at the center of the system - analysing requests by the user and responding accordingly. The information retrieval and recommending logic needed to identify venues is stored in a different directory within the system. This is due to the frequent use of the logic and the size of each recommending component - some components using hundreds of lines of code. Storing the logic in a separate directory helps to modulate the code, conforming to the DRY (Don't Repeat Yourself) programming best practice and providing the most readable code possible.

The view accesses this directory and calls upon the recommender logic needed for a request. This is seen in *Fig 4.2*, with *Recommender* being the system box where a high-level representation of the recommender is shown. Within this component there is *Collection Size* which represents the count of the venues available to a location that the user has searched. For example, a small village may have 5 venues whereas a large city may have 500. Depending on the collection size, the sub-component feeds into either *Cosine Similarity* or *K-Means Clustering*. As discussed in *Background*, Clustering is increasingly efficient with large datasets, and due to the nature of the system, there will not always be many venues in each location. When this is the case, the cosine similarity between the Talent and Venues is taken and the recommendations are derived that way. Each of these sub-components feed into *Venues*, which represents the list of venues the system has chosen to recommend to the user. These venues are then passed back to the view to be analysed further, or combined with a Template to be returned to the user.

When creating templates with the array of Venues, the View calls upon external APIs to gather the location information and build the map associated with the location - representing each venue as a marker.

4.2.1 Venue - Talent Handshake

Mezzanine must also offer the booking functionality between Venue and Talent as this is a must have requirement - *reference*. This is the core business logic associated with the system as booking Talent for an Event by a Venue is not the same as other booking systems such as: booking a table at a restaurant or booking your seat on an airplane. A Venue may take a long time to respond to a Talents application for an Event, and in some cases may not respond at all. A Talents application may also be unsuccessful. For these reasons, it is not in the Talents best interest to be able to apply for only one Event at a time - Talent must be able to apply to many Events, even if the times overlap, meaning if they are not successful in one application, they still have a chance of a booking for another.

Venues must also be able to view applications, and decide which applicant is best for the Event. As Talent may apply for more than one position, a Talent may be successful in many applications and have to decide between offers - meaning a Venue may not get their first pick for Talent. This results in a back-and-fourth communication between the Talent and Venue users where the booking process requires many stages of interaction before a booking can be finalised. This can be seen in Fig 4.3, where a sequence diagram shows the back-and-fourth interaction. The solution is to offer different stages of acceptance before a finalised booking can be made.

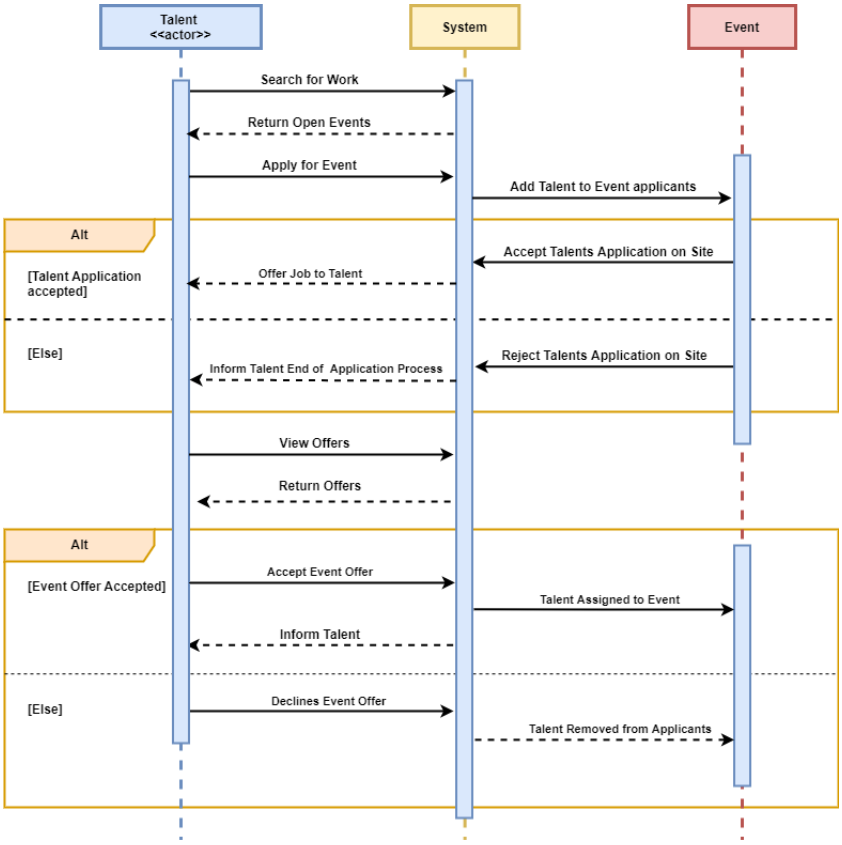


Figure 4.3: *Sequence Diagram highlighting the back and forth interaction between Talent and an Event, with the System representing Mezzanine processing the interactions.*

4.3 User Interface

Due to Mezzanine being a heavily user-interaction based system, the front end design process was an integral part of the development that involved lots of research and problem solving. The concept of Mezzanine, as mentioned in the aims section 1.2, is to be *"modern and responsive with an emphasis on ease of use"*. This means that a potential Talent or Venue owner should be able to view the site and have an established idea of how the site works just from the information given. This was proven to be difficult, with many ideas being generated and disfavoured for others.

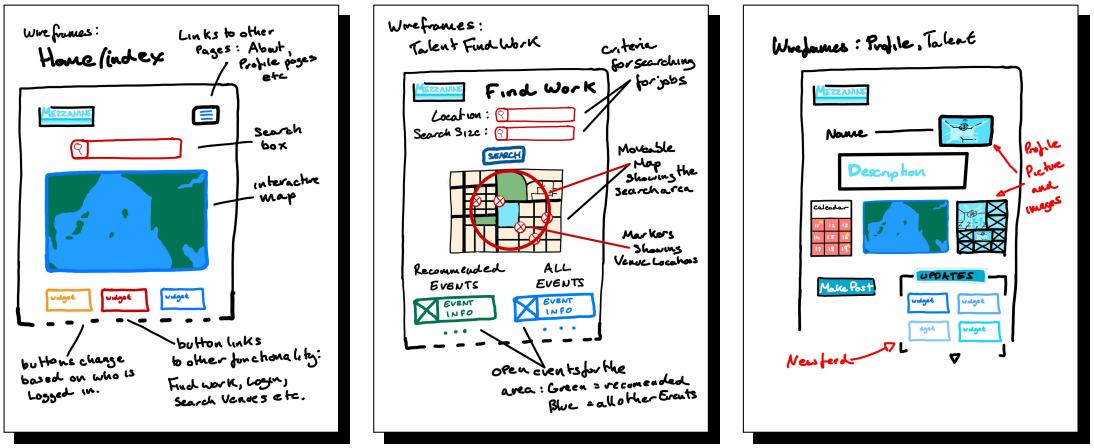
4.3.1 System and Account Features

Talent and Venue users have features that are similar across both user types. These include the registration process, profile pages and account updating and deleting functionality.

Mezzanine's Home Page Due to the location filtering functionality of Mezzanine, it was decided that an interactive map should be present on the home screen, with the option to search for Venues. This functionality was to be open for use by any user type - even non-users of the site. Because of this, it made sense to have it as the welcoming page, and not behind a login wall. This decision satisfied requirement 4. In the idea generation stage of Mezzanine, a third user type was discussed. Where a user could register without having to conform to either a talent or venue, but just as a *standard* user for the purpose of interacting with Venues and Talent profiles. By having the mapping and searching functionality on the home screen, these same features are

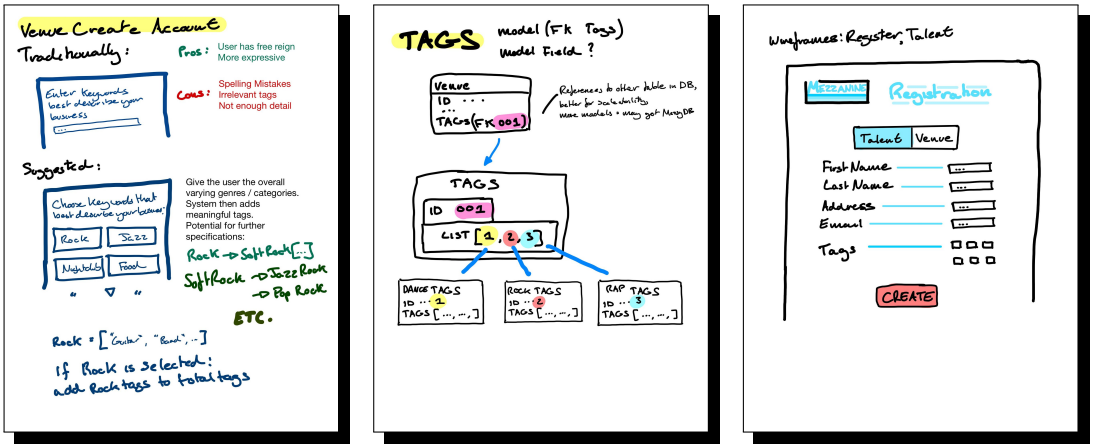
acquired without the need for another user-type.

Profile Pages Each users profile must also match the requirements of [x,y,z] in order to have a descriptive and easy to understand profile. Each profile page shares similar attributes: name and description. Venue accounts show additional address information since Talent and non-users may browse venues and require this knowledge. The keyword tags are also shown on each account to represent the accounts interests/themes. Both Venue and Talent accounts show upcoming events from the hosting/performing at perspectives.



(a) Home Page Wire-frame *(b) Find Work Wire-frame* *(c) Users profile Wire-frame*

Figure 4.4: *A subset of the initial conceptual wireframes with annotations of each component.*



(a) Key-words being shown to the user instead of a text-input of tags *(b) Theorising how the tags will link into the back-end of the system* *(c) Sketching ideas for the front-end of the system*

Figure 4.5: *Tagging functionality concept generation and front-end design prototype*

Registration A user may use the site to browse locations of Venues, view Events and therefore also view Talent profiles. If a user decides to register, they are taken to the pre-registration page where they are greeted with information on each user type. In the *Background* section, the *cold start* problem is discussed. This is a large problem with recommending systems, and finding ways on how to mitigate this problem in the design was a main focus within the idea generation phase. As the system uses TF-IDF to quantify the user types, a tag-based system was designed

originally where a Talent/Venue inputs individual words that describe them. This posed some issues: a user may miss-spell words, not provide enough detail or ignore the tag. These problems were mitigated by designing a select-box style approach to the tag system. This involved a user selecting words that best fit their profile - which was then linked to a dataset in the database containing tags based on that word.

4.3.2 Venue Functionality - Event Creation

Event hosting by Venues is a core requirement as seen in *insert requirement*. When designing a system that allows Events to be made, the business logic must be catered for which refers to requirements *insert requirements*. It is common for Venues to host events that cater to different genres or themes than the venue itself. For example, it is common for a nightclub to have different themes throughout the week, i.e. Thursday - hip-hop night, Friday - Electronic Dance Music, Friday - Rock and Pop Music. Mezzanine has been designed to allow each Event their own description and tags. When passed into the recommender, this data is processed and combined with the Venue data to create a more descriptive representation of the event. The tag layout of the event creation is similar to the registration of profiles as seen in *Fig. 4.4 (c)*, where the keywords are presented as check boxes.

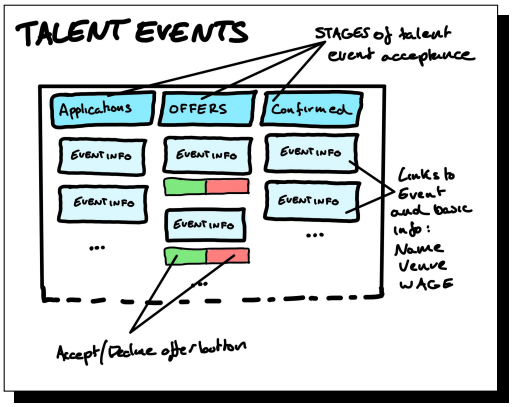
4.3.3 Talent Functionality - Applying for Events

When applying for Events, the Talent user must first sign into their account, which changes the home page to show a 'Find Work' button. This adds a link to the recommender functionality of the system, which follows the steps as described in the System Architecture section. As seen in *Fig. 4.3 (b)*, the user is presented with the map of the location they have searched - if the user navigates to the page without inputting a location, then the location stored in the users profile should be the default. The page should conform to requirement *insert here*, that the site should be efficient to use and understand. Because of this, the page should be properly labeled and laid out accordingly. The recommended events are presented in green, whereas all other open events are presented in blue. Each event should have a corresponding marker on the map that shows the location of the Venue where the event is hosted.

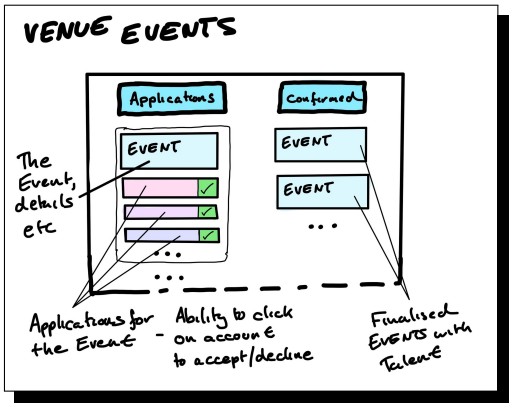
As described in the background, the system should recommend Events using either the Cosine-Similarity algorithm, or through cluster analysis using the K-Means Algorithm. Where cosine-sim measures the size of the angle from origin between the two points in vector space, and where K-Means analyses the euclidean distance between the Talent coordinate and Event Coordinates.

4.3.4 Talent and Venue Handshake

The design for the handshake between the talent and venue, as seen in *Fig. 4.3*, must incorporate each stage of the process. Business logic from both sides must also be incorporated. For example, a Talent should be able to view all of their applications, offers and confirmed events - whereas a Venue must be able to see all open events, applications for each one, and Events where the Talent has been confirmed. The solution is to offer each user a web-page within their account which highlights all of the stages of the process, and offers interaction of the applications. By housing all stages and required information on the one page, requirements *insert requirements here* are fulfilled.



(a) Key-words being shown to the user instead of a text-input of tags



(b) Key-words being shown to the user instead of a text-input of tags

Figure 4.6: *Tagging functionality concept generation and front-end design prototype*

4.4 Tools and Technologies

4.4.1 Back-End

Django The Django framework was chosen due to a multiple reasons: There is an emphasis on fast deployment of websites - this helped with each sprint iteration as changes could be made quick and checked immediately [1]. When researching different technologies that were needed to fulfill the requirements, many of the potential libraries and API's that were considered offered Python compatibility. By using a back-end in Python, the integration of these resources would be more seamless than another frameworks using different languages.

M-V-T Architecture As mentioned in this chapter, Django utilises the Model View Template architecture for web development. This framework offers a logical split of tasks between the front-end, logic and data storage. It creates an efficient solution to building dynamic web pages compared to other frameworks. The design of the system indicates the site will be mainly dynamic, making MVT the best choice.

Resources Django has a large and open source community, meaning plenty of resources were available when issues and bugs were raised. Online videos, forums and the Django documentation smoothed the learning curve and allowed a quick transition into implementation. Forms, account registration and redirecting URLs are all built in with Django, meaning extremely fast development for core user functionality - therefore more time could be spent building the matching functionality.

Admin The Django framework comes with an inbuilt admin panel for the system. It includes CRUD functionality with the database in a high-level user interface. This inbuilt feature can also be added to and edited in the Django code base, allowing for creation of bespoke admin panels. For example, the way the models are visualised can be changed and bespoke for each type. This functionality satisfies requirements [x,y,z] for an interactive admin interface.

Database The framework comes with a server-side SQLite database that is used for storing the model data. Due to Mezzanine being a proof-of-concept creation, the inbuilt database was light enough to handle the requirements of the system. Django also creates its own SQL based on commands passed to the model from the view. This increases efficiency when writing code used for accessing the database and makes for more readable code.

[1] https://docs.djangoproject.com/en/4.0/

4.4.2 Front-End

Bootstrap The design of the front end was created through the use of bootstrap and Django's inbuilt template variables. Together, they offered the user an efficient and easy to use system. Bootstrap is used inline with HTML, removing the need for multiple style sheets and greatly increasing the amount of options available to use [2]. Because of this, the front-end can be created at the same time as the back-end effectively.

4.4.3 Data-Science Libraries

Multiple libraries were used to aid in the creation of the recommender side of the system. These primarily helped in the NLP and TF-IDF vectorisation of the Venue, Talent and Event objects. Cosine similarity and k-means clustering libraries were also used to increase efficiency when performing the tasks of clustering and finding the most similar venue.

NLTK The NLTK library for python, which stands for Natural Language Toolkit, is used in the NLP component of the system. It contains multiple features, including a large corpus containing stop-words of each language. It also contains a tokenize method, used for turning a string into its separate 'tokens'[3].

Scikit-Learn The Scikit-Learn library is Machine Learning library that contains multiple methods and objects [4]. This streamlines the software engineering process for machine-learning projects by providing the core features, such as K-Means Clustering Objects, TF-IDF Vectorizers, Cosine-Similarity methods.

4.4.4 Maps and Location

GeoPy GeoPy is a an external API that provided coordinate data needed to represent a location through geocoding [5]. GeoPy is not the direct service provider, but rather acts as a link to different external mapping APIs. *OpenStreetMap Nominatim* is a free geocoder service, provided by the GeoPy API, that is used for retrieving the coordinate data. [Appendix]

Folium Folium is a python library that is used to connect to the JavaScript library, Leaflet.js. Folium builds on the location data received from GeoPy (or other mediums) to create an interactive map. It is a server-side service which offers many additional mapping features such as markers, shapes and different map types - that can be used for visualising Events and Venues [6].

4.5 Summary

This chapter provided an overview of the design of Mezzanine. Starting with the idea-generation process, followed on by the back-end and front-end designs of the system. The chapter was concluded with an overview of the Tools and Technologies that were researched and chosen to start the implementation.

[2] https://getbootstrap.com/docs/5.1/getting-started/introduction/
[3] https://www.nltk.org/
[4] https://scikit-learn.org/stable/index.html
[5] https://pythonsimplified.com/geocoding-in-python-using-geopy/
[6] https://python-visualization.github.io/folium/

5 Implementation

This chapter will review the implementation process of the core functionality discussed in the *Background*, *Requirements* and *Design* sections. The recommending process will be explained in detail, making reference to the logic and sub-components used in each stage. The implementation of the creation and population of interactive Maps will also be discussed further. The section will conclude with an overview of the implemented front end and how technologies discussed in the Design chapter were used.

5.1 Software Engineering Process

Throughout the development, software engineering practices were adhered to. These were introduced to increase efficiency, consistency, and aid in the development of the system. Version control was used throughout the development, updating the code base regularly. Feature branching was used, where large features of the site were branched into their own version for development, then merged with the main. Issues were used within the time-log. The issues were listed at the start of each week, then the development for them was undertaken that week, with any incomplete issues moving onto the next weeks list. This created the concept of solo sprints, where each week set tasks are aimed for completion. As discussed in Hesenius et al. (2019), processes for machine learning related tasks were adhered to, in order to receive the optimum outcome.

5.2 Recommender Architecture

In *Fig. 4.2*, a high level overview of the recommender was shown within the System Architecture of Mezzanine. This subsection, *5.1, Recommender Architecture*, aims to analyse the implemented process for recommending Events to Talent; starting with the total set of Venues, the recommending process, and then returning the web page to the user.

The recommender system of Mezzanine contains multiple stages that directly relate to the research carried out and discussed in the *Background* and *Design* sections. These stages are:

1. Natural Language Processing
2. Cosine Similarity
3. K-Means Clustering
4. Map Creation and Population

When a user searches by location, the system follows the process seen in *Fig 5.2*. First the location and surrounding area is retrieved using GeoPy. GeoPy is an external API that receives a location string (Any combination of City, Town, Postcode) and returns the locations' longitude and latitude coordinates. GeoPy is described more in *Section 5.4*. Once the area is calculated, Venues within this area, and all Events they are hosting, are retrieved from the database. Due to the nature of Mezzanine, some locations may not contain any Venues, this is discussed further in the *Background* and *Design* sections. If this is the outcome, then no recommendation has to be made.

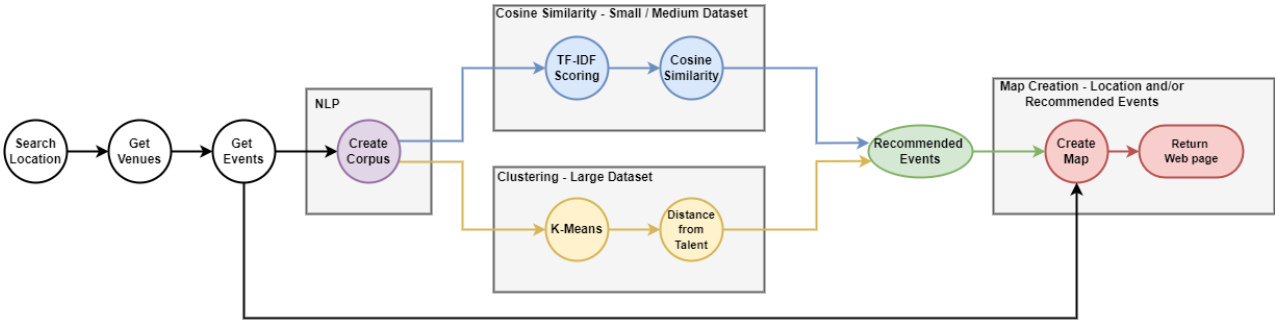


Figure 5.1: *The process used by the Mezzanine recommender in order to return Events. Starting with the location, and ending with the returned webpage to the User. A lower level representation of the Recommender component of the system architecture diagram seen in Fig. 4.2.*

The system therefore bypasses the recommendation stage and builds the map of the location, without the population of markers.

