University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Seminar Roulette: An Event Recommendation Platform for University Academics

Ollie Gardner

April 9, 2021

Abstract

The University of Glasgow does not offer a software solution that adopts a unified approach to seminar organisation and management. Current systems are ineffective in aiding time-poor academics to find new research seminars that they have not thought about attending before. Furthermore, current platforms do not include any personalised recommendation features. Seminar Roulette is a new web-based platform that scrapes seminar data from various University of Glasgow event feeds. It utilises an algorithmic recommender system to provide academics with personalised seminar suggestions based on how they have rated seminars that they have previously attended. An extensive evaluation was conducted, and it was concluded that Seminar Roulette effectively provides highly valuable event recommendations to members of the University of Glasgow academic community.

Acknowledgements

I would like to thank my supervisor, Dr Jeremy Singer, for his guidance and support throughout the duration of this work. I would also like to thank the participants who completed my user evaluations as without them, this project would not have been possible.

Education Use Consent

I hereby grant my permission for this project to be stored, distributed and shown to other University of Glasgow students and staff for educational purposes. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Signature: Ollie Gardner Date: April 9, 2021

Contents

1 Introduction

This chapter will introduce the Seminar Roulette project. It will investigate the need for a web-based platform that enables members of the University of Glasgow academic community to discover research seminars that they are unaware of. Subsequent chapters in this paper will document the project's background, requirements, design, implementation, evaluation and conclusion.

1.1 Motivation

Every week, the University of Glasgow host an average of 15.4 (std. dev. 2.7, 169 seminars) research seminars that are attended by staff and students (see Figure 1.1). Due to the nature of their work, seminar attendees are time-poor and do not want the hassle of trawling through lists of events to find the ones which are of interest to them. Academics are often unsure if a seminar will be of interest to them or not, so they will choose not to attend. Furthermore, their diary schedule changes frequently, making it difficult to commit to the same seminars each week.

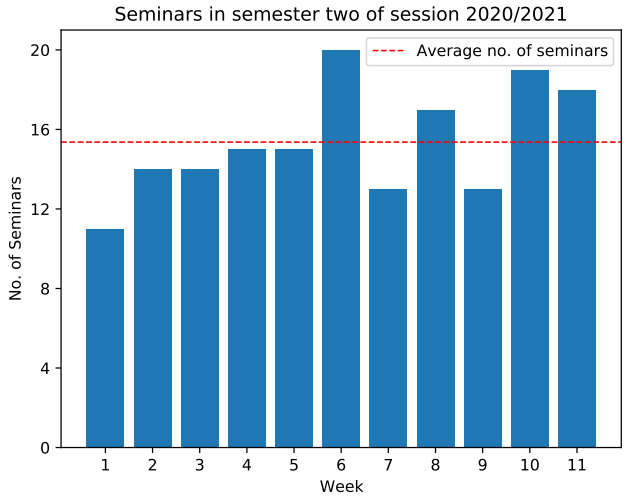


Figure 1.1: *A column chart showing the number of research seminars taking place each week at the University of Glasgow during semester two of session 2020/2021.*

Currently, seminars and events at the University of Glasgow are advertised through various platforms. The most popular of these sources being; Samoa Events (Samoa Events 2021), Eventbrite (Eventbrite 2021) and the University of Glasgow website (University of Glasgow 2021a). These systems will be analysed in-depth in Chapter 2. By having a mix of both internal and external sources, it can be an unattractive prospect for academics to spend the time to find a seminar that is of interest to them. They also must ensure that it fits around their timetable. User interaction with advertised events on the aforementioned platforms is minimal, so their feature sets are limited.

For example, there is no way for academics to rate a seminar for other people to see. Furthermore, Samoa Events and the University's website do not offer personalised recommendations to users that are tailored towards their interests.