If the location returns >0 Events, then the pre-processing stage of the recommender can be completed. As discussed in *Background*, before vectorising a Talent, Venue Profile or Event, preprocessing must be completed to remove the 'noise' from the data. Each objects string representation is then added to a corpus. This is done using Natural Language Processing libraries and techniques, discussed further in *Section 5.3.1*.

The next stage involves analysing the corpus and deciding which recommender to select. In *Background*, the efficiency of clustering was discussed, and how cluster analysis can be used to make recommendations. Cosine similarity was also discussed, and how this method of recommendation is efficient with small-to-medium length datasets was analysed. Due to the nature of Mezzanine, locations may return a a large amount of Venues and Events (hundreds, thousands, or more) - when this is the outcome, clustering is utilised to make the recommendation. The same system may also return a small dataset containing 1 to N Venues, this smaller dataset may not be efficiently used with clustering, in which case the Cosine Similarity method of recommendation is used.

These stages analyse the data available and choose an appropriate recommending technique. Once the recommender has returned the subset of Events, the map is created and populated, before being returned by the View to the user.

5.3 Recommender Algorithm

The matching algorithm contains 3 main components- NLP, Clustering, and Cosine-Similarity - mentioned at a high level in section *5.2*. In this section, *5.3*, these components will be broken down and the implementation of them will be discussed.

5.3.1 Natural Language Processing

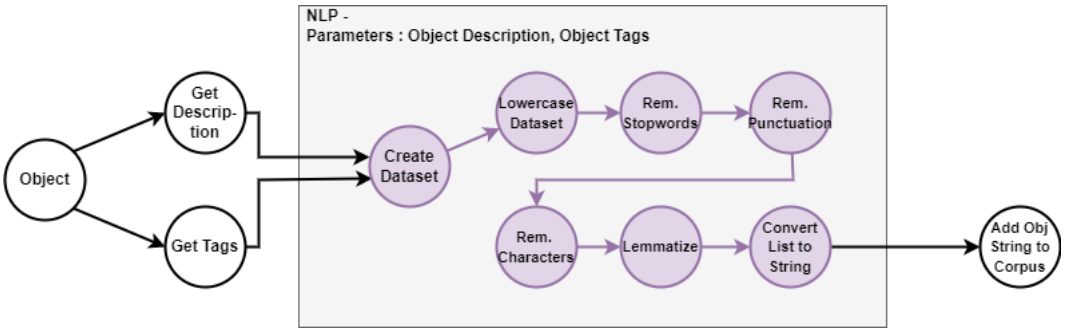


Figure 5.2: *The Natural Language Processing used in Mezzanine to take each object and add it to a corpus to be used in recommending.*

Mezzanine utilises natural language processing techniques to build a pre-processing component which conforms to industry recommendations for removing the 'noise' from textual data. The component receives an objects description and an objects tags as parameters. In this case, and all other cases, 'object' refers to instances of Talent, Venue and Event objects. The component then feeds the parameters through multiple sub-components, which each alter the data before returning a string representation of the object. This string can then be used in TF-IDF Vectorization, the first step used in Cosine-Similarity and K-Means clustering. *Fig. 5.2* highlights the flow of data of this process. Starting with the object and returning the pre-processed string representation.

The description and tags are passed into the NLP component as parameters then combined into one list object. Further operations, like; lower-casing, the removal of stop-words/punctuation/characters and lemmatizing are more easily applied to items within a list due to the types iterative capabilities, compared to words in a string. The stop-word corpus is supplied by the NLTK Library, mentioned in section 4.4.3, which is used to identify each stop-word in the object list and remove it. NLTK also provides the lemmatize functionality used on each item in the list.

The final list is then converted back to a string, separating each item in the list with a space, and then added to the corpus. The starting object has now been pre-processed, and now has a corresponding string representation that can be used in TF-IDF Vectorisatoin.

5.3.2 Cosine-Similarity Recommender

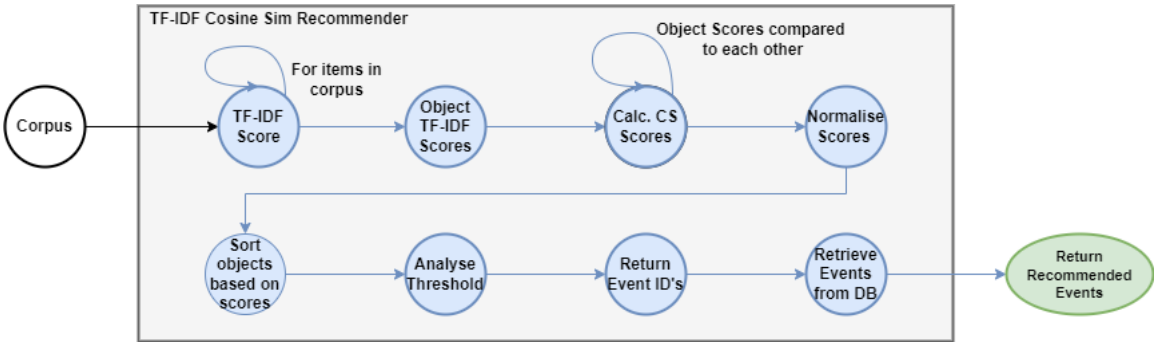


Figure 5.3: *The process of recommending Events to Talent using TF-IDF and Cosine Similarity. Starting with the corpus obtained in Fig. 5.2 and ending with the subset of Events.*

The cosine-similarity based recommender is a crucial component in Mezzanine, used for handling small to medium size datasets, whose length falls below the threshold for efficient clustering analysis to be used. The similarity process is the same as discussed in the *Background* section - where the angle between two vectors is a measure of the similarity. The paired scores return between 0 and 1 - where a score of 1 is an objects comparison to itself. In Mezzanine, the paired score refers to the cosine similarity between a Talent vector and an Event vector.

Fig. 5.3 highlights the flow of data of this process. Starting with the corpus created by preprocessing seen in section *5.3.1*, and returning a subset of events - the recommended events.

The corpus contains the string representation for each object in a given location - searched for by the user. The first step in recommending events is to calculate the TF-IDF score for each item in the corpus. The SKLearn library provides a TF-IDF Vectorizer object, which includes a *fit transform* method that is used in calculating the TF-IDF score of an object.

The Talent and Event vector can be calculated this way. Once calculated, the vectors can be plotted in vector space, and the cosine similarity can be calculated as a measure of similarity. In the implementation, the Cosine-Similarity function was originally created as a separate function, using the same formula found in the *Background*, but upon further research the method provided by the NLTK Library was chosen instead. This method returns a matrix array. With each row being an object, and each column being the objects cosine score compared to the other documents in the corpus. In the implementation, this matrix is transformed into a Pandas dataframe.

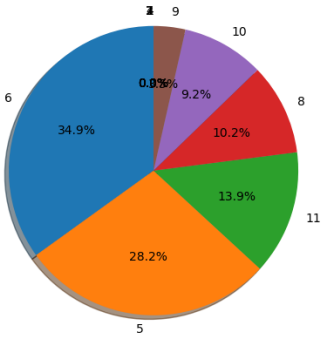


Figure 5.4: *The weight of the similarity as a percentage after normalisation between Talent and 10 Venues.*

The next stage in the recommender is to decide which subset of venues to recommend. The first concept was to recommend the top N venues - however, in doing so lacked completeness. For example, if the scores of the Events at N -1 and N were 0.6 and 0.52 respectively, and the scores at event N +1 and N +2 were 0.519 and 0.518 respectively - then the scores after the threshold of N should be included as they are almost identical to the cosine-score at N. This problem was mitigated through the use of normalisation and analysing the threshold thereafter. Each Event cosine score was normalised between the value of 0 and 1. This allowed for easier analysis of similarity as it showed the true weight of each score. In some occasions a score of 0.3 may hold more weight than a score of 0.5 due to differing subset sizes. therefore, Normalisation allows for dynamic recommendations. As seen in *Fig. 5.4*, the normalised scores of Venue with IDs 6 (Blue) and 5 (Orange) show more dominance over the other Venues returned by the system. From here the top 40% inclusive objects were recommended. This threshold value changed multiple times during the implementation process, and is further discussed as a limitation of the project in the *Evaluation*.

Once this subset of Events has been identified using the process described, the IDs are returned in their order of recommendation via a Pandas data frame. The IDs are then used to retrieve the Event objects from the database and returned to the user. The recommendation process of Events using TF-IDF, Cosine Similarity and normalisation is now complete.

5.3.3 K-Means Recommender

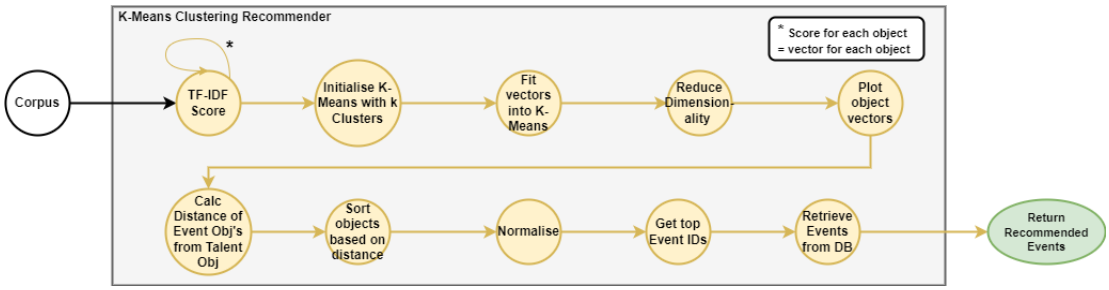


Figure 5.5: *The process of recommending Events through cluster analysis using K-Means clustering algorithm. Starting with the corpus obtained in Fig. 5.2 and ending with the recommended Events*

The cluster analysis method of recommendation is the second way in which Mezzanine analyses Event datasets to make recommendations. Clustering is used as the recommender of choice in Mezzanine for larger datasets, outwith the threshold for suitable cosine similarity recommendation. The process involves similar set-up steps as seen in *5.3.2*, such as vectorization of Events and Talent objects. However, unlike cosine-similarity, the similarity is not based on the size of the angle between Event and Talent vectors, but rather the euclidean distance between the Talent and Event vectors within vector space. A k-means algorithm object is created, supplied by the SKLearn library, then the corpus of objects are vectorized and fit - resulting in their vector-space coordinates. As mentioned in the *Background*, the smaller the distance between items, the more similar they are. *Fig. 5.5* highlights the flow of data through the sub components found in Mezzanines clustering process, using the K-Means clustering algorithm.

The corpus used in the K-Means clustering process is retrieved from the NLP component of the recommender system and contains the string representations of the Talent object and all Event objects. The first step in K-Means is the same as the first step in the cosine-similarity recommender - where each object must be turned into its vector representation through TF-IDF vectorization. This is done using the *TF-IDF Vectorizer* and *fit transform* methods from the SKLearn library. *Fit transform* receives the corpus as a parameter and returns each documents vector.

The SKLearn is also used for the main functionality of K-Means clustering by importing a *KMeans* object. The model takes 4 parameters: the value for k (the number of clusters), the clusters initial starting points, the number of iterations of the algorithm per object call, and the number of times the algorithm will be ran with different centroid seeds/starting points. The k-means skeleton is now created and can be populated with the vector representation of each Event in the corpus.

The most efficient value for K can be found by using the 'elbow method'. This method involves iterating through different values for K and calculating the *cost* for each cluster. As the cluster count increases, the items per cluster decreases - with each of these items being closer to the centroid. The elbow point refers to the number of K that achieves the best balance of items per cluster. In implementation, the value of 3 was found to be the best fit.

As mentioned in the introductory paragraph of this section, the recommendation is based on the euclidean distance between the Talent vector and Event vectors. The euclidean distance refers to shortest distance between two points, and in the case of clustering, the Events closest to Talent are the most similar Events. The closest Events are calculated and their IDs added to a dataframe. This data frame is then sorted, with the closest Events appearing first. It is then returned back

to the Mezzanine recommender which looks up the IDs of the Events, and retrieves the Event objects from the database.

5.4 Maps using External API

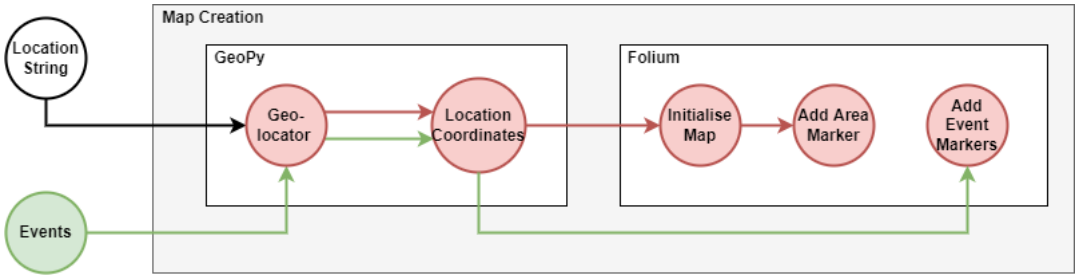


Figure 5.6: *The use of external APIs, GeoPy and Folium, to create the interactive Map used in Mezzanine.*

Location visualisation its a core part of Mezzanine. As a lot of the features are location based - Venue addresses, Talent Location, location search and surrounding are functionality - an interactive map provides the visual detail to partner with address information. Each map is made, and if necessary, populated with markers indicating a Venues location. Mezzanine uses the GeoPy and Folium APIs, discussed in *4.4.3*, for these location services.

When creating a Map to be used in Mezzanine, the requirements of the map must be analyzed. For example, the home page map is a view of the world, users can search for Events and Venues, and maps are a part of Venues profile page. Each of these maps must be created separately based on their needs. Because of this, 5 different types of maps are made:

1. Home Page
2. Location Only with Threshold
3. Search Venues
4. Search for Work (Events)
5. Venue Profile

The process of map creation incorporates 3 main stages. First the requirements of the map must be understood. Then the map location and surrounding area must be identified. Then all Events/Venues in this area must be populated into the map using Markers. Map type 1 is a simple generation of the Folium Map, without any additional add-ons or required parameters. The purpose of this is cosmetic only, however users are still able to zoom, click and rotate the map. Maps 2-4 incorporate user input for the location and surrounding area, and maps 3-5 are populated with markers, however have different needs.

Map 3 focuses on the Venue search and threshold. As seen in *Fig. 5.6*, the process for creating this map involves retrieving the longitude and latitude of a location using the GeoPy geocoder method, and then initialising the Map with this location. The surrounding area is also then received, then any Venues within this area are identified. Visually, the surrounding area is represented using a red circle and each Venue is represented by a marker that contains Venue information when hovered over and clicked. The map, marker, and circle creation are all handled by a method within Mezzanine's map component. The component receives the location, surrounding area threshold and list of Venues, and returns the populated Map.

Map 4 uses the core procedure as above, however the component receives a dictionary of all Events as parameters. The dictionary item is the Event object, each with either 'recommend' or

'standard' as the key. The Events hosting Venue and all its location details can be received due to the FK constraint between the objects. This data is then used in the population of the map. As above, the map produces a red circle showing the surrounding area and each Event is represented as a marker - recommended Events in green, and standard Events in blue.

Examples of Maps 3, 4 and 5 can be found in the front end section, *Fig 5.7*.

5.5 Talent Venue Handshake

The handshake between the Talent and Event was implemented through the use of relationships fields between each model. The Event model contained an additional boolean attribute, is_open, which acted as a switch to indicate when the Event was taking requests and when it was not. Initially, is_open is set to true, and then set to false once a Talent has been confirmed for the Event. This is seen in *Fig. 5.7*. This method of implementation was chosen, as it was the most efficient way to represent the requirement of the sequence diagram shown in *Fig. 4.3*.

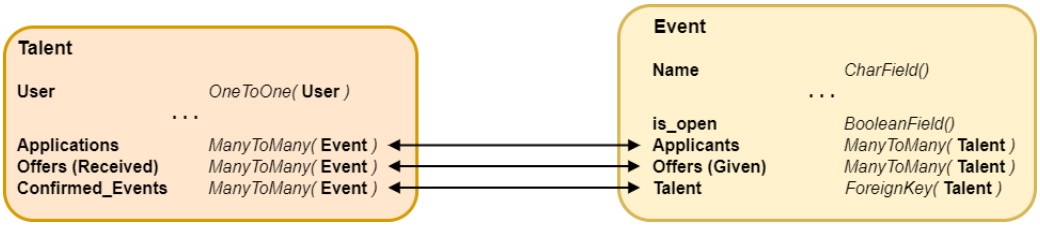


***Figure 5.7**: Model interaction diagram, showing the relationship between each attribute in the models used for the handshake.*

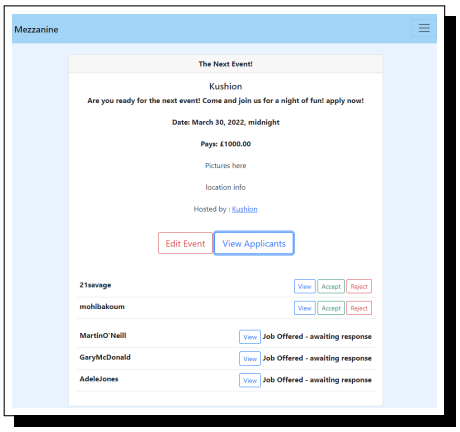
The Talent object contains 3 Many-To-Many fields, as seen in *Fig. 5.7*. These attribute represent the stages of a Talents application process: Applying for Events, Receiving Offers/Being Declined, and being Confirmed onto an Event. The steps for an Events approval process are also stores, and updated dynamically when Talent responds. Mezzanine updates each field with the Event ID through each step.

For example: when Talent has applied for an Event, the Event ID is stored in applications. If an offer is received, the Event ID is removed from applications and added to Offers. If a Talent then confirms the offer, the Event is added to the Confirmed_Events field, and the Event object is updated with Talent as the confirmed act. The confirmed Events are stored to aid in the presentation of the Events on the Talents profile page.

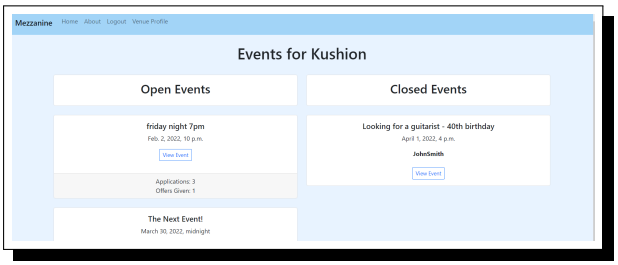
A similar process happens within the Event table, where applicants and given offers are also stored. Once the handshake process is complete, the 'Talent' foriegn key in the Event model is updated with the final confirmed Talent User.

The implementation of the design involved many challenges. It was decided that the optimal approach would be to offer the handshake over as little web pages as possible. This was due to mezzanines commitment to ease of use, and also to reduce the unnecessary transfer of data across web pages.

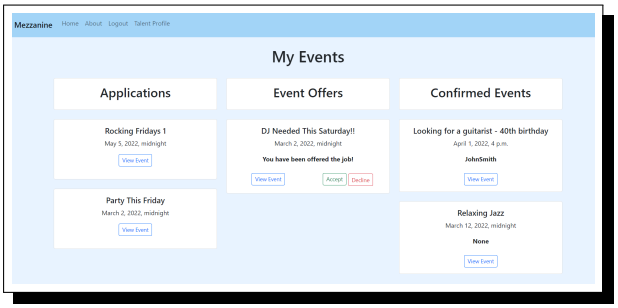
It can be seen in *Fig. 5.8* how the Event applications has been conceptualised. The Venues landing page, *Fig. 5.8 (b)*, shows the Venues current open Events and all closed Events. It was decided that the Event applicants should be shown on the same page as the Event details - as to satisfy requirement *Insert here*: The system should be intuitive and easy to understand. The functionality (accepting, viewing and denying of Talent profiles) is all within one button click, removing all excess 'noise' from the talent-venue handshake procedure.



(a) Venues view of Events applicants, with the ability to view the Talents profile, accept or deny their application



(b) Venues' Event applicants landing Page



(c) Talent's Application landing page

Figure 5.8: *The front-end solution to the Talent-Venue handshake, a landing page for Venues and Talent applications/offers.*

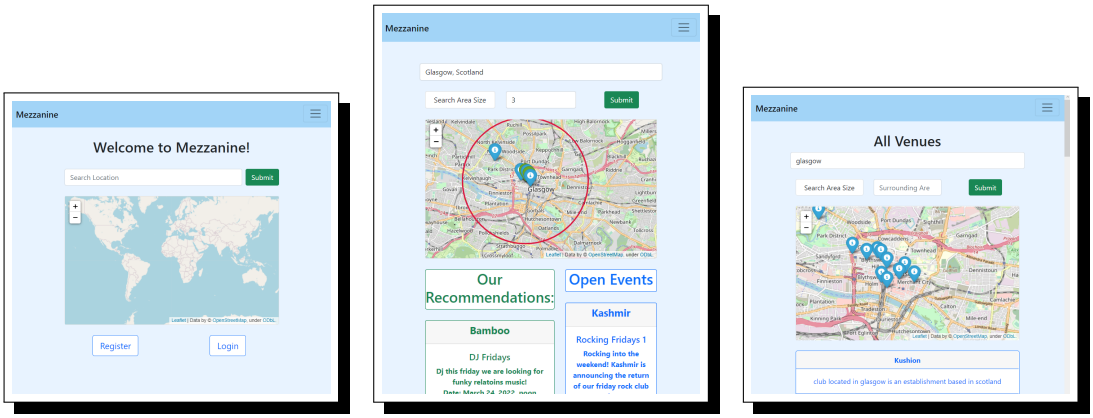
5.6 Front End

Mezzanine was developed with ease of use and efficiency in mind, as per *requirement x*. When implementing the front end, the designs and markups created in the idea generation phase of the process were adhered to as accurately as possible. This created consistency throughout the design and implementation stages which meant no last-minute changes were being made. Multiple changes without correct planning documents could have negatively affected the development. Multiple implementation techniques were deployed for implementing a dynamically created and efficient front end. These techniques will be discussed in this section.

5.6.1 Django - Template Tags

The front end of Mezzanine is generated dynamically by the View within the Django framework. As mentioned in the *Background*, Django operates using a Model-View-Template structure. The model contains the Event, Talent and Venue data, while the Templates are the skeleton HTML of each page. The view retrieves the recommender logic, map-creation functionality, data from the models and combines it all before populating the template and returning it to the user.

This interactivity and population of HTML is done through the use of Djangos 'template-tag' functionality. Where variables can be passed from the view to the template and then populated directly into the HTML. Template tags also include functionality such as loop and conditional flow. *Fig. 5.9 (b)* shows how this is utilised with Event applications. In this case, the view has retrieved all Event's, and passed them to the Template as lists. The template then populates the Event data into the HTML using the bootstrap cards layout.



(a) Mezzanine's Home page, when a user has not signed into a profile.

(b) Find Work page, showing recommended Events in green.

(c) Venues section of Mezzanine, showing all Venues on the map.

Figure 5.9: *Final deliverable of Mezzanine, with examples of the Recommending, Map and search functionality.*

5.6.2 Bootstrap

Bootstrap, as discussed in *Tools and Technologies*, was used to handle the front-end layout and styling. Bootstrap was chosen instead of creating bespoke CSS files due to the styling and resources it offered. The Bootstrap Docs [reference] offered pre-created code snippets. This meant there was a minimum learning curve needed, unlike other methods of styling. Bootstrap features are accessed by connecting each template tag to the resource via a link. This link was added into Mezzanines Base.HTML - a skeleton template which each subsequent template builds on top of. Core front end concepts like the size of the Map, styling of the buttons, Event cards and nav-bar were all supplied by bootstrap.

5.6.3 Interactive Map

As mentioned previously, the Folium import builds the map, populates it and returns it to the view as a HTML component. Once returned, the map is returned to the Template as a Django Template-Tag, allowing it to be populated in the web page. The map was chosen to be a main feature throughout the site to aid in the visibility of locations. The final implementation of the map can be seen in *Fig. 5.9*, where it is used differently in the Home, Find-Work and All-Venues pages of the site.

5.7 Summary

This chapter provided a brief overview of the software engineering practices used in the development and also the back-end and front-end implementation of Mezzanine. The recommender logic and architecture was shown at a high level, then each component of the architecture was broken into its sub-components and discussed, highlighting how Mezzanine makes recommendations. The hand-shake and interactive map implementation was also covered, before finishing the chapter with the technologies used in populating and styling the front-end.

6 Evaluation

In this section, the evaluation techniques used to evaluate Mezzanine and its effectiveness as a system are discussed. -quote about effectiveness. The system contains many features and web-pages that can be evaluated, however, the requirements of Mezzanine can be generalised into two research questions:

1. Does Mezzanine provide an easy to use interface with minimal noise.
2. Does Mezzanine recommend relevant Events based on a Talents profile.

These two research questions incorporate the Human Computer Interaction (HCI) aspect of the product and also the recommender functionality of the product. Together, an overall evaluation of the site can be completed, and a benchmark for future development can be realised. In total, 11 evaluations were completed by participants who has not interacted with Mezzanine previously. The evaluation was completed while adhering to the School of Computing Science checklist found in apendix x.

6.1 Pilot Study

Preceding the full usability and recommender evaluation taking place, a pilot study was conducted that involved just one participant completing the survey and tasks required. The purpose of this study was to ensure the tasks were valid for the other participants, that the system worked as it should, and that there was sufficient time to fix any bugs before the wide-scale testing began. This study received verbal, qualitative data through discussion, and quantitative data through the tasks and survey response. During the study, the participant was allowed to ask questions with regards to the tasks given - this allowed for the wording of questions to be analysed and changed if needed and aspects of the site to be changed. When conducting the study, multiple issues were raised with regards to different parts of the system. These issues were analysed and then relevant changes were made when they could be. Other issues were not relevant to the functional or non-functional requirements and could be disregarded.

6.1.1 Changes Made

The participant highlighted that the changing buttons on the home screen were not entirely visible as they were positioned below the map which covered a large portion of the screen. This problem was mitigated by moving the buttons from below the map, to above the map. No issues were raised in the full evaluation about the visibility of these buttons.

The participant also highlighted that the wording of the survey was unclear. The likert scale was used with the terms 'very good' and 'very bad' as opposite threshold limits. These terms were confusing for the participant for tasks related to the navigation and layout of the site. The participant was also unsure if a score of 3 was good or bad. For the final user evaluation the terms were changed: Tasks relating to the layout of the site were given threshold terms of 'very challenging' to 'very simple' and tasks relating to the layout were given terms 'very uninformative' to 'very informative'. An advisory sentence was also added, informing the participants that a score of 3 is neutral.

The participant could also access editing functionality for other Venues when they navigated to a venues profile page. This was due to a permission conditional flow error and was resolved before the final survey was handed out.

6.1.2 Changes Disregarded

The participant highlighted that the system was slow in between page loads. This is an error that was realised throughout development and solutions were theorized. However, the purpose of the project is to provide a proof-of-concept of the overall system. This means an emphasis on the front-end usability, and the back end recommender(s) were prioritised over a quick response time.

6.2 User Evaluation - Usability

Mezzanine is a system designed to be used by Venue owners and Talent, meaning the front-end of the system should be self describing and easy to operate for all software-fluency levels of user. The usability Evaluation relates to the first research question: Does Mezzanine provide an easy to use interface with minimal noise?. A heuristic evaluation of the site was necessary to gain quantitative and qualitative feedback, to gain an understanding of whether or not research question 1 was satisfied.

6.2.1 Experiment Methodology

The usability evaluation was created to test the user interface of Mezzanine and the level of ease participants felt when carrying out tasks (Alonso-Ríos et al. (2018)). Each task was grouped into three metrics, which aimed to gain feedback on the overall user experience. There was also an extended response section:

1. The User Input and Interaction pages
2. The navigation of the site
3. The layout and styling of the web pages
4. Extended response to gain qualitative feedback

For tasks 1 and 2, the likert scale was used with the range of 1 to 5, however different descriptive words were used. Navigation used the terms 'very challenging' to 'very simple' (Boone and Boone (2012)). Layout questions used the terms 'very uninformative' to 'very informative'. A response of 3 was considered neutral.

For each task, there was also an optional extended response section, where participants could add in additional comments they had about their experience. This section allowed users to explain their score if they so wished.

6.2.2 Results

Results of 4 or 5 in the likert scale were considered positive. With scores of 3 and under being considered negative. Although 3 is a neutral response, a neutral score does not reflect positively on Mezzanines first research question which aims for ease of use.

User Interactions

The user interaction tasks related to the pages where input was required by the user. therefore satisfying *requirementx* and helping answer the research question 1. As mentioned, the likert scale was used to gain feedback on users experiences. The score for each page was found and averaged for each participant. Scores of 4 and 5 represent positive interactions, where scores of 1-3 represent overly challenging or confusing user input pages.

	Web Pages with Interactions							
User	Register	Create Event	Sign In	Search Events	Applicants	Edit Account	Sign Out	Avg Score
A	5	4	5	5	4	5	5	4.71
B	5	4	5	4	5	4	5	4.57
C	5	5	5	5	4	3	5	4.57
D	5	5	5	5	5	5	5	5
E	3	4	4	4	4	4	4	3.86
F	4	4	4	4	4	4	5	4.14
G	4	3	5	5	4	5	5	4.42
H	5	5	5	3	5	5	5	4.71
I	5	4	4	4	4	5	5	4.43
J	5	5	5	5	4	4	5	4.71
K	5	5	4	3	4	4	4	4.14

Table 6.1: *User Input and Interaction with Mezzanine*

The lowest average score received was 3.87. This occurred when participant E gave a score of 3 for the registration page and a score of 4 for each other page. The highest score received for user interaction was by participant's A and J. They both had a split of two '2' scores and five '5' scores. When all average scores were combined and averaged the final average User Interaction Score was 4.49. This score was satisfactory to satisfy the ease of use requirement of Mezzanine. As stated in the introductory paragraph to this section, scores between 4 and 5 were considered acceptable. The average falls within this threshold.

Layout and Styling

	Pages with different Layout and Styling					
User	Register	Create Event	Search Venues	Profile Page	Find-Work	Avg Scores
A	5	5	5	5	5	5
B	4	5	5	4	5	4.66
C	5	5	5	4	5	4.83
D	4	5	5	5	5	4.83
E	4	4	4	4	3	3.87
F	3	3	4	4	5	3.87
G	5	4	5	4	2	4
H	5	5	5	5	5	5
I	5	5	5	5	5	5
J	4	5	5	5	5	4.83
K	5	4	4	4	5	4.41

Table 6.2: *User feedback on the layout and styling of the web pages.*

The Layout and Styling Results relate to the opinions of the participants about the visual aspects of Mezzanine. Scores of 4 and 5 represent positive feedback, where scores of 1-3 represent inconsistencies or unlined styling and must be investigated.

The lowest average score received was 3.87. This score was the average of participants E and F. The highest average score received for Layout and Styling was by participant's A, H and I, who gave each page scores of 5. The lowest single result obtained was by participant G, who gave a score of 2 to the 'find-work' page. When average scores were combined and averaged the final average Layout and Styling Score was 4.54. This score was enough to satisfy the styling and simplistic design aspect of the requirements of Mezzanine. As stated in the introductory paragraph to this section, scores between 4 and 5 were considered acceptable. The average falls

within this threshold.

Navigation

	Navigation to Different Pages						
User	Registration	Create Event	Search Venue	Profile Page	Events	Find Work	Avg Score
A	5	5	5	5	5	4	4.83
B	5	5	3	4	5	4	4.33
C	5	5	5	3	5	5	4.66
D	5	4	5	5	5	4	4.66
E	4	4	4	4	4	4	4
F	4	4	4	4	4	4	4
G	5	3	5	5	5	4	4.5
H	5	5	3	5	5	4	4.5
I	5	4	5	4	4	5	4.5
J	4	5	5	5	4	5	4.66
K	5	5	4	4	4	5	4.5

Table 6.3: *User response for the navigation of the system*

The Navigation Results indicate the experience of participants on moving around the system from one web page to another. Scores of 4 and 5 represent positive feedback, where scores of 1-3 represent negative feedback and indicated the site it not self describing.

The lowest average score received was 4.33. This score was received by participant B, who scored 3 for the 'search-venue' page, bringing the average down. The highest average score received for Navigation was from participant A, who gave an average score of 4.83. The lowest single result obtained was by participant's E and F, with average scores of 4. When all average scores were combined and averaged the final average Navigation Score was 4.47. This score was enough to satisfy the ease of use aspect of the requirements of Mezzanine. The average score falls within the 4-5 threshold and can be considered a positive result.

Extended Responses

Extended response questions for Usability evaluation varied. Participant B suggested the find-work hyperlink should also be present in the Nav-Bar: this would aid navigation into the find-work section from any page in the site as currently the only way to get into the section is through the home-page. Participant C suggested that the edit-account functionality did not navigate to a new web page, but instead allowed immediate editing in the same card. Participants C and K asked for a 'scrolling' feature to be added and that the registration page could be nicer. These improvements were noted.

6.3 User Evaluation - Recommender

Mezzanine's core functionality is to recommend Events to Talent based on the information retrieved from the Talents user profile. The recommender evaluation relates to the second research question: Does Mezzanine recommend relevant Events based on a Talents profile? An evaluation of the core functionality of the site was necessary to gain quantitative and qualitative feedback and to gain an understanding of whether or not research question 2 was satisfied.

6.3.1 Experiment Methodology

The recommender evaluation survey and tasks were created to test the effectiveness of Mezzanine as a recommender system. The survey was created to test the two types of recommending algorithms used in Mezzanine: cosine-similarity based and cluster analysis based. The calculations for precision, recall and the f1 score within Dalianis (2018) were adhered to.

A set of tasks were created which involved navigation, applying for events, and giving an opinion on the recommended events. This set of tasks was identical for both experiments, in order to keep the results consistent. Consistent questions meant both recommenders could be compared and a final decision can be made about effectiveness. The tasks and responses allowed for the precision, recall and F1 score of the recommender to be calculated, as mentioned in *Background 1.2*.

Each user has the ability to input a description, allowing free expression of their own personality. Mezzanine is built to cater for all types of Venue and Talent - meaning any performer from any genre of music is encouraged to use the site. These reasons pose challenges when creating a suitable evaluation metric for the recommender as participants may test the recommender as a DJ, classical singer, rock-band drummer, etc. Venues and Events must also be created that cater to these needs - creating a suitable number of Events and Venues to cover every genre and sub-genre of the music sector was unfeasible.

The solution was to require the participant to sign up as a Talent user, while adopting a *Persona*. The persona featured information on the background, interests and characteristics of a guitarists, as well as motivations for using Mezzanine. They were asked to register for an account with this information, and were free to select any tags, and create a description they deemed suitable. This allowed for a realistic approach to the registration process, where the likely-hood of two Talent accounts having the same user profile description and tags is low.

A test environment for Mezzanine was deployed. Multiple Venues and Events objects were created, catering to different genres of music performer. Hotels, Nightclubs, Pubs, Bars were created and Events made for each including karaoke, Venues asking for DJs, rock-nights and hip-hop nights.

Once an adequate break was taken after the completion of section 1, and the new persona account was registered, participants were moved onto the tasks in section 2.

6.3.2 Experiment 1 - K-Means Clustering

The first experiment involved using the K-Means clustering based recommender. Participants were asked to Navigate to the find-work page in Mezzanine. From there, they were asked to search for work in *Glasgow*. Glasgow was populated with more than 20 Events, therefore utilising the K-Means distance algorithm.

6.3.3 Experiment 2 - Cosine Similarity

The second experiment involved using the cosine-similarity based recommender. Participants were asked to navigate to the find-work page in Mezzanine. From there, they were asked to search for work in *Edinburgh*. Edinburgh was populated with less than 20 Events, therefore utilising the Cosine-Similarity algorithm.

6.3.4 Results

The results obtained from the survey helped to identify the effectiveness of the recommender systems within Mezzanine. The user was asked to search for work within a location and evaluate the recommended Events, highlighting if they found them to be relevant or not. This data was

then used to calculate the precision, recall and F1 Score of each recommender. The results can be disputed and will be discussed in the *limitations* section.

K-Means Clustering Algorithm

User	Missed Events	Total Rec'd	Applied from Rec'd	Relevant from Rec'd	Precision	Recall	F1 Score
A	1	2	2	2	1	0.67	0.8
B	1	3	2	3	1	0.75	0.86
C	2	3	3	3	1	0.6	0.75
D	1	2	2	2	1	0.67	0.80
E	0	3	3	3	1	1	1
F	0	3	3	3	1	1	1.00
G	2	3	3	3	1	0.6	0.75
H	2	3	1	1	0.33	0.33	0.33
I	1	2	2	2	1	0.67	1
J	1	3	3	2	0.67	0.75	1.00
K	1	3	3	3	1	0.75	1

Table 6.4: *K-Mean Clustering Algorithm Outcome*

The K-Means results obtained from the surveys showed an overall positive review of the recommender due to high scores for Precision, Recall and F1. One of the tasks involved asking the participant to apply for 3 Events, then inputting how many of these Events were recommended. The results were analysed - with 7 participants applying directly from the recommendations, 3 participants applying for 2 recommendations and 1 participant applying only for 1 recommended Event. This means that the recommender returned at least one relevant Event on each occasion. In total, 81.8% of all Event applications were for Events recommended by the K-Means clustering algorithm.

Precision, Recall were calculated for the algorithm. The lowest instance of precision and recall was in participant H, with scores of 0.33 for both, which resulted in an F1 score also of 0.33.

The highest score for precision and recall was for participant A, where scores of 1 and 0.75 were found - this resulted in an F1 score of 0.86.

The average precision and recall for the clustering recommender was 0.90 and 0.70 respectively. This resulted in the k-means recommender having a final F1 score of 0.78. This score is suitable as it indicated a balance between precision and recall. As each participant applied to at least one recommended Event, it indicates that the algorithm has a 100% success rate at showing at least one item in which the user believes to be relevant.

Cosine-Similarity Algorithm

The Cosine-Recommender results obtained from the surveys showed a positive response of the recommender due to high scores for Precision, Recall and F1. The same tasks required for the K-Means recommender were used for the Cosine-Similarity recommender.

The 3-Event-Applications task was analyzed - with 7 participants applying to all 3 directly from the recommendations, 2 participants applying for 2 recommendations and 2 participant applying only for 1 Event from the recommended subset. This means that the recommender returned at least one relevant Event on each occasion. The application outcome for the Cosine-Similarity recommender was the same as the K-Means method with 81.8% of all Event applications being recommendations.

User	Missed Events	Total Rec'd	Applied from Rec'd	Relevant from Rec'd	Precision	Recall	F1 Score
A	1	4	2	3	0.75	0.75	0.75
B	2	4	3	3	0.75	0.6	0.67
C	1	3	3	3	1	0.75	0.86
D	1	5	3	3	0.6	0.75	0.67
E	0	4	1	4	1	1	1
F	1	5	3	4	0.8	0.8	0.80
G	1	4	2	3	0.75	0.75	0.75
H	1	4	3	4	1	0.8	0.89
I	0	4	3	4	1	1	1
J	1	5	3	3	0.6	0.75	0.67
K	0	4	1	4	1	1	1

Table 6.5: *User response for the navigation of the system*

Precision, Recall were calculated for the algorithm. The lowest instances of precision was in participants J and D, with scores of 0.6. The lowest instance of Recall was in participant B, with a score of 0.6. The lowest F1 score instance was 0.67 and recorded three times in participants B, D and J.

The highest score for precision and recall was for participants E, K and I, where scores of 1 and 1 were equated - this also resulted in an F1 scores of 1.

The average precision and recall was 0.84 and 0.81 respectively. This resulted in an average F1 score of 0.82. This score is suitable as it indicated a balance between precision and recall. Similarly to clustering, each participant applied to at least one recommended Event, meaning that the algorithm has a 100% success rate at showing at least one item in which the user believes to be relevant.

6.3.5 Recommender Comparison

Both recommenders produced satisfactory results, with F1 Scores approaching 1. Clustering produced an average Precision and Recall of 0.9 and 0.7 respectively, with cosine producing scores of 0.84 and 0.82. Analysis of this data shows clustering with the larger recall, but less precision when recommending Events. The final F1 scores were: 0.86 for the K-Means Clustering distance based algorithm and 0.82 for the cosine-similarity based recommender. Both recommenders show promise, with cosine being more balanced across precision and recall.

6.3.6 Limitations

The precision and recall for the evaluation would be subject to change with larger datasets. Participants may also have different perspectives on what is a suitable recommendation - meaning the opinion of each participant was a direct factor in calculating the effectiveness of Mezzanine. The evaluation should be re-completed on a larger scale, across many subsets in order to get a true measure on the effectiveness of Mezzanine.

6.4 Summary

This section discussed the Evaluation techniques used in Mezzanine. The chapter began with the acceptance testing / alpha testing stage in the form of a pilot study before analysing the front end and back end evaluation techniques and results. The results are then compared and limitations are discussed.

7 Conclusion

This paper began with a discussion on the aims and motivations behind Mezzanine, before analysing the background research that was undertaken. Within the background section all aspects were analysed; from market research and business logic to the information retrieval and similarity techniques used to make recommendations. The subsequent chapter derived the requirements of the system, highlighting the elicitation process, then showcased the features derived from this process and their priority levels. This list of requirements acted as a benchmark for the design process, which involved innovating and creating solutions to satisfy the requirements of the system. The successful ideas moved through the process of theorising, sketching, annotating and then recreating the sketches - before using UML to make the technical design documents that were used to aid in the implementation. The implementation section discussed how each of these ideas were developed in a real world software engineering environment. Software practices are discussed before elaborating on the Recommender System Architecture behind Mezzanine and all internal components. The front end implementation process was also analysed - discussing how the front end functionality was created, as well as the styling used. Finally, the site was delivered to a group of individuals for an evaluation. The evaluation focused on the usability of the system and the effectiveness of both recommending systems used in Mezzanine. Questions were asked that were designed to collect the relevant data from the users, and to answer the main question posed in the introduction: Is Mezzanine "an easy-to-use interactive web-app" and does it "utilise information retrieval and Machine Learning practices to streamline the job-hunting process faced by musicians, and the recruiting process faced by venues.".