Recognising that an opportunity exists for a system that adopts a unified approach to seminar management, the prospect of a new application can start to be envisaged. This development will overcome the weaknesses of the platforms currently accessible to academics. Additional functionality will be added to the application which is presently unavailable on a University of Glasgow platform. There is potential for the new system to become a staple piece of software in the University's seminar community and be rolled out campus-wide.

1.2 Aims

This project proposes Seminar Roulette, a new web-based platform that will allow University academics to discover research seminars that are of interest to them. The web application will be accessible to the whole University community and it endeavours to become the centralised source for seminar information. By being available to the entirety of the University, the system must support various types of users - staff members, students and researchers.

Seminar Roulette intends to provide academics with the ability to view events taking place at the University through one portal without having to browse various different websites. They will be able to login to the system using the University's single sign-on service, Shibboleth (Shibboleth 2021). Users will be able to enter their personal interests and rate seminars that they have previously attended. In response, Seminar Roulette will present them with recommendations based on past seminars which they have rated highly. Furthermore, a seminar's description will be analysed, with high-frequency words extracted, forming a set of keywords. These words will then be matched up with a user's interests using an algorithmic recommender system.

Requirement elicitation will be carried out using a survey sent out to a subset of staff members and students. In particular, the questionnaire will target those who regularly attend seminars to build a picture of what is missing within the event platforms currently available. Seminar Roulette should be responsive and able to be used without any prior training. It should have a visually pleasing user interface in order to drive user engagement with the system. High uptake with the platform will result in users benefiting from highly valuable seminar recommendations. To ensure Seminar Roulette is accessible to the whole University community, the application should be responsive to mobile and tablets whilst having the same feature set as desktop devices. Enabling the system to be accessible on any device will result in users being able to find seminar events whilst on the move. The sole deliverable of this project will be a web application hosted on a virtual machine and accessible via a domain name using any web browser.

Before the development of this project can begin, various preliminary stages need to be conducted, including; a period of initial requirement gathering with avid seminar attendees, establishing the core functional and non-functional requirements of the application and creating initial paper prototypes for the user interface. Following this introductory stage of the project, the software implementation will begin using a set of modern technologies that will allow for the group of requirements to be achieved. Finally, the usability and suitability of Seminar Roulette will be assessed through a questionnaire. The evaluation will collect two types of data - quantitative and qualitative. Quantitative data will be gathered through a Post-Study System Usability Questionnaire (PSSUQ) and will measure the end users' perceived satisfaction with the platform. Ideas of how Seminar Roulette could be improved in future iterations will be evaluated using five open-ended questions.

1.3 Summary

This chapter has introduced the motivation and aims of Seminar Roulette, comparing the project proposal with pre-existing services available to the University of Glasgow community. This paper will go on to outline how the motivation behind the project was developed into a working prototype. The subsequent chapters in this dissertation are structured as follows:

- **Chapter 2** discusses how recommender systems are used within society. It also examines and compares pre-existing related seminar products, identifying their weaknesses and how they differ from Seminar Roulette.
- **Chapter 3** outlines the requirement gathering process and analyses the findings to build a set of functional and non-functional requirements.
- **Chapter 4** explains the design process, highlighting how Seminar Roulette was constructed. Furthermore, each design decision will be explored in-depth to ensure that it links back to the project requirements devised in Chapter 3.
- **Chapter 5** describes the methods used to build and implement the project and explores the software engineering techniques utilised to ensure effective and timely delivery of the end product.
- **Chapter 6** details the processes carried out to evaluate the system, including; unit testing, system performance, two end-user evaluations and requirement validation.
- **Chapter 7** begins with a reflection of the overall success of Seminar Roulette. Next, the potential for future work and enhancements is discussed, and the paper concludes with a summary of the project's findings.

2 Background

This chapter will discuss the project's background, examining how algorithmic recommender systems are used within popular e-commerce platforms. Following this, related products will be investigated, reviewing the drawbacks of these applications and how the new system will overcome them. Both internal and external software solutions to the University will be considered so that the aims set out in Section 1.2 can be achieved.