7.1 Summary

In summary, Mezzanine is a recommender system purposely designed for the hospitality and events sector. It aims to connect Talent to Events hosted by Venues by using various aspects of computer science to make recommendations. As well as the functional back end, it should provide a simplistic and informative user interface, allowing users with all experience levels to access its features. Mezzanine offers recommendations of Events for Talent objects, the booking functionality involving Talent, Events and Venues and delivers on its promise for an intuitive front end with minimal 'noise'. These core aspects of the system were tested through the use of User Evaluation testing which covered the Navigation, User Interaction, and the Layout and styling. Mezzanine uses two recommender systems within the back end, each of these were tested and compared. It was found that Mezzanines front-end was above satisfactory for all users of the tests, while both recommenders managed to deliver at least one useful recommendation on every occasion. therefore, Mezzanine as a system completed the tasks which it set out to accomplish, and with some further work, it could be deployed and utilised by the Food and Beverage sector to aid in post-pandemic recovery.

7.2 Limitations

There are a number of limitations with Mezzanine. It is slow when interacting with the back end database services. This is due to the size and scale of Mezzanine and the light-weight inbuilt

database offered by Django. This was understood throughout the development and it was decided that a functioning, but slow, version of Mezzanine should be prioritised in order to show the proof-of-concept. The user-evaluations also contain limitations. The evaluation asks participants to select 'relevant' Events in order to perform the calculations used for evaluating effectiveness. The concept of relevance is a thought that is unique to each person, meaning different users may interpret the same dataset differently. To mitigate this, more testing should be completed on a wider scale. In this development, the time and resources were not available to complete this. Within the threshold limit for recommending based on the cosine similarity, a value of 0.3 is manually inputted. This hard coded value was the most efficient for the test environment - but poses limitations as the dataset increases. Clustering is also used when the size of Events is greater than 20. This was used to test the k-means functionality, but should be tested further on larger datasets to find an optimal threshold limit.

7.3 Future Work

As it stands at the moment, Mezzanine has been developed as booking system for Venue and Talent, while also providing recommending functionality for Talent and Events. This achieves the minimal viable product and creates a strong infrastructure for future development. As discussed in *Section 4.1*, the MVP was realised, and then further iterations were conceptualised. A third user-type, a standard user, was also discussed. This was where a user could join the site for the sole purpose of interacting with Venues and Talent. This opened up the possibilities for Mezzanine to act as a booking service between standard-users and Events, where users could buy tickets to upcoming Events. The option for Talent and Venues to make posts, comment and rate Venues was also planned for development, but due to set-backs and time constraints, these features were disregarded. Following, followers and news feed features would make a suitable addition to Mezzanine, transitioning the site from only being a recommender, to a social networking site for the Food and Beverage sector.

Including additional user-interaction related features, the system would benefit greatly from back-end work. A correctly implemented database management system would allow for faster retrieval of items from the database, faster recommendations and faster map population. Although the usability evaluations showed the overall approval of the site, a recurring complaint was related to the speed of the page loads. Creating a bespoke back-end would mitigate these problems and allow Mezzanine to be deployed for full use.

7.4 Final Reflection

I am extremely satisfied with both of the recommender systems created and believe developing them improved my overall capabilities as a software engineer. I had to research, develop and discuss each recommender without having any experience of the technologies used, and found the research and discovery of these technologies exciting. I believe that the Talent-Venue Handshake was the main challenge that was overcome. The business logic behind applying for an Event is not the same as other booking systems, such as applying for a job or booking a seat in a cinema. It involves multiple steps and stages of acceptance. I believe Mezzanine creates solution to this problem in a clean and efficient way and was a large task to overcome.

I pursued the idea of Mezzanine as a self-defined project as I truly believe there is a gap in the market and a need for this system. It has not only been my 4th year dissertation project, but also a personal passion project. Working on Mezzanine from the ground up, from sketches to evaluation, has been an extremely rewarding experience for me - and a project that I am proud to put my name beside.

A Database Schema

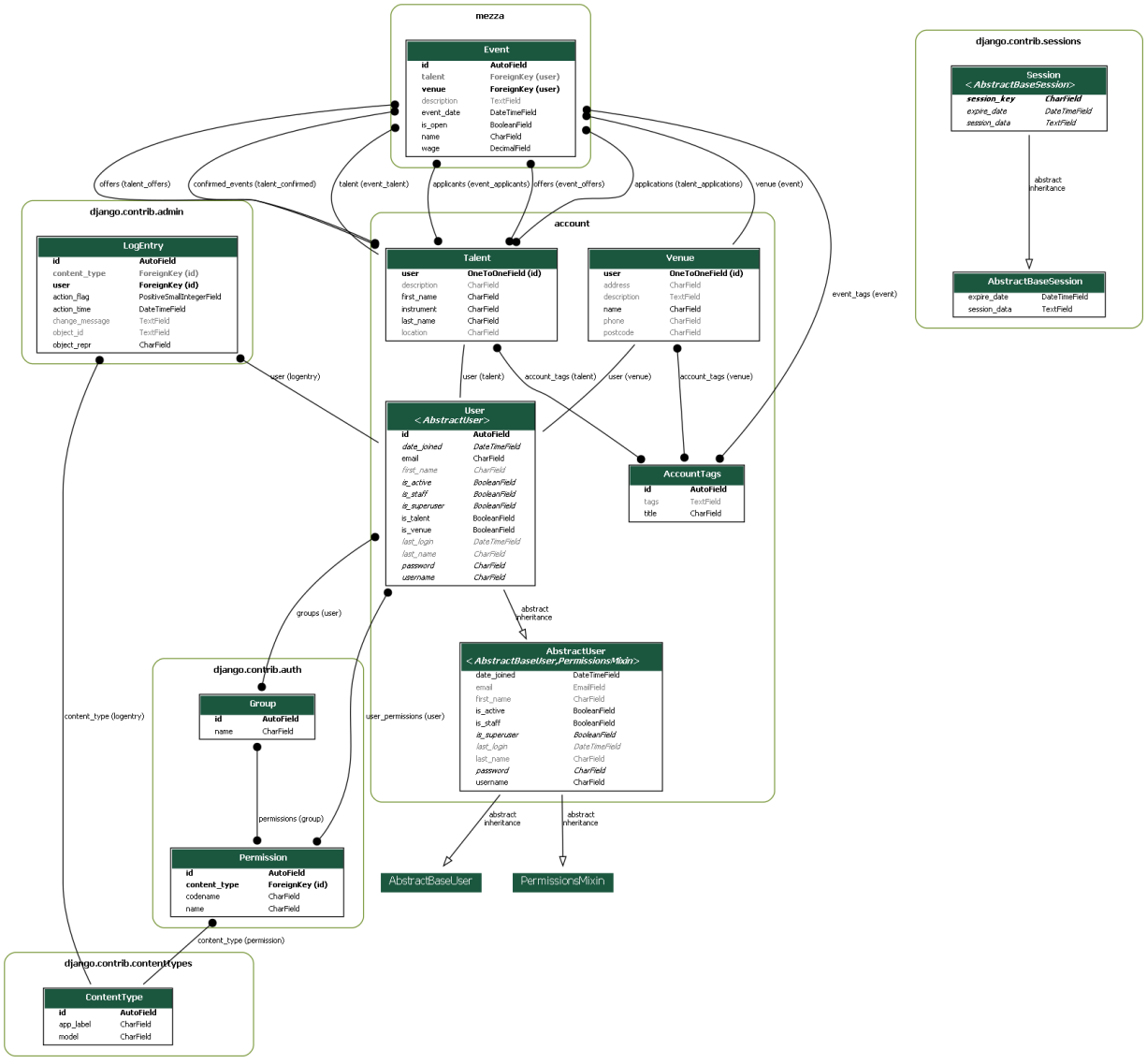


Figure A.1: *Automatically Generated UML DB Schema of Mezzanine*

B Conceptual Designs

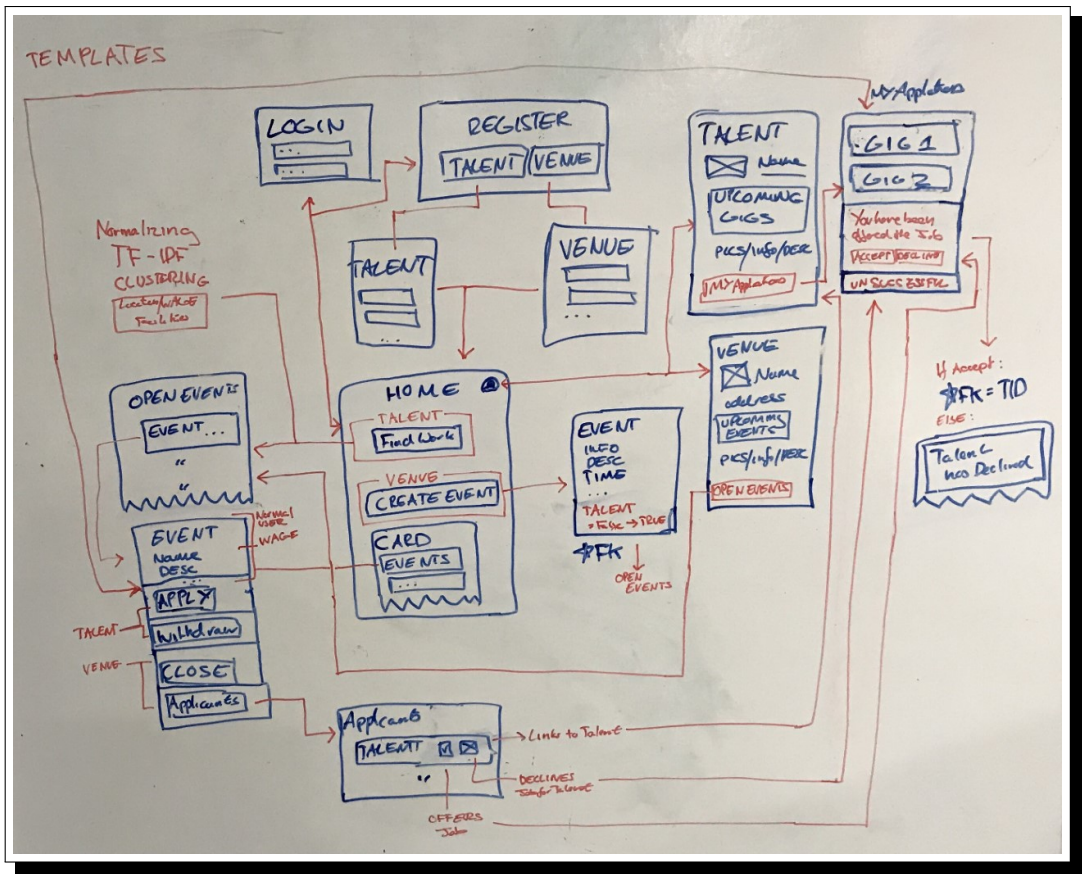


Figure B.1: *Conceptualised Wire frame for Mezzanine*

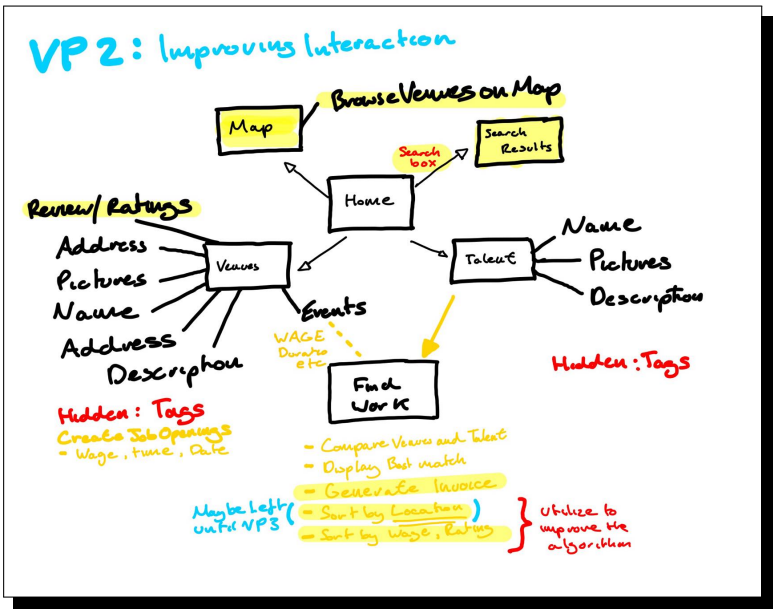


Figure B.2: Second iteration of the product, building on top of the MVP.

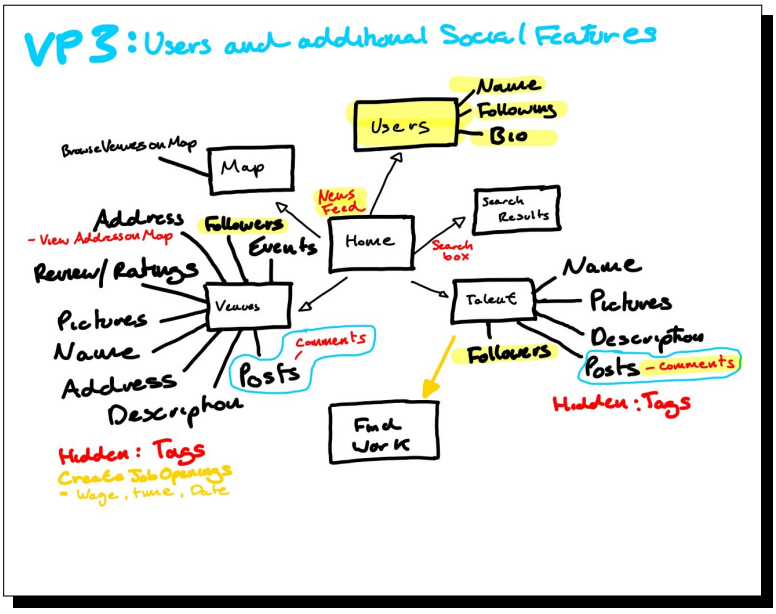


Figure B.3: Third iteration of the product, building on top of the VP2.