2.1 Recommender Systems in E-Commerce

The e-commerce industry is growing by 23% every year, attracting billions of monthly customers (Djuraskovic 2021). It is common for users of these platforms to become susceptible to information overload. Recommender systems overcome this issue through the use of custom user experience by searching through dynamically constructed data to display personalised content (Isinkaye et al. 2015).

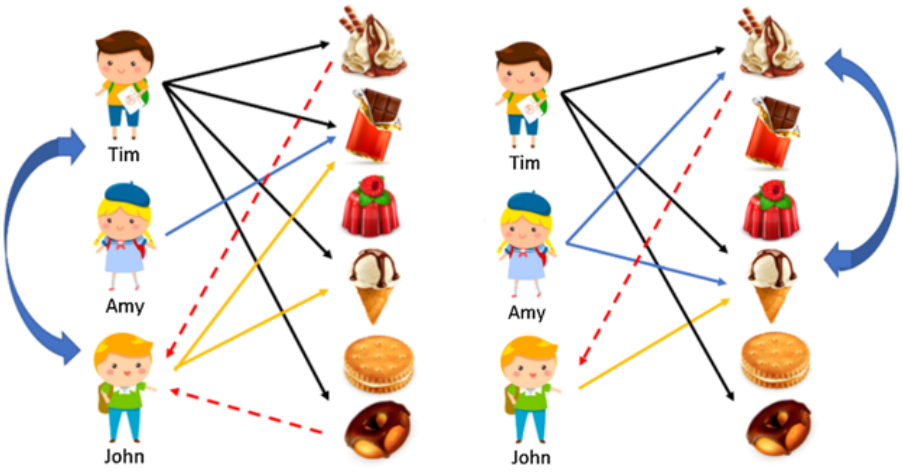
In Sections 2.1.1 and 2.1.2, Netflix and Amazon will act as case studies of e-commerce companies that utilise recommender system algorithms within their platforms. The products will be analysed, showing the different techniques of providing recommendations and how they can be applied to Seminar Roulette to enable academia to benefit from recommendation engines.

2.1.1 Netflix

As of 2020, Netflix has over 203 million paid subscribers, making it one of the largest streaming platforms in the world (Stoll 2020). A key feature of Netflix is their recommender system which allows their members to find videos to watch which they have not seen before. Research conducted by Netflix in 2016 revealed that people lose interest in films and TV shows after just 60 to 90 seconds after looking at anywhere between 10 and 20 shows (Gomez-Uribe and Hunt 2016). Recommendation is so vital to Netflix that in October 2006, they released a data set containing 100 million movie ratings and tasked the world with developing an algorithm that could beat the accuracy of its own (Bennett et al. 2007). If anyone could develop such a system, there was a million-dollar prize to be won.

Netflix implement multiple strategies for recommending movies to their customers, nicknamed; *Personalised Video Ranker*, *Top-N Video Ranker* and *Video-Video Similarity* (Gomez-Uribe and Hunt 2016). The most noteworthy of these is their *Personalised Video Ranker* algorithm which uses item-based collaborative filtering to provide recommendations. This technique is used for the content which cannot be described by metadata. A database of user ratings is created, turned into a user-item matrix and similarities between user ratings are calculated to make movie recommendations (Isinkaye et al. 2015). A high-level representation of item-based collaborative filtering is pictured in Figure 2.1b.

Due to the new platform being similar to Netflix but using events instead of movies and TV shows, item-based collaborative filtering could be applied to give academics a personalised experience on the site. It is a tried and tested method that will result in Seminar Roulette providing valuable seminar recommendations to its users.



(a) User-based filtering. *(b) Item-based filtering.*

Figure 2.1: *Pictorial representation of user and item-based collaborative filtering using people and food. (a) shows user-based filtering - Tim's food choices have been recommended to John as they have both rated other foods similarly. (b) shows item-based filtering - an ice cream sundae has been recommended to John because he likes an ice cream cone. Figures adapted from Pipis (2020).*

2.1.2 Amazon

Amazon's website attracts 2.5 billion monthly visitors (Djuraskovic 2021) and as a result, the company has become well known for its personalisation features. The functionality has been a staple in Amazon's user experience for over two decades, resulting in every user viewing the platform differently. In 1998, Amazon released its item-based collaborative filtering algorithm, allowing product recommendations to be made at a scale never seen before (Smith and Linden 2017). As the company expanded from only selling books to every product one will ever need, their recommender algorithm had to be made smarter. Their solution to this was to suggest related products to users. For example, if a customer buys a camera, they are likely to purchase a memory card too. This is referred to as *item-to-item correlation*, where the recommender system has the ability to propose complementary products (Schafer et al. 1999).

Amazon deploys similar techniques to Netflix when it comes to collaborative filtering. They use item-based filtering to recommend products to customers based on their previous ratings. On top of this, Amazon also utilises *people-to-people correlation*, using a user's profile as a metric instead of an individual item. This is known as user-based collaborative filtering (see Figure 2.1a). To make recommendations, this method takes advantage of the *K-Nearest Neighbour* algorithm (see Figure 2.2) to find users who are most alike.

Input : $\mathcal{Q}$, a set query points and $\mathcal{R}$, a set of reference point;
Output: A list of k *reference* points for each *query* point;

1 **foreach** *query point* $q \in \mathcal{Q}$ **do**
2 **compute** distances between q and all $r \in \mathcal{R}$;
3 **sort** the computed distances;
4 **select** k-nearest reference points corresponding to k smallest distances;

Figure 2.2: *Pseudocode for K-Nearest Neighbour algorithm. Figure adapted from Arefin et al. (2012).*

Although Amazon benefits immensely from user-based filtering, it is unlikely that the new platform would be able to do the same. This is due to data sparsity within user's profiles, resulting

in a small number of similar items between users. The absence of data of this nature leads to unreliable similarity information and ultimately resulting in poor recommendations.

2.2 Related Products

The University uses a range of internal and external software solutions to handle seminar management, having similar goals to Seminar Roulette. This section includes a detailed review of the most popular systems, detailing the weaknesses of each which the new platform will overcome. Further, the reasoning behind how each of the products influenced the design of Seminar Roulette will be discussed.

2.2.1 Samoa Events

Samoa Events (Samoa Events 2021) is an event management system developed at the University of Glasgow by Andrew Ramsay (see Figure 2.3). The platform presents many functionalities by collating seminar data from various schools; Mathematics & Statistics, Psychology and Social Sciences. It is the first and only application in the University's software portfolio to provide a single central location for its users to discover events taking place at the University. Along with viewing, the system also allows academics to search for and edit events.

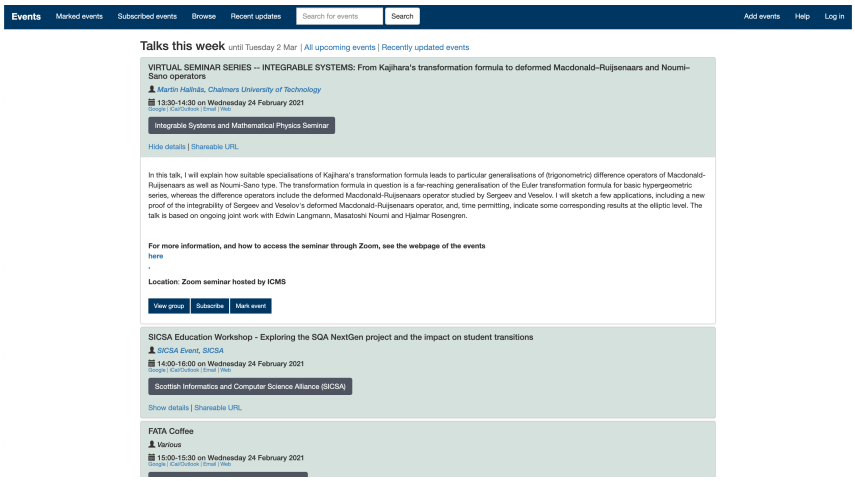


Figure 2.3: *Samoa Events homepage, showing upcoming seminars that are happening this week.*