C Design Documents

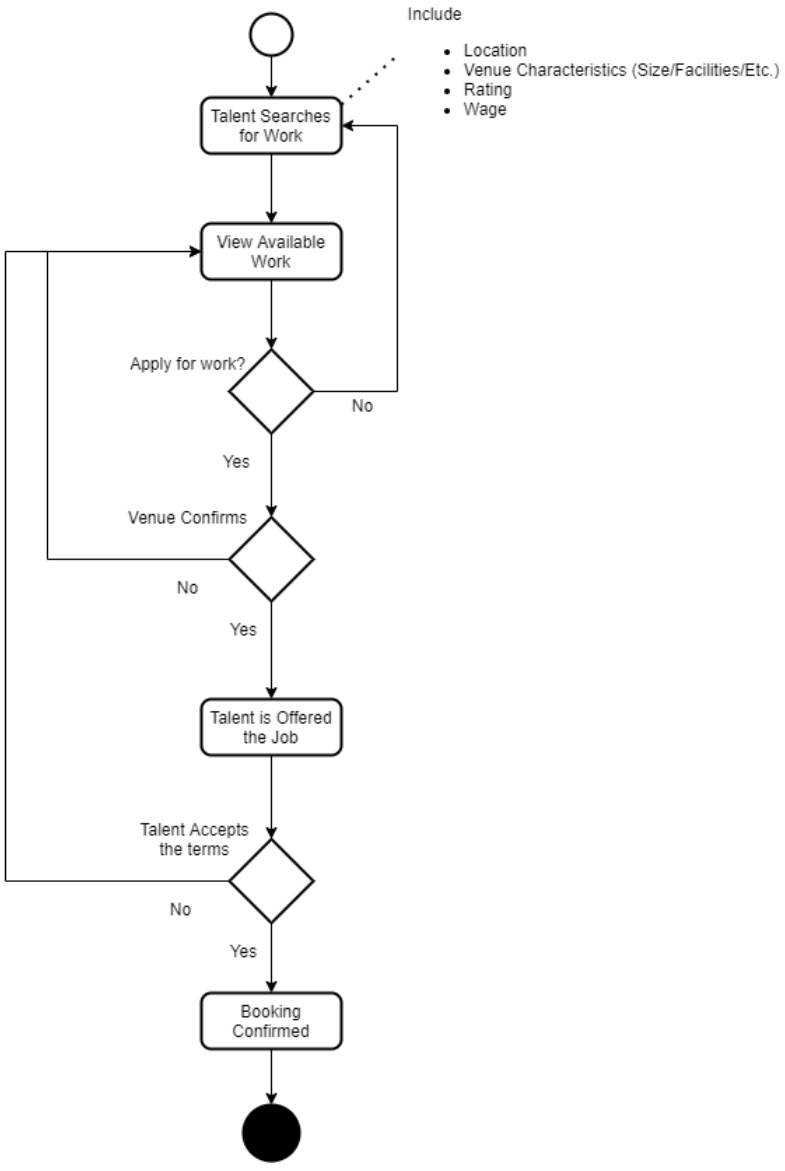


Figure C.1: *Talent Applying For Work Activity Diagram*

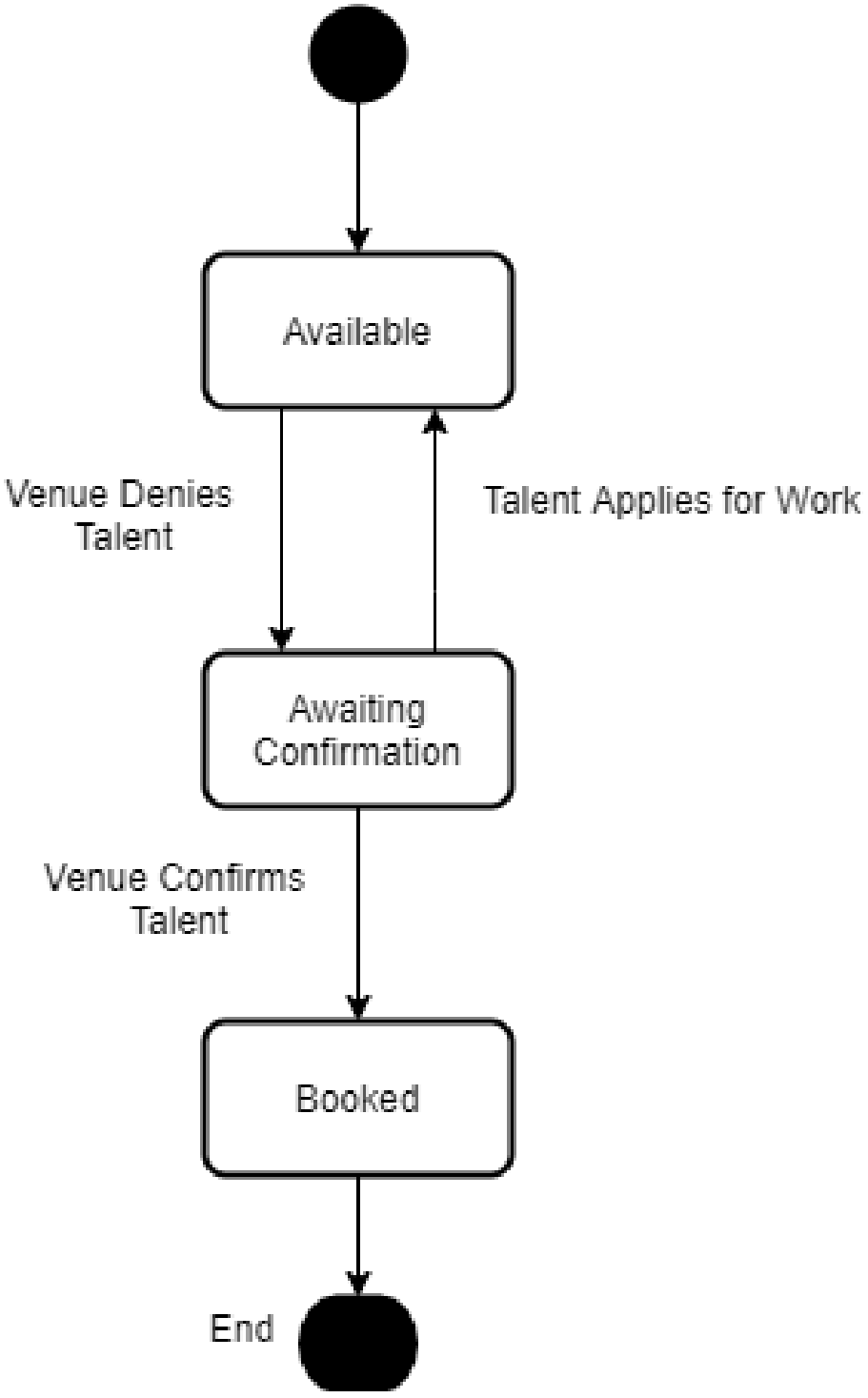


Figure C.2: *Talent Applying For Work State Diagram*

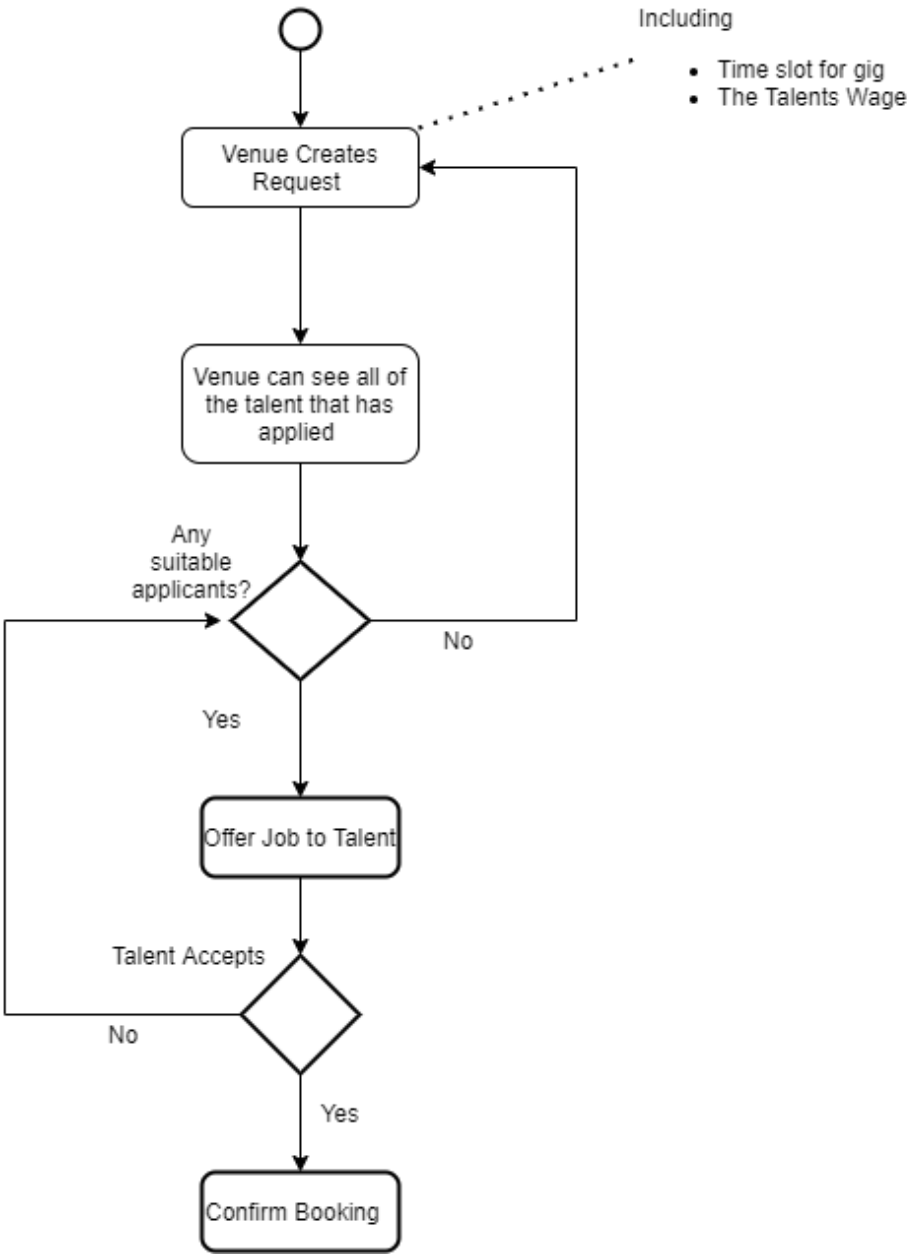


Figure C.3: *Venue Booking Talent Activity Diagram*

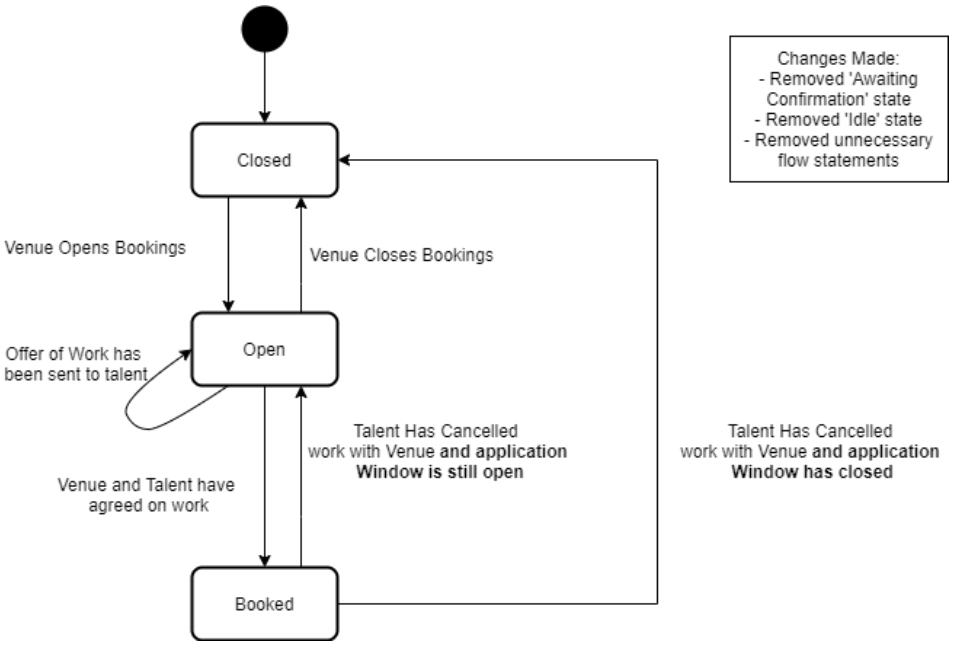


Figure C.4: *Venue Booking Talent State Diagram*

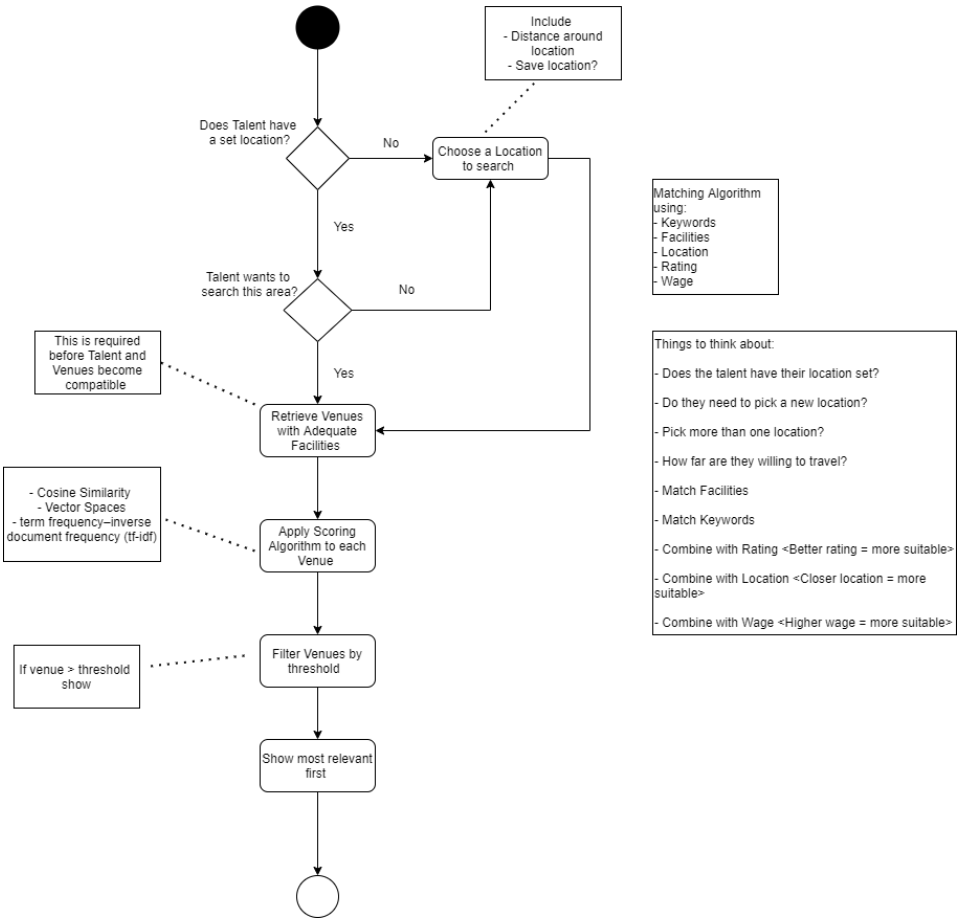


Figure C.5: *Initial design flow for the Recommender Algorithm, Activity Diagram*

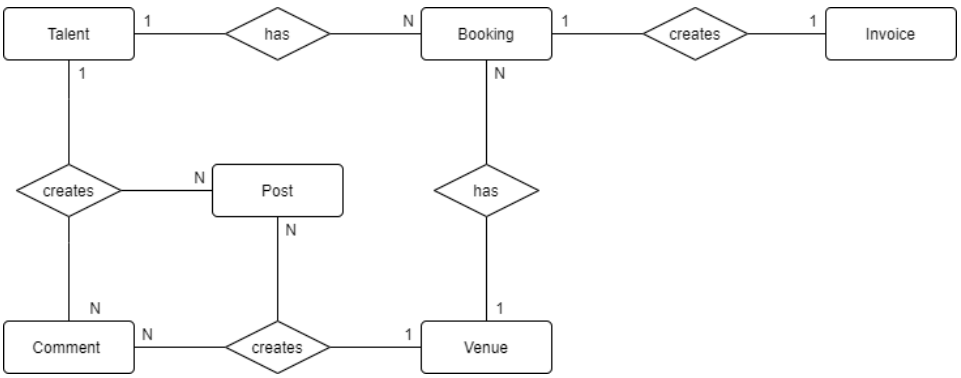


Figure C.6: *Initial ERD Diagram*

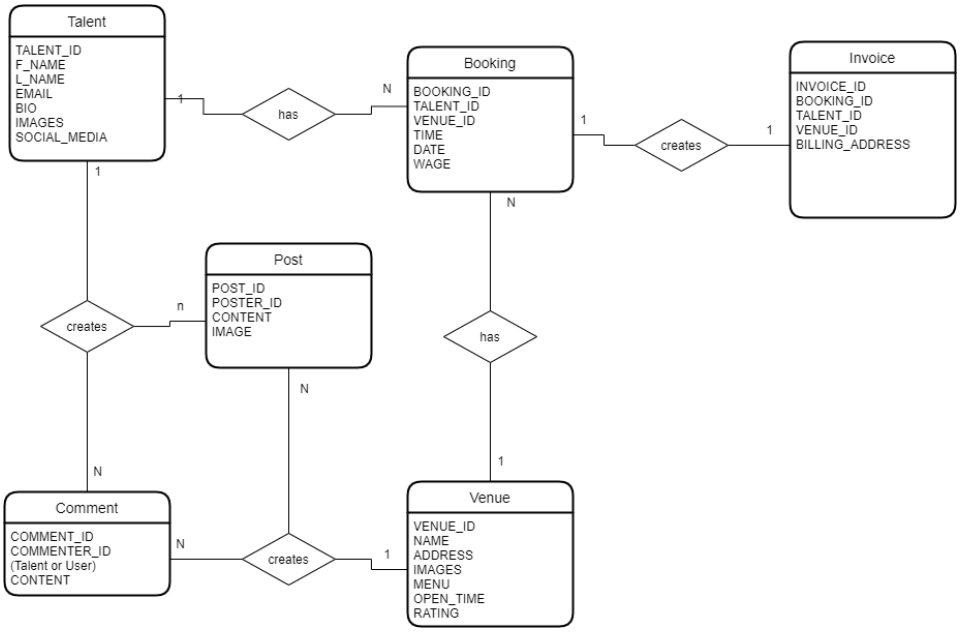


Figure C.7: *Initial Class Diagram*

D Implementation

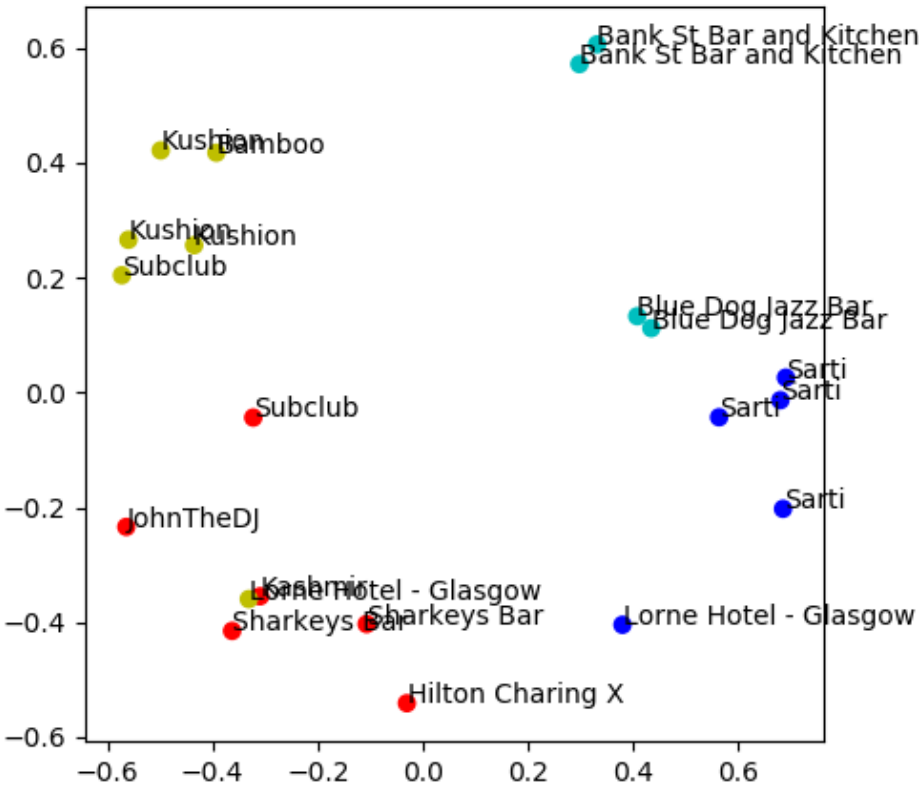


Figure D.1: *Outcome of clustering a small set of Events. Where each label is the Venue name, and each colour is a different cluster.*

E | Final Deliverable

Figure E.1: *Home Page - Talent Signed In*

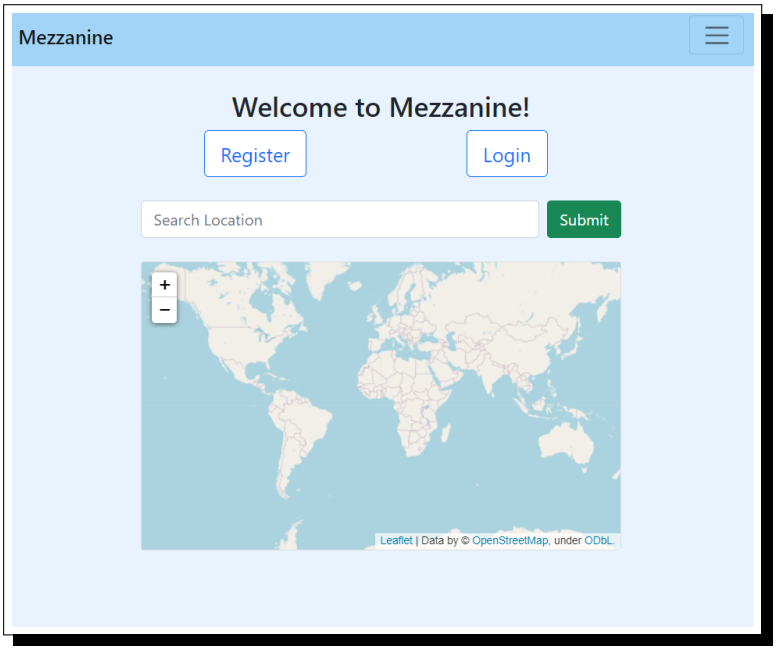


Figure E.2: *Home Page - User signed out*

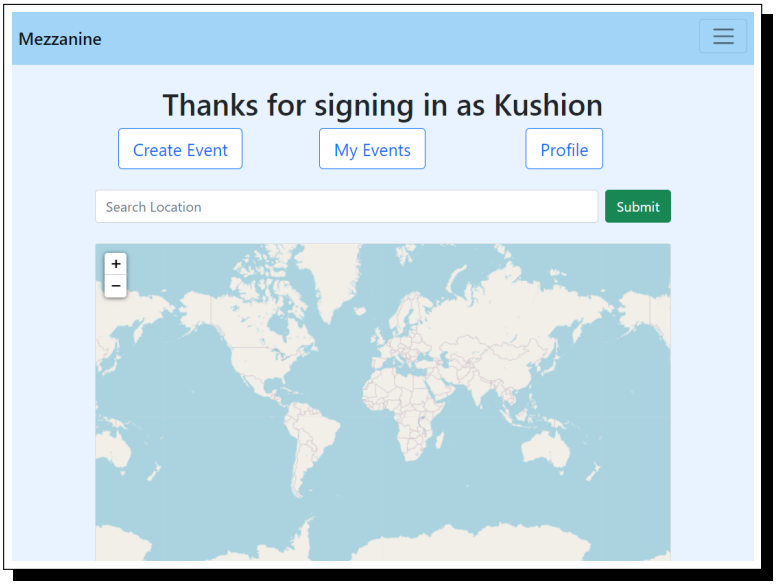


Figure E.3: *Home Page - Signed in as Venue*

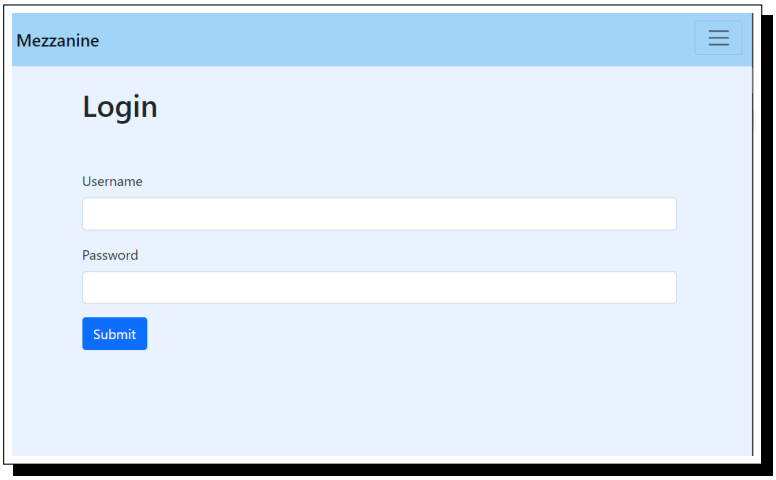


Figure E.4: *Login Page*

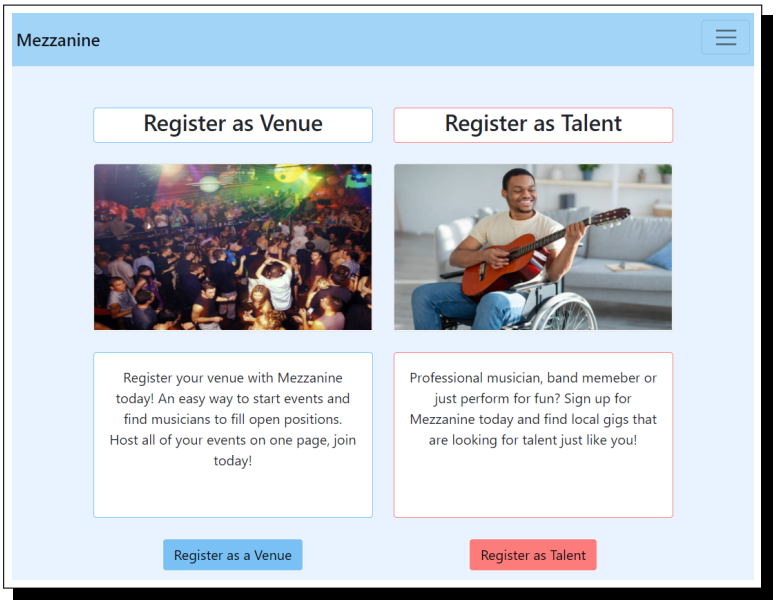


Figure E.5: *Pre-Register Page*

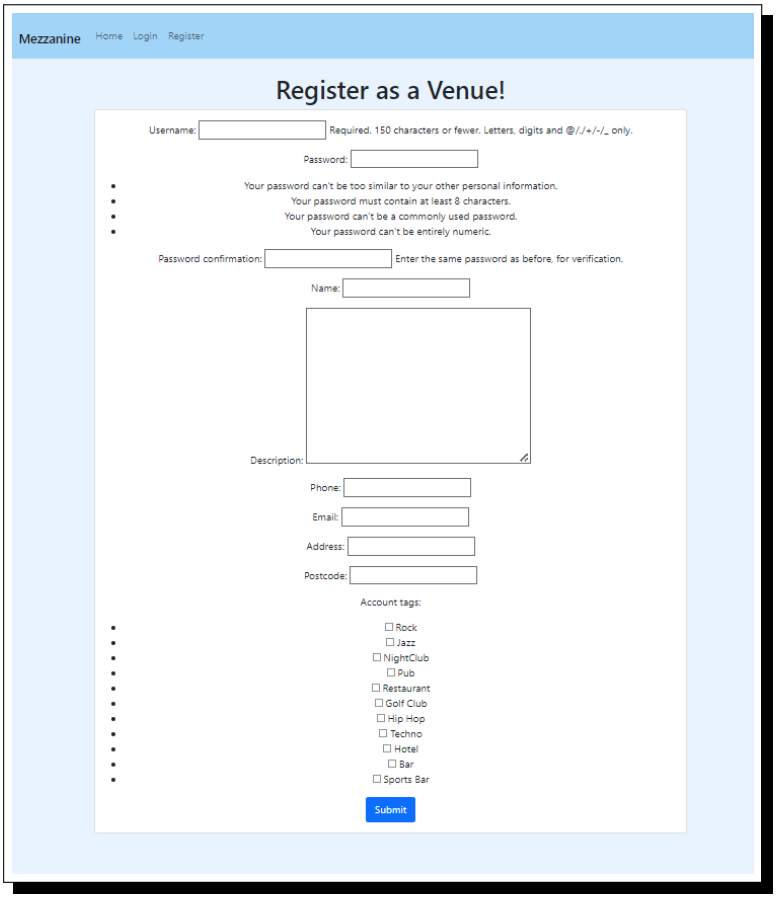


Figure E.6: *Register as a Venue. The Talent register is similar, with different text-box values*