The platform's main limitation is that it does not provide users with a personalised experience, resulting in each person viewing the same content. Plus, there is no functionality to rate events that academics have previously attended, resulting in the application not being able to learn what one's interests are. Although Samoa Events pulls in data from multiple locations, it does not collect information from external sources, such as Eventbrite. It solely focuses on internal systems, and consequently, the web application does not encapsulate all of the seminars and events hosted at the University.

The development of Samoa Events has been the first step in building a seminar community within the University. It is a prevalent tool for seminar organisation - on average, users view four pages per visit and spend ~five minutes on the platform (Alexa 2021). Seminar Roulette intends to include data from external sources, provide academics with personalised recommendations and learn what each user is interested in. Samoa Events has influenced the new platform's design by showing the extensive amount of event data it holds but does not further analyse.

2.2.2 University of Glasgow Website

The University of Glasgow website (University of Glasgow 2021a) provides each school with its own designated page to post upcoming seminars and events. The site is a natural location for staff members and students to look at to find seminars that are of interest to them. Academics are often unaware of other services available to them and accept this as the go-to place for event inspiration.

As pictured in Figure 2.4, the University's website does not provide a user-friendly method of finding out about seminars taking place. The web pages are static and do not show personalised information to the user. As events are displayed by school, it is not easy to explore the complete picture of seminars taking place at the University. The website does not allow users to see all events in one place, resulting in a frustrating user experience. This does not aid time-poor academics in identifying seminars that they have not thought about attending before. Furthermore, users cannot advertise their own seminars, and it is unclear how they are posted onto the website.

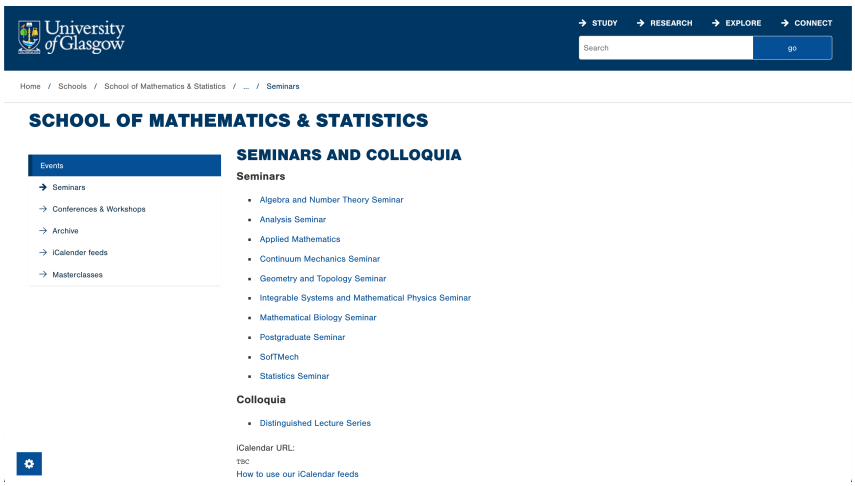


Figure 2.4: *School of Mathematics & Statistics' seminar page on the University of Glasgow website.*

The new platform will endeavour to show data from various sources in one central location and be as interactive as possible. In turn, this will allow for a pleasant user experience and enable time-poor academics to explore events seamlessly. The University's website has acted as an example of what not to do when it comes to the design and development of Seminar Roulette. The website has not been created with event management in mind and has to display a variety of information. The new platform will solely focus on seminars and events, with user experience being a top priority.

2.2.3 Eventbrite

Whilst platforms internal to the University have been analysed in Sections 2.2.1 and 2.2.2, this section will discuss an external product, Eventbrite. Eventbrite is an event management and ticketing website which allows organisers to create events and promote them on the platform (Eventbrite 2021). Users can browse for events taking place in their local area. In 2019, 4.7 million events spanning 180 countries were advertised on Eventbrite (Perez 2020). It is dominating the online event industry with a 62% market share (Datanyze 2021).

The major drawback of Eventbrite is that it was not specifically designed to advertise research seminars. It is a global platform, so it must encapsulate the whole of the event industry's requirements. When used by the University's academic community, they do not have a tailored seminar

experience. Although Eventbrite contains a recommender system, the algorithm will suggest events that are unrelated to academics. This results in students and staff members having to spend the time finding seminars that are academic-related and hosted within the University. This goes against the project's motivations outlined in Section 1.1, where it was stated that *"academics are time-poor and do not want the hassle of trawling through lists of seminars to find one that is of interest to them"*.

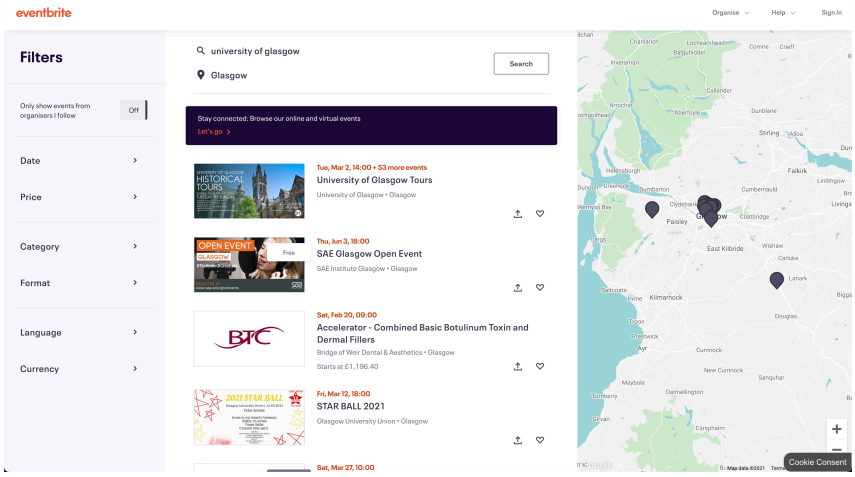


Figure 2.5: *Eventbrite's user interface, displaying results following a search for events relating to the University of Glasgow.*

Seminar Roulette aims to address this by only displaying academic-related events and would be of interest to the University community. Eventbrite provides its users with suggestions, but they are not tailored towards University events. The new application's recommender system will achieve this by deploying similar techniques but ensuring that only events hosted at the University are recommended to users. Seminar Roulette will use Eventbrite as a data feed, being the first University application with the functionality to display events from an external system. Eventbrite's user interface is minimalist, showing only the most essential information (see Figure 2.5). The new platform will replicate this, having a clean and straightforward frontend as seen on Eventbrite.

2.2.4 University of Cambridge Talks

In Section 2.2.1 to 2.2.3, systems used by the University of Glasgow for handling event management have been analysed. In this section, the University of Cambridge's in-house web application, University of Cambridge Talks or Talks.cam for short, will be examined (see Figure 2.6). Talks.cam was developed in 2006 by Tom Counsell after Cambridge struggled to find a way of collating and circulating lists of events happening at the University (University of Cambridge 2006).