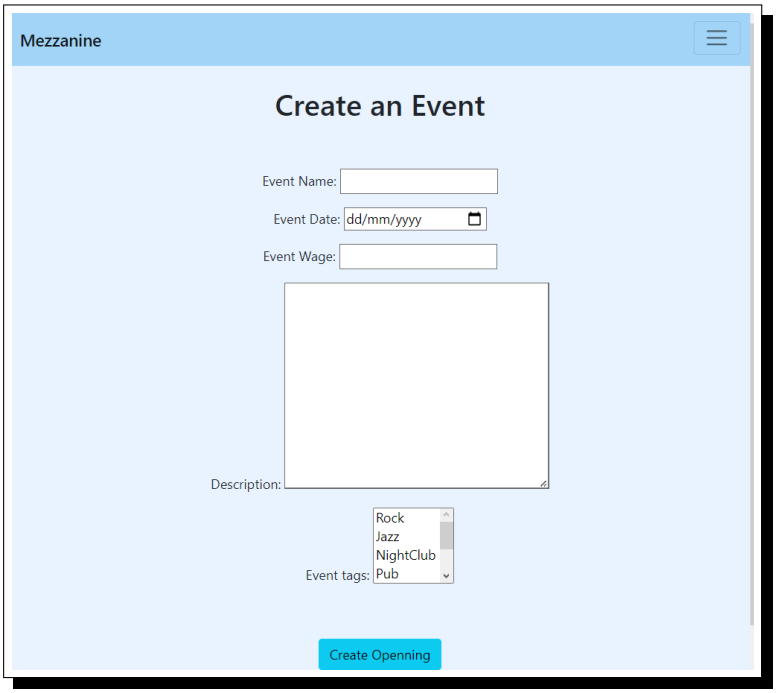


Figure E.7: Event Creation

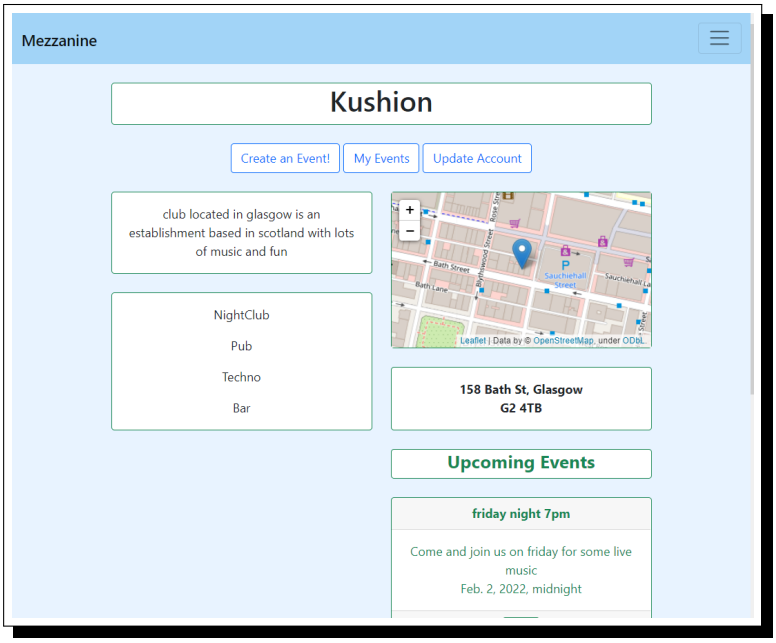


Figure E.8: Venue Profile Page

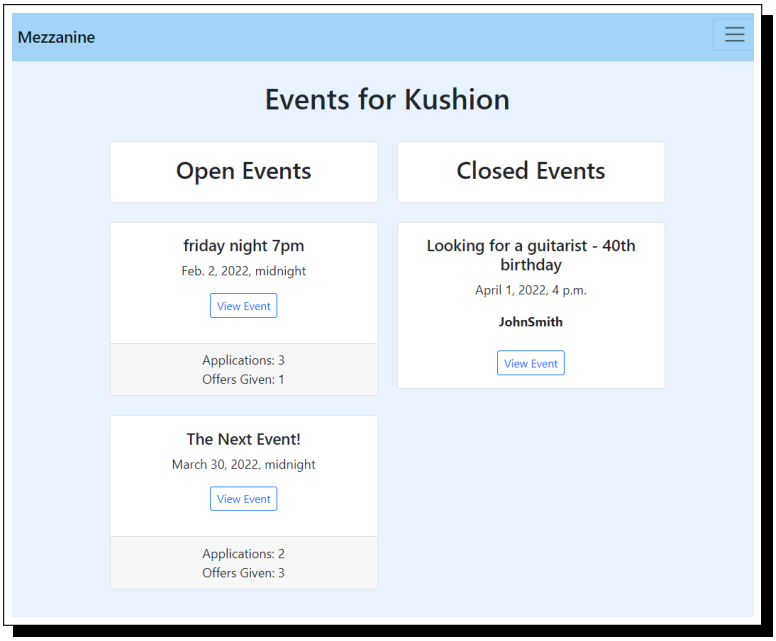


Figure E.9: *Venue Events Page*

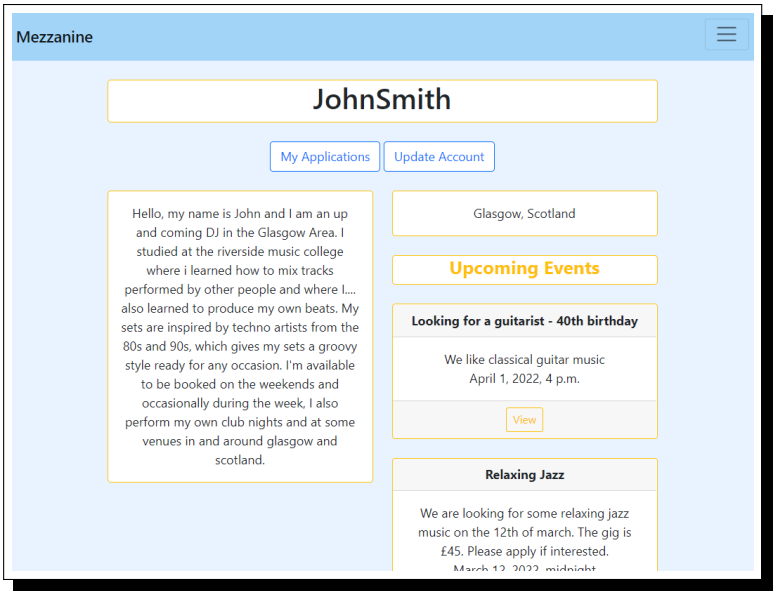


Figure E.10: *Talent Profile Page*

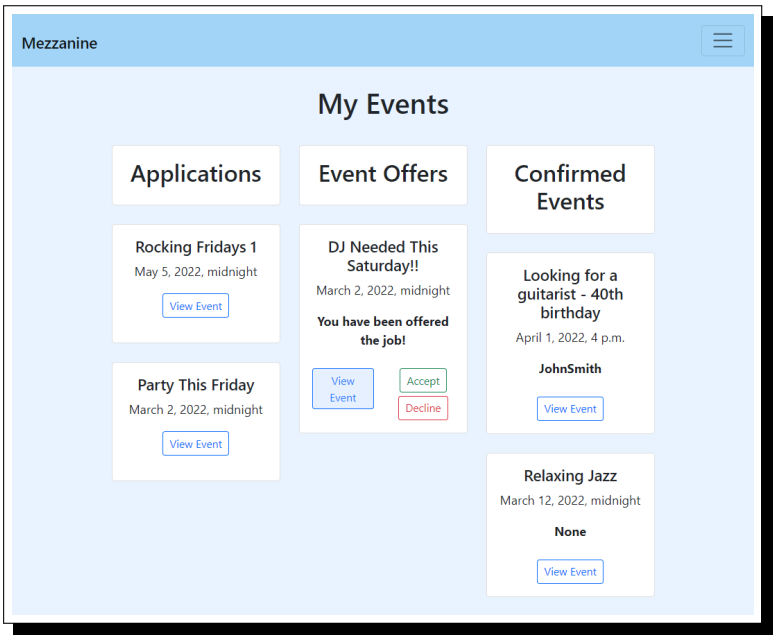


Figure E.11: *Talent Events Page*

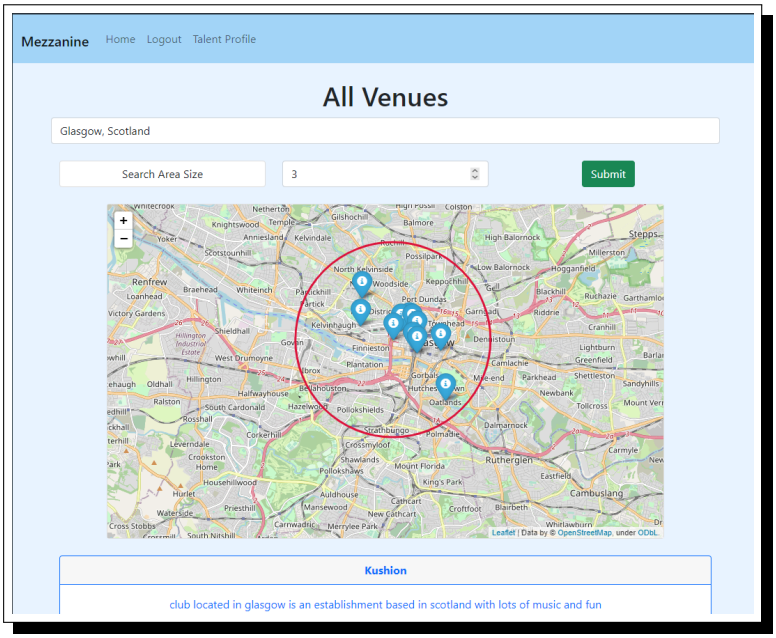


Figure E.12: *All Venues Page*

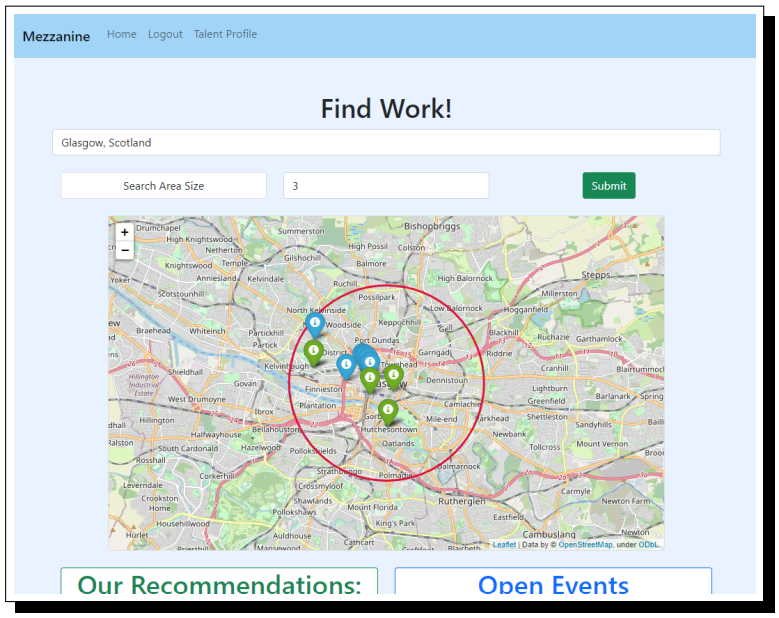


Figure E.13: *Find Work Page (a)*

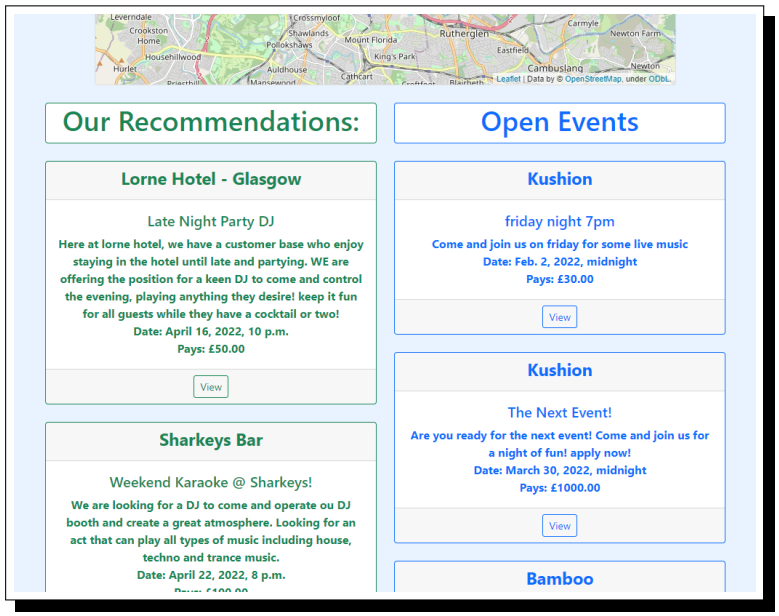


Figure E.14: *Find Work page (b)*

F Ethics Approval

School of Computing Science
University of Glasgow

Ethics checklist form for 3rd/4th/5th year, and taught MSc projects

This form is only applicable for projects that use other people ('participants') for the collection of information, typically in getting comments about a system or a system design, getting information about how a system could be used, or evaluating a working system.

If no other people have been involved in the collection of information, then you do not need to complete this form.

If your evaluation does not comply with any one or more of the points below, please contact the Chair of the School of Computing Science Ethics Committee (matthew.chalmers@glasgow.ac.uk) for advice.

If your evaluation does comply with all the points below, please sign this form and submit it with your project.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life.
   Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that occur outside usual laboratory areas, or that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, that use sensory deprivation (e.g. ear plugs or blindfolds), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback

2. The experimental materials were paper-based, or comprised software running on standard hardware.
   Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, laptops, iPads, mobile phones and common hand-held devices is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.
   If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant.

   Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script.

4. No incentives were offered to the participants.
   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.

Figure F.1: *Ethics check sheet - 1 of 2.*

5. No information about the evaluation or materials was intentionally withheld from the participants.
   Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

6. No participant was under the age of 16.
   Parental consent is required for participants under the age of 16.

7. No participant has an impairment that may limit their understanding or communication.
   Additional consent is required for participants with impairments.

8. Neither I nor my supervisor is in a position of authority or influence over any of the participants.
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.

9. All participants were informed that they could withdraw at any time.
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.

10. All participants have been informed of my contact details.
    All participants must be able to contact the investigator after the investigation. They should be given the details of both student and module co-ordinator or supervisor as part of the debriefing.

11. The evaluation was discussed with all the participants at the end of the session, and all participants had the opportunity to ask questions.
    The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation. In cases where remote participants may withdraw from the experiment early and it is not possible to debrief them, the fact that doing so will result in their not being debriefed should be mentioned in the introductory text.

12. All the data collected from the participants is stored in an anonymous form.
    All participant data (hard-copy and soft-copy) should be stored securely, and in anonymous form.

Project title: __Mezzanine____

Student's Name: __Marty O'Neill_____

Student Number: _2335418o_____

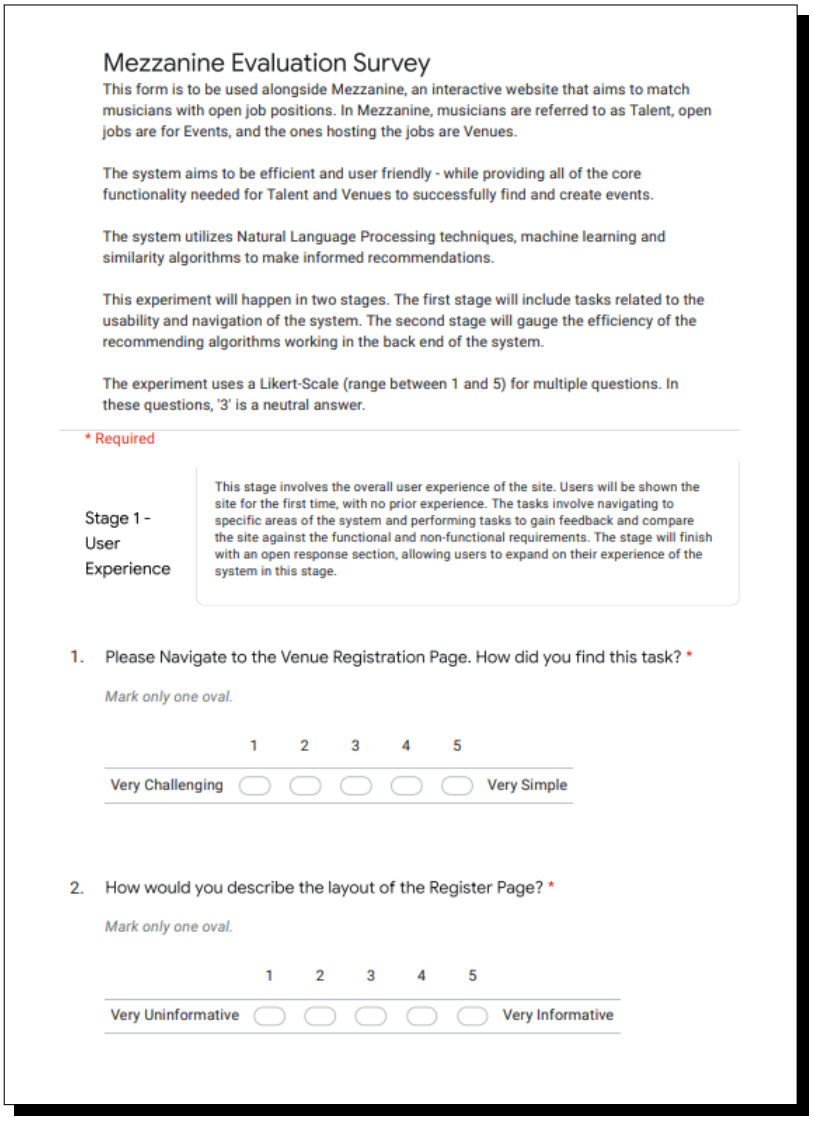
Student's Signature: ___MartyONeill___

Supervisor's Signature: __Christos Anagnostopoulos___

Date: __31/04/2022__

Figure F.2: *Ethics check sheet - 2 of 2. Electronically signed by MartyONeill and Christos Anagnostopulos*

G Evaluation Survey - Usability and Recommender



Mezzanine Evaluation Survey

This form is to be used alongside Mezzanine, an interactive website that aims to match musicians with open job positions. In Mezzanine, musicians are referred to as Talent, open jobs are for Events, and the ones hosting the jobs are Venues.

The system aims to be efficient and user friendly - while providing all of the core functionality needed for Talent and Venues to successfully find and create events.

The system utilizes Natural Language Processing techniques, machine learning and similarity algorithms to make informed recommendations.

This experiment will happen in two stages. The first stage will include tasks related to the usability and navigation of the system. The second stage will gauge the efficiency of the recommending algorithms working in the back end of the system.

The experiment uses a Likert-Scale (range between 1 and 5) for multiple questions. In these questions, '3' is a neutral answer.

* Required

Stage 1 - User Experience

This stage involves the overall user experience of the site. Users will be shown the site for the first time, with no prior experience. The tasks involve navigating to specific areas of the system and performing tasks to gain feedback and compare the site against the functional and non-functional requirements. The stage will finish with an open response section, allowing users to expand on their experience of the system in this stage.

1. Please Navigate to the Venue Registration Page. How did you find this task? *

Mark only one oval.

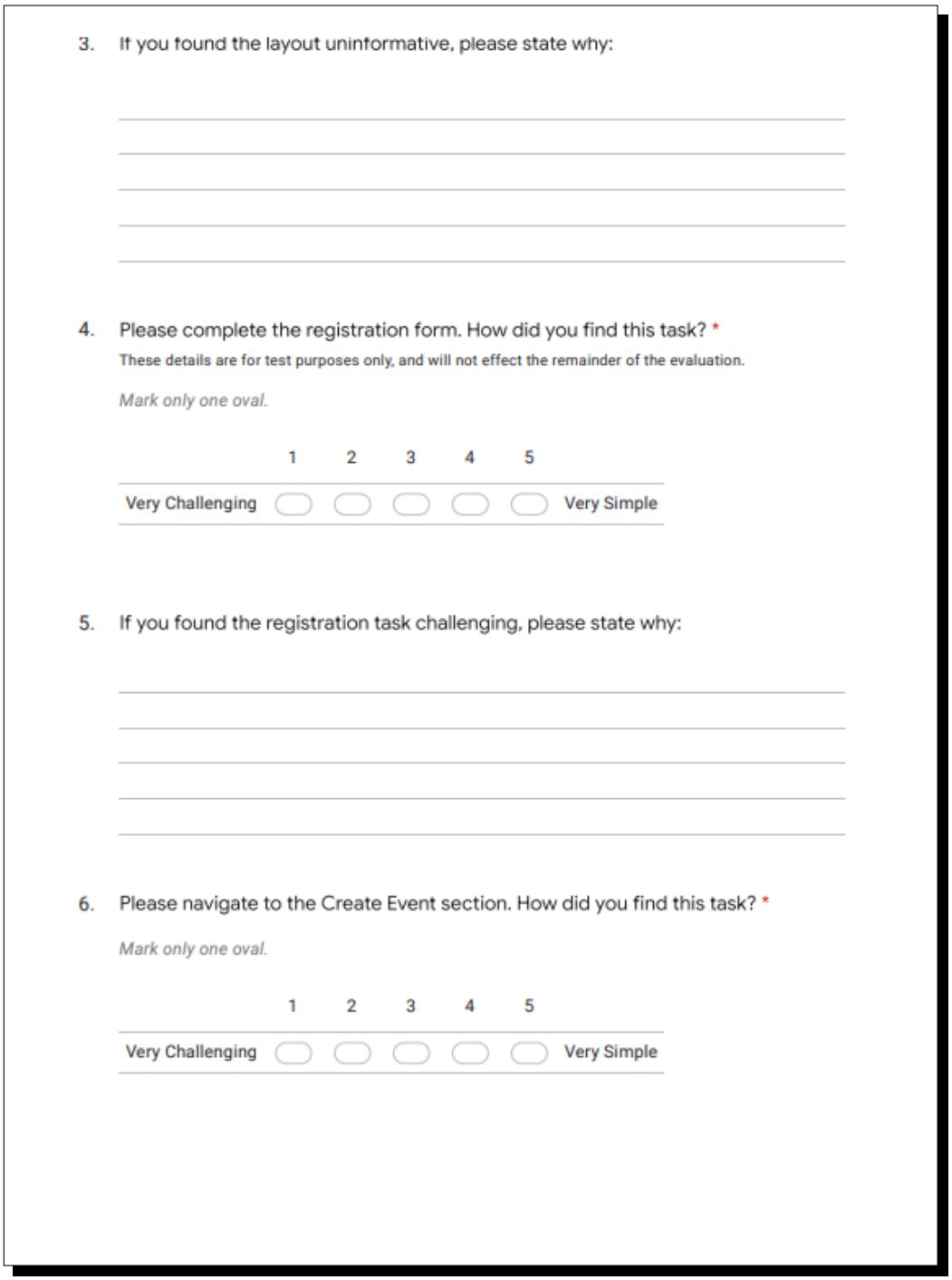
	1	2	3	4	5	
Very Challenging	◯	◯	◯	◯	◯	Very Simple

2. How would you describe the layout of the Register Page? *

Mark only one oval.

	1	2	3	4	5	
Very Uninformative	◯	◯	◯	◯	◯	Very Informative

Figure G.1: *Evaluation sheet, 1/12*



3. If you found the layout uninformative, please state why:

4. Please complete the registration form. How did you find this task? *
These details are for test purposes only, and will not effect the remainder of the evaluation.

Mark only one oval.

	1	2	3	4	5	
Very Challenging	◯	◯	◯	◯	◯	Very Simple

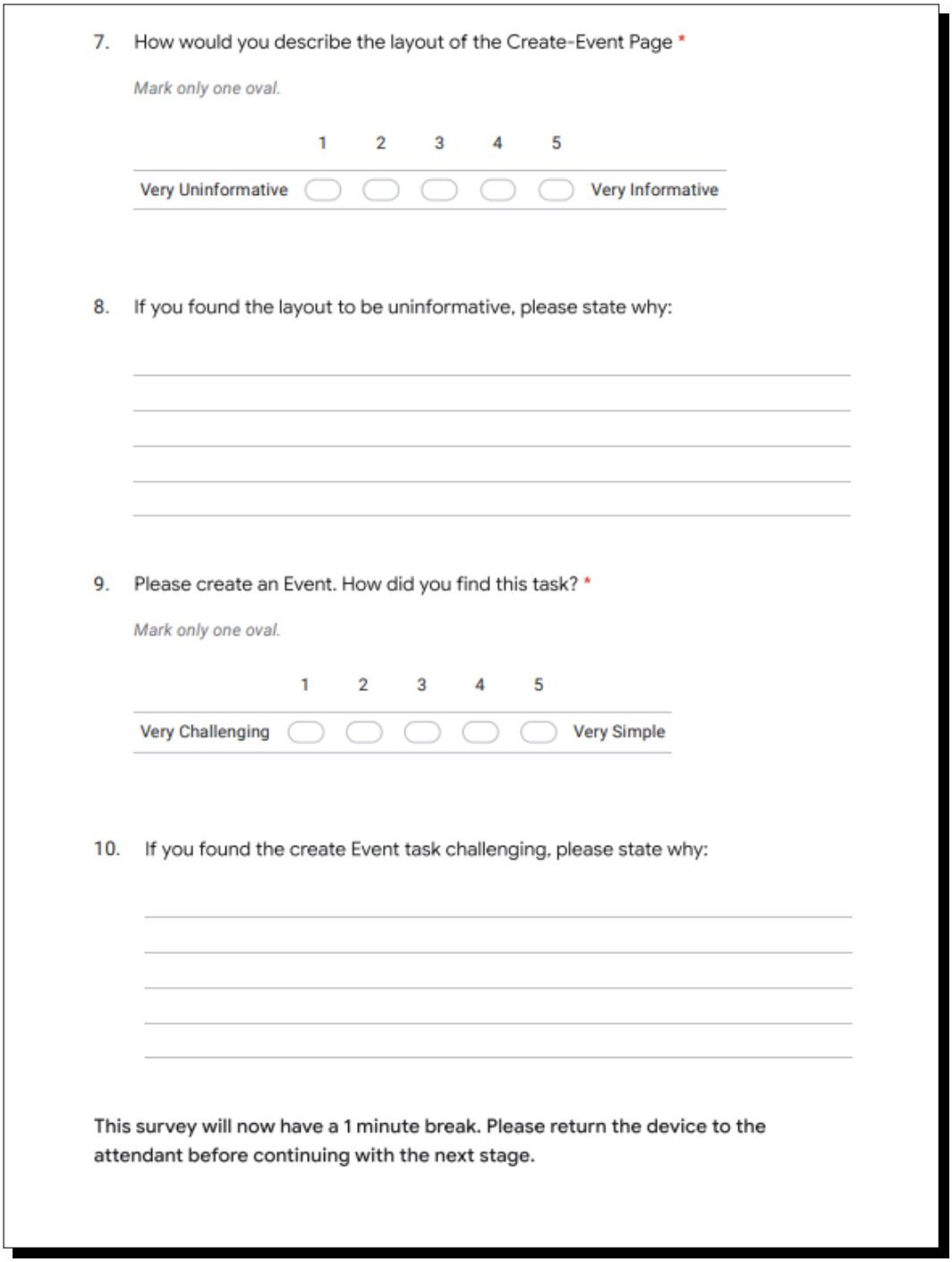
5. If you found the registration task challenging, please state why:

6. Please navigate to the Create Event section. How did you find this task? *

Mark only one oval.

	1	2	3	4	5	
Very Challenging	◯	◯	◯	◯	◯	Very Simple

Figure G.2: *Evaluation sheet, 2/12*



7. How would you describe the layout of the Create-Event Page *

Mark only one oval.

	1	2	3	4	5	
Very Uninformative	◯	◯	◯	◯	◯	Very Informative

8. If you found the layout to be uninformative, please state why:

9. Please create an Event. How did you find this task? *

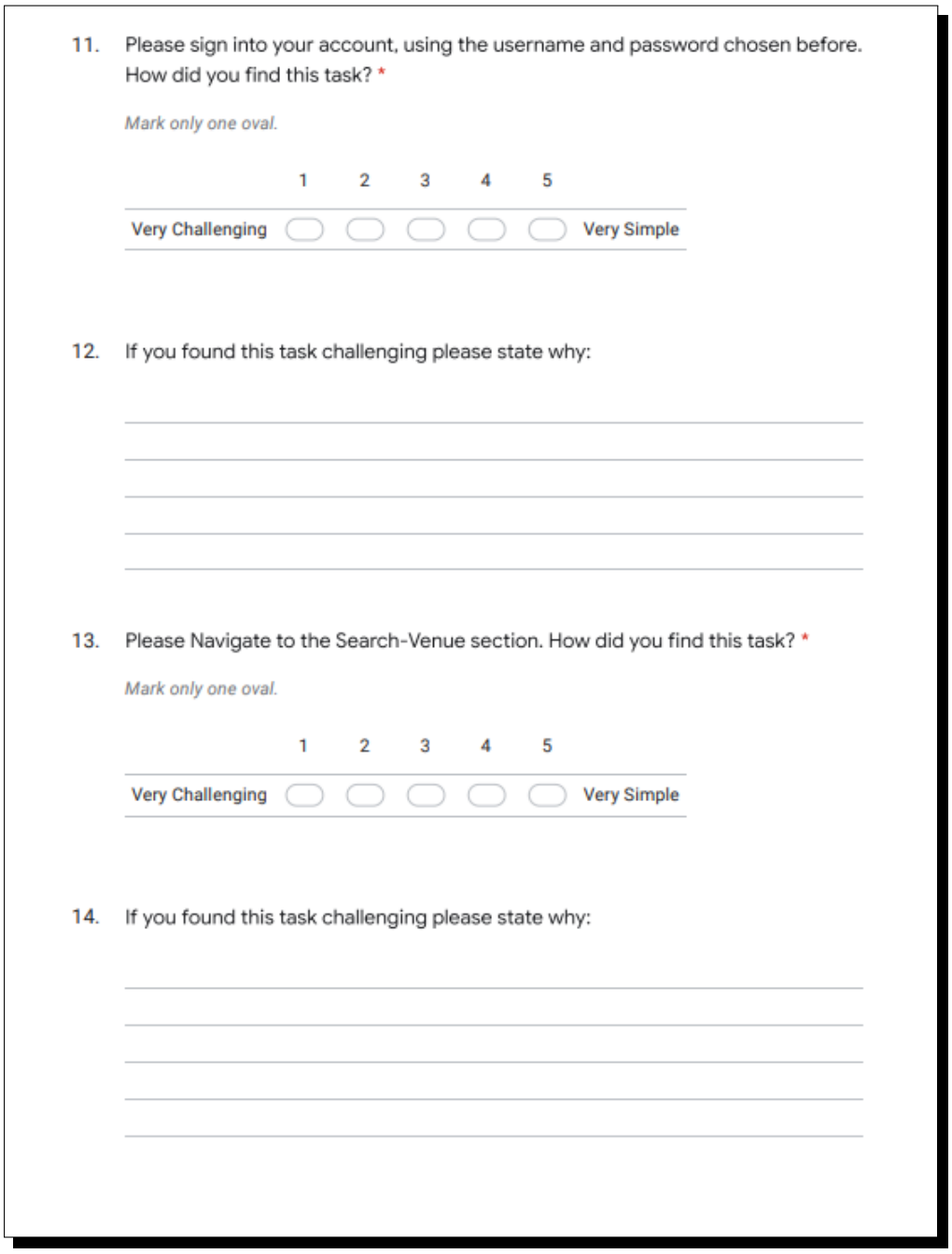
Mark only one oval.

	1	2	3	4	5	
Very Challenging	◯	◯	◯	◯	◯	Very Simple

10. If you found the create Event task challenging, please state why:

This survey will now have a 1 minute break. Please return the device to the attendant before continuing with the next stage.

Figure G.3: *Evaluation sheet, 3/12*



11. Please sign into your account, using the username and password chosen before. How did you find this task? *

Mark only one oval.

	1	2	3	4	5	
Very Challenging	◯	◯	◯	◯	◯	Very Simple

12. If you found this task challenging please state why:

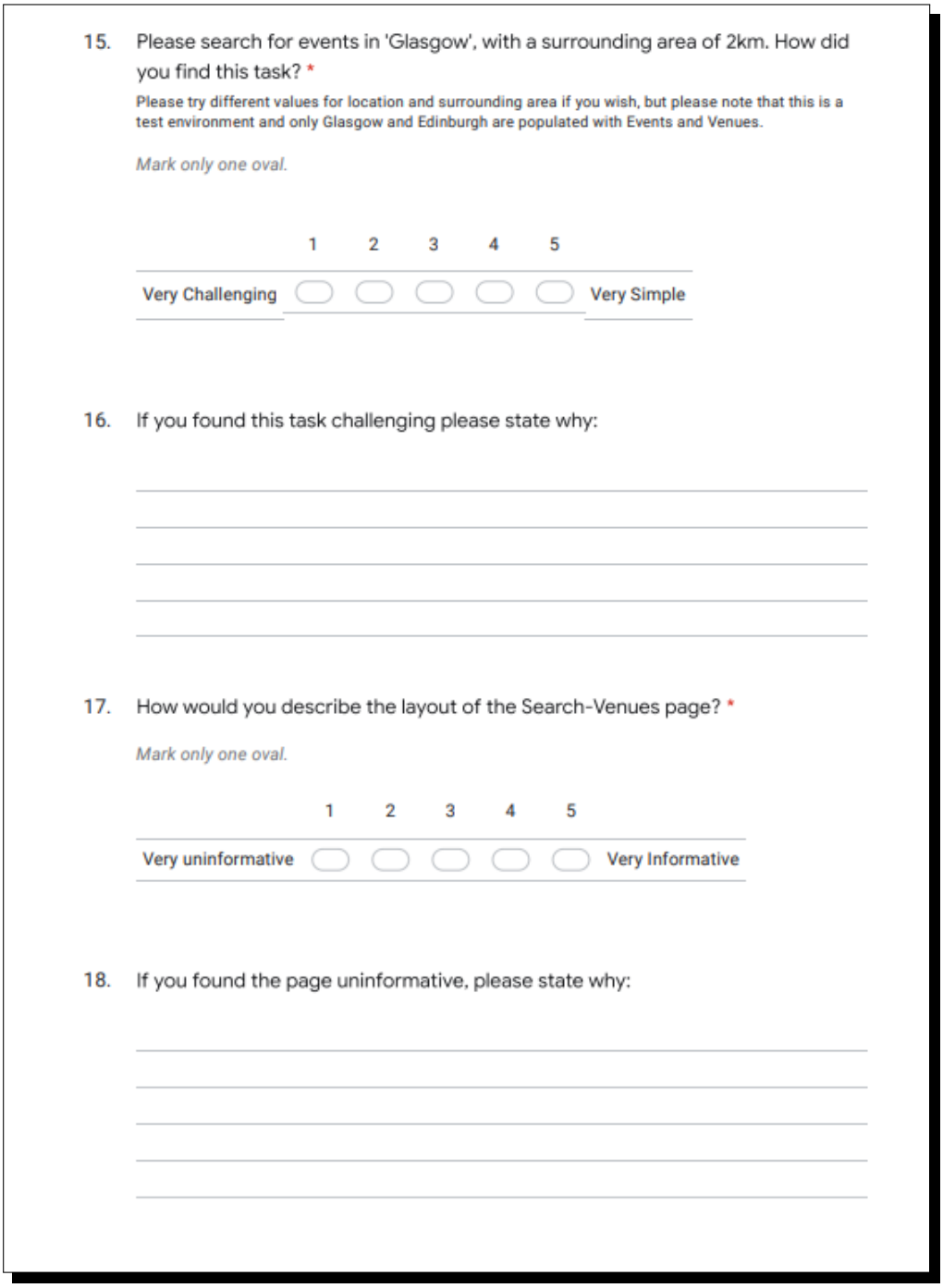
13. Please Navigate to the Search-Venue section. How did you find this task? *

Mark only one oval.

	1	2	3	4	5	
Very Challenging	◯	◯	◯	◯	◯	Very Simple

14. If you found this task challenging please state why:

Figure G.4: *Evaluation sheet, 4/12*



15. Please search for events in 'Glasgow', with a surrounding area of 2km. How did you find this task? *

Please try different values for location and surrounding area if you wish, but please note that this is a test environment and only Glasgow and Edinburgh are populated with Events and Venues.

Mark only one oval.

	1	2	3	4	5	
Very Challenging	◯	◯	◯	◯	◯	Very Simple

16. If you found this task challenging please state why:

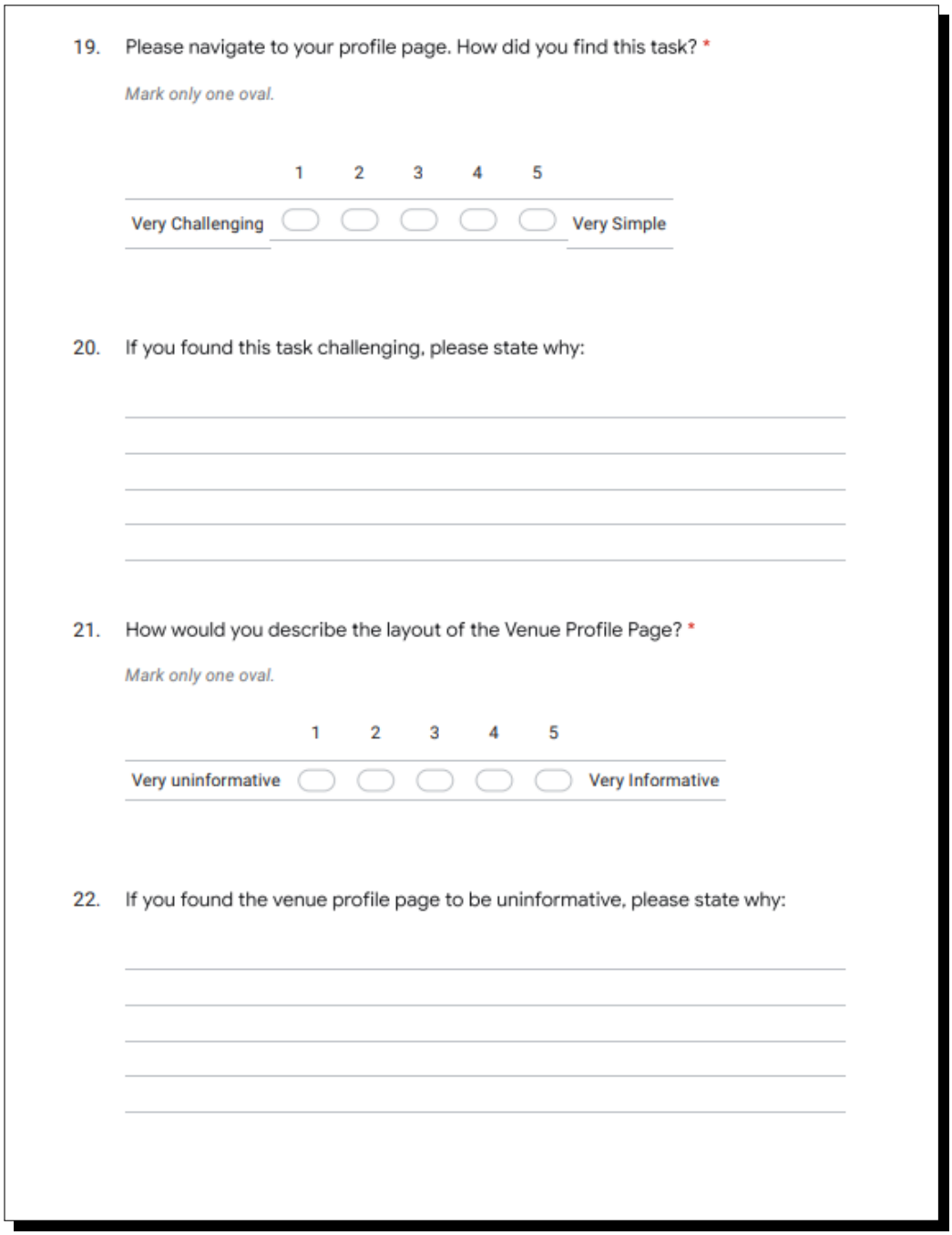
17. How would you describe the layout of the Search-Venues page? *

Mark only one oval.

	1	2	3	4	5	
Very uninformative	◯	◯	◯	◯	◯	Very Informative

18. If you found the page uninformative, please state why:

Figure G.5: *Evaluation sheet, 5/12*



19. Please navigate to your profile page. How did you find this task? *

Mark only one oval.

	1	2	3	4	5	
Very Challenging	◯	◯	◯	◯	◯	Very Simple

20. If you found this task challenging, please state why:

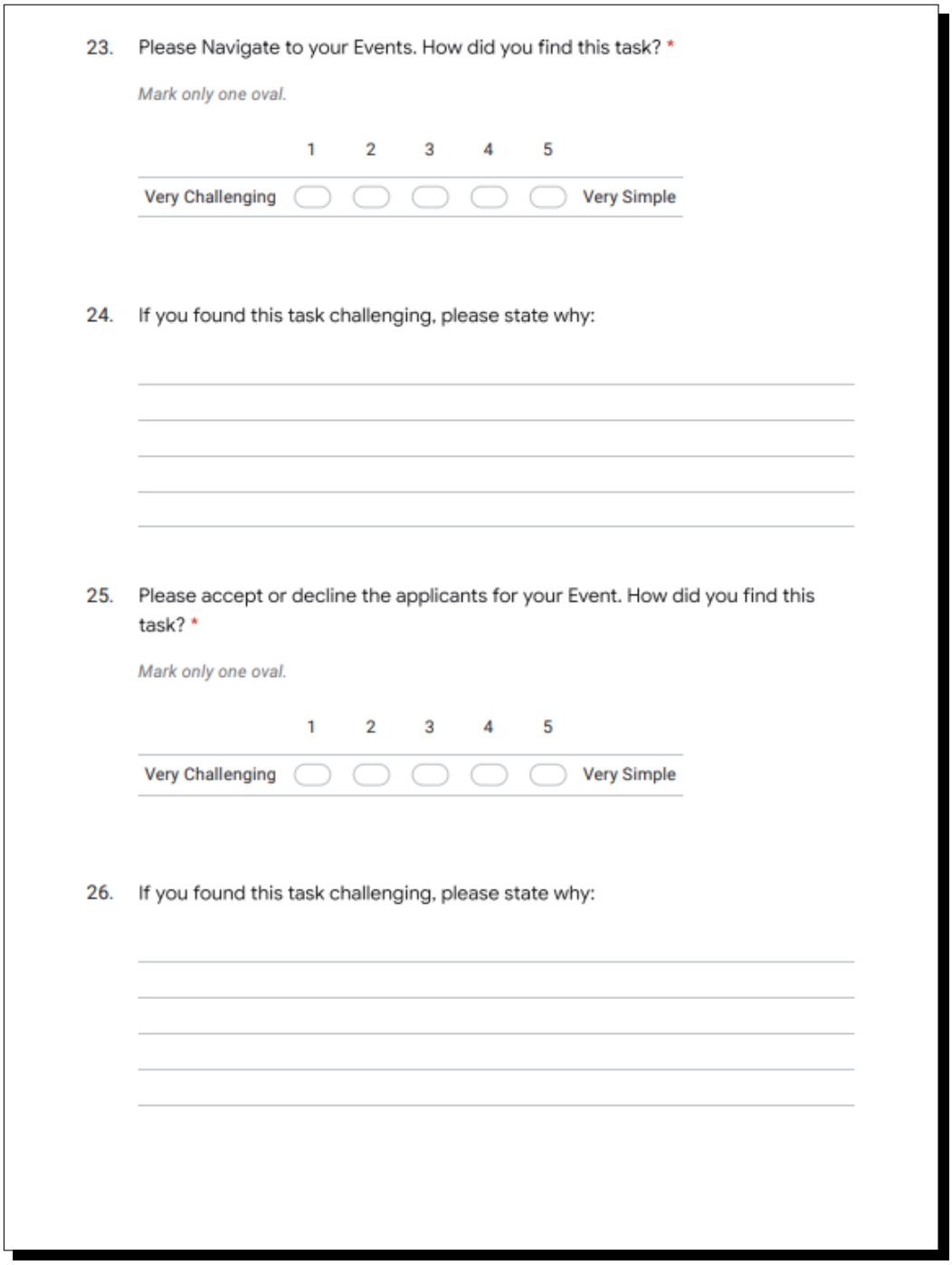
21. How would you describe the layout of the Venue Profile Page? *

Mark only one oval.

	1	2	3	4	5	
Very uninformative	◯	◯	◯	◯	◯	Very Informative

22. If you found the venue profile page to be uninformative, please state why:

Figure G.6: *Evaluation sheet, 6/12*



23. Please Navigate to your Events. How did you find this task? *

Mark only one oval.

	1	2	3	4	5	
Very Challenging	◯	◯	◯	◯	◯	Very Simple

24. If you found this task challenging, please state why:

25. Please accept or decline the applicants for your Event. How did you find this task? *

Mark only one oval.

	1	2	3	4	5	
Very Challenging	◯	◯	◯	◯	◯	Very Simple

26. If you found this task challenging, please state why:

Figure G.7: *Evaluation sheet, 7/12*

27. Please change your description on your account. How did you find this task? *

Mark only one oval.

	1	2	3	4	5	
Very Challenging	◯	◯	◯	◯	◯	Very Simple

28. If you found this task challenging, please state why:

29. Please Sign out of profile. How did you find this task? *

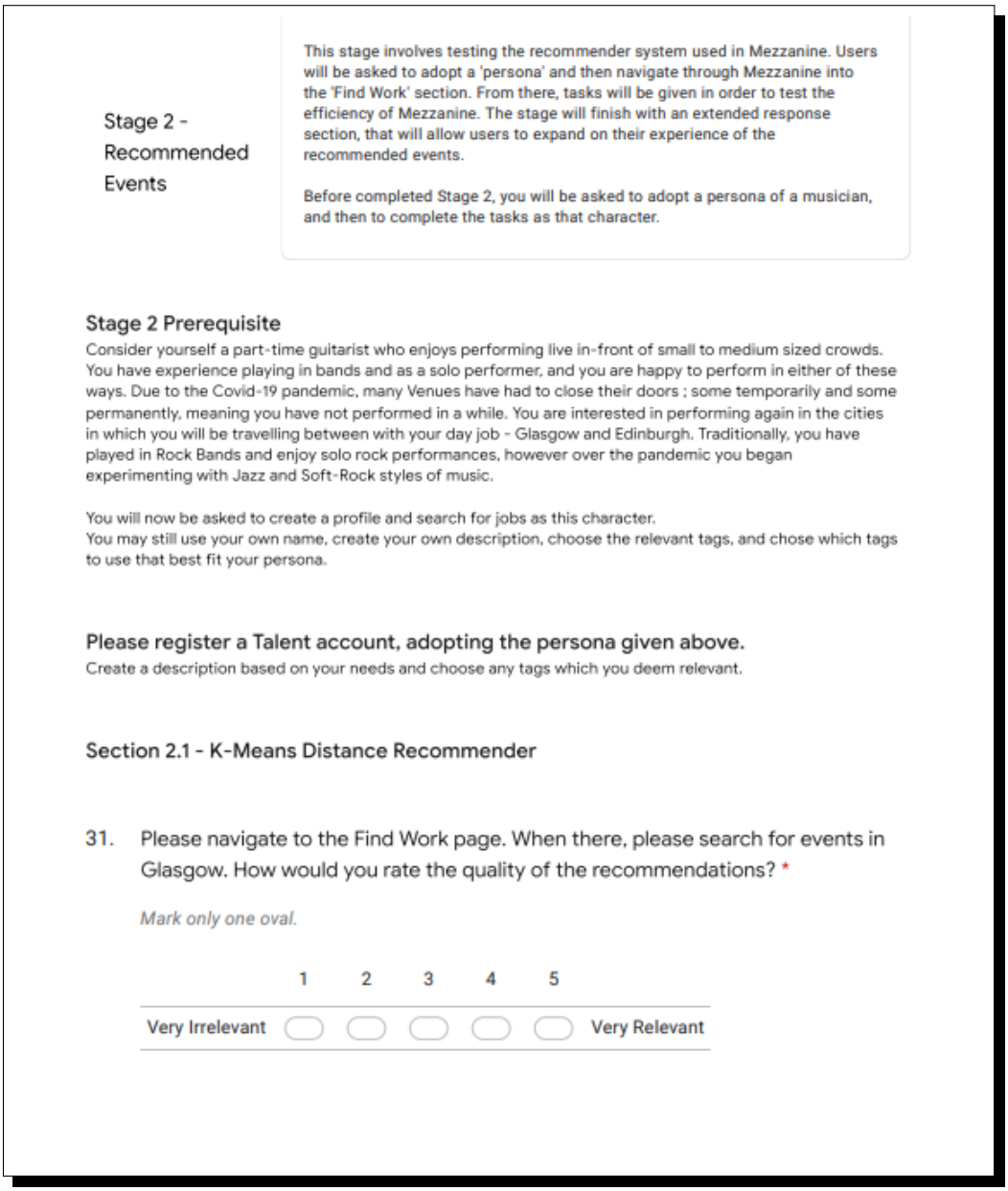
Mark only one oval.

	1	2	3	4	5	
Very Challenging	◯	◯	◯	◯	◯	Very Easy

30. If you have any additional comments about the lay-out of Mezzanine, please state here:

You have reached the end of Stage 1. Thank you for completing this section. Please move onto Stage 2 when you are ready.

Figure G.8: *Evaluation sheet, 8/12*



Stage 2 - Recommended Events

This stage involves testing the recommender system used in Mezzanine. Users will be asked to adopt a 'persona' and then navigate through Mezzanine into the 'Find Work' section. From there, tasks will be given in order to test the efficiency of Mezzanine. The stage will finish with an extended response section, that will allow users to expand on their experience of the recommended events.

Before completed Stage 2, you will be asked to adopt a persona of a musician, and then to complete the tasks as that character.

Stage 2 Prerequisite

Consider yourself a part-time guitarist who enjoys performing live in-front of small to medium sized crowds. You have experience playing in bands and as a solo performer, and you are happy to perform in either of these ways. Due to the Covid-19 pandemic, many Venues have had to close their doors ; some temporarily and some permanently, meaning you have not performed in a while. You are interested in performing again in the cities in which you will be travelling between with your day job - Glasgow and Edinburgh. Traditionally, you have played in Rock Bands and enjoy solo rock performances, however over the pandemic you began experimenting with Jazz and Soft-Rock styles of music.

You will now be asked to create a profile and search for jobs as this character.
You may still use your own name, create your own description, choose the relevant tags, and chose which tags to use that best fit your persona.

Please register a Talent account, adopting the persona given above.

Create a description based on your needs and choose any tags which you deem relevant.

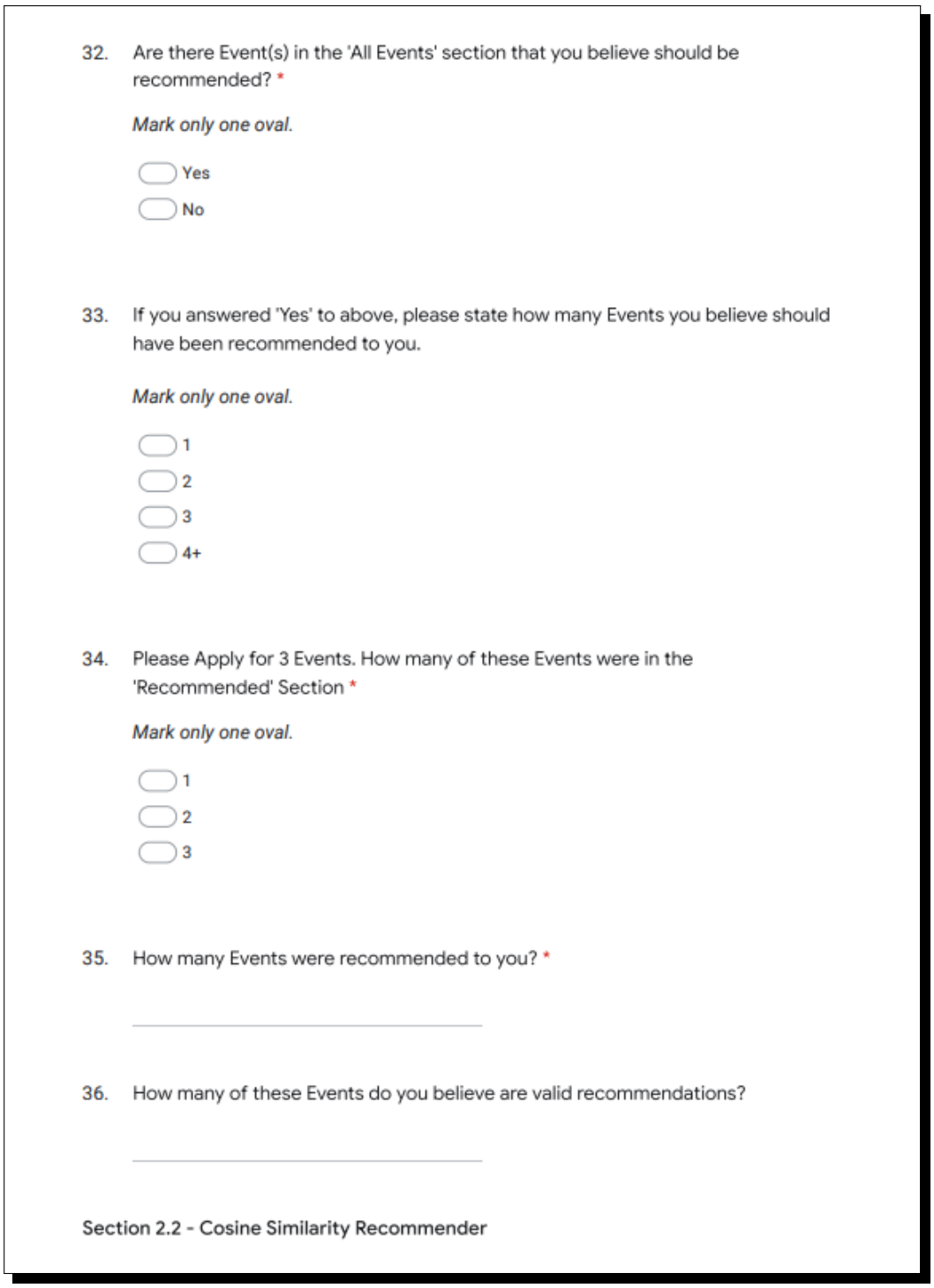
Section 2.1 - K-Means Distance Recommender

31. Please navigate to the Find Work page. When there, please search for events in Glasgow. How would you rate the quality of the recommendations? *

Mark only one oval.

	1	2	3	4	5	
Very Irrelevant	◯	◯	◯	◯	◯	Very Relevant

Figure G.9: *Evaluation sheet, 9/12*



32. Are there Event(s) in the 'All Events' section that you believe should be recommended? *

Mark only one oval.

- Yes
- No

33. If you answered 'Yes' to above, please state how many Events you believe should have been recommended to you.

Mark only one oval.

- 1
- 2
- 3
- 4+

34. Please Apply for 3 Events. How many of these Events were in the 'Recommended' Section *

Mark only one oval.

- 1
- 2
- 3

35. How many Events were recommended to you? *

36. How many of these Events do you believe are valid recommendations?

Section 2.2 - Cosine Similarity Recommender

Figure G.10: *Evaluation sheet, 10/12*

37. Please navigate to the Find Work page. When there, please search for events in Edinburgh. How would you rate the quality of the recommendations? *

Mark only one oval.

	1	2	3	4	5	
Very Irrelevant	◯	◯	◯	◯	◯	Very Relevant

38. Are there Event(s) in the 'All Events' section that you believe should be recommended? *

Mark only one oval.

- ◯ Yes
- ◯ No

39. If you answered 'Yes' to above, please state how many Events you believe should have been recommended to you.

Mark only one oval.

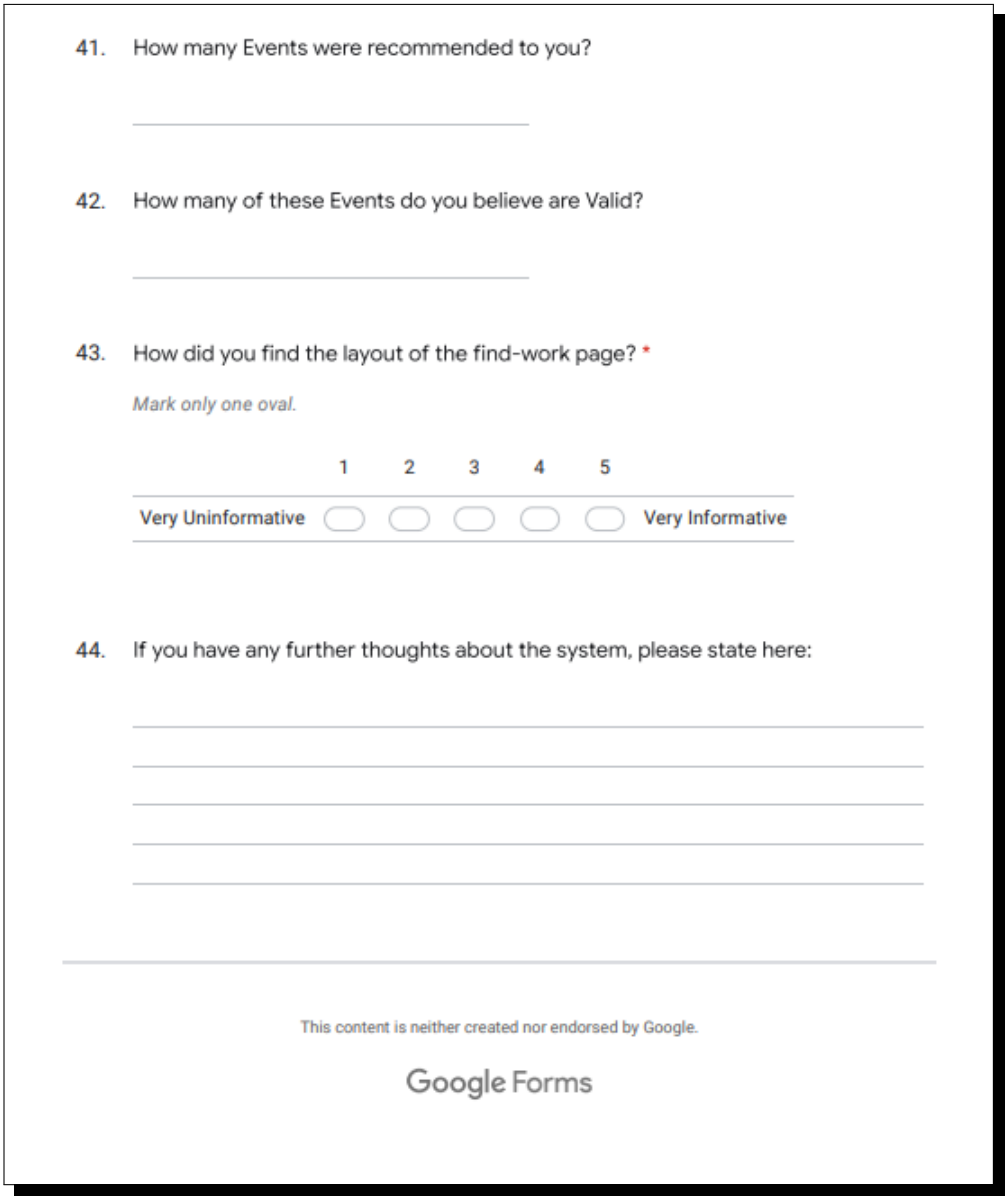
- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4+

40. Please Apply for 3 Events. How many of these Events were in the 'Recommended' Section *

Mark only one oval.

- ◯ 1
- ◯ 2
- ◯ 3

Figure G.11: *Evaluation sheet, 11/12*



41. How many Events were recommended to you?

42. How many of these Events do you believe are Valid?

43. How did you find the layout of the find-work page? *

Mark only one oval.

	1	2	3	4	5	
Very Uninformative	◯	◯	◯	◯	◯	Very Informative

44. If you have any further thoughts about the system, please state here:

This content is neither created nor endorsed by Google.

Google Forms

Figure G.12: *Evaluation sheet, 12/12*

Bibliography

Alfirna Rizqi Lahitani, Adhistya Erna Permanasari, N. A. S. (2020), 'Cosine similarity to determine similarity measure: Study case in online essay assessment', https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7577578&tag=1. Last accessed: 2022-06-03.

Alodadi, M. and Janeja, V. P. (2015), 'Similarity in patient support forums', https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7349760. Last accessed: 2022-03-11.

Alonso-Ríos, D., Mosqueira-Rey, E. and Moret-Bonillo, V. (2018), 'A systematic and generalizable approach to the heuristic evaluation of user interfaces', *International Journal of Human–Computer Interaction* **34**(12), 1169–1182.

Beri, A. (2020), 'Stemming vs lemmatization', https://towardsdatascience.com/stemming-vs-lemmatization-2daddabcb221. Last accessed: 2022-03-15.

Boone, H. N. and Boone, D. A. (2012), 'Analyzing likert data', *Journal of extension* **50**(2), 1–5.

Brownlee, J. (2017), 'A gentle introduction to the bag-of-words model', https://machinelearningmastery.com/gentle-introduction-bag-words-model/. Last accessed: 2022-03-16.

Dalianis, H. (2018), Evaluation metrics and evaluation, *in* 'Clinical text mining', Springer, pp. 45–53.

Glinz, M. (2007), On non-functional requirements, *in* '15th IEEE International Requirements Engineering Conference (RE 2007)', pp. 21–26.

Hesenius, M., Schwenzfeier, N., Meyer, O., Koop, W. and Gruhn, V. (2019), Towards a software engineering process for developing data-driven applications, *in* '2019 IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)', pp. 35–41.

Kuhn, J. (2009), 'Decrypting the moscow analysis', http://www.itsmsolutions.com/newsletters/DITYvol5iss44.pdf. Last accessed: 2022-03-14.

Lam, X. N., Vu, T., Le, T. D. and Duong, A. D. (2008), Addressing cold-start problem in recommendation systems, *in* 'Proceedings of the 2nd international conference on Ubiquitous information management and communication', pp. 208–211.

Likas, A., Vlassis, N. and Verbeek, J. J. (2003), 'The global k-means clustering algorithm', *Pattern recognition* **36**(2), 451–461.

Log, S. (2009), 'Precision, recall, f1 score intuitively explained', https://www.youtube.com/watch?v=8d3JbbSj-I8. Last accessed: 2022-03-14.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. and Brinkkemper, S. (2016), The use and effectiveness of user stories in practice, *in* 'International working conference on requirements engineering: Foundation for software quality', Springer, pp. 205–222.

Madhulatha, T. S. (2012), 'An overview on clustering methods', *arXiv preprint arXiv:1205.1117* .

Menzli, A. (2020), 'Tokenization in nlp: Types, challenges, examples, tools', https://neptune.ai/blog/tokenization-in-nlp. Last accessed: 2022-07-03.

SAS (2022), 'Natural language processing (nlp) what it is and why it matters', https://www.sas.com/en_us/insights/analytics/what-is-natural-language-processing-nlp.html. Last accessed: 2022-03-08.

Shung, K. P. (2018), 'Accuracy, precision, recall or f1?', https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. Last accessed: 2022-03-15.