Talks.cam is primarily related to Samoa Events in terms of feature sets, showing that different universities have similar strategies for event organisation and management. The main difference between the platforms is that Talks.cam displays talks taking place at the University of Cambridge, whilst Samoa Events shows events happening at the University of Glasgow. Furthermore, Talks.cam is more advanced in terms of its recommendation features. Although it does not have a recommender system in place, the application can send users daily or weekly emails reminding them about upcoming talks that they might be interested in. A convenient feature that Talks.cam has implemented is the ability to view an upcoming talk and enter a friend's email address so that they can become informed about the event. Time-poor academics typically use their email inbox

as reminders, and so this can benefit them in finding seminars that are of interest to them. This feature has aided Cambridge in creating a social aspect to University events, enabling people to go to the same seminars as their friends.

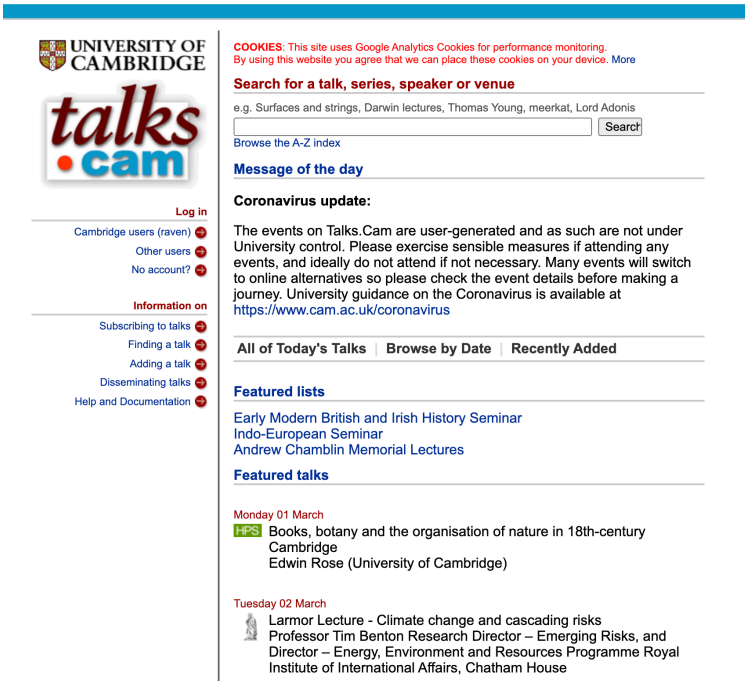


Figure 2.6: *University of Cambridge Talks homepage, displaying featured and upcoming talks.*

The main disadvantage of Talks.cam is that it does not give its users the ability to sort or filter upcoming events. It is relatively basic in terms of this functionality and does not allow one to amend the upcoming talks section to fit one's requirements. Whilst Samoa Events also does not offer this functionality, users can filter by seminar group, such as College of Arts, which allows them to view seminars hosted by that department. Seminar Roulette aims to use Talks.cam as inspiration of how another academic institution allows its cohort to browse upcoming events. The new system intends to build upon Talks.cam's current feature set, ensuring that the site is easy to use without any prior training.

2.3 Summary

This chapter examined how two large e-commerce companies, Netflix and Amazon, implement recommender systems into their platforms and how the same logic could be applied to Seminar Roulette. Internal and external software products to the University were also reviewed, outlining their shortcomings and showing that there is a need for an events recommender system within academic institutions.

3 Requirements Analysis

This chapter will outline the requirements gathering phase of the project, discussing the methods of elicitation. Before system development began, a set of functional and non-functional requirements were devised and prioritised. As features were implemented, requirements were re-prioritised taking into account the time constraints of this work.

3.1 Requirement Elicitation

Seminar Roulette's requirements were formed using two different strategies. The first of these was the development of user stories that acted as the high-level inspiration for the entirety of the project. Along with user stories, an initial user survey was distributed amongst various stakeholders to gauge what the University of Glasgow academic community wanted to see from an application of this nature. Upon completion of the gathering process, a deeper requirement analysis was performed.

3.1.1 User Stories

User stories were created for each user type - staff member, student and researcher. One of Seminar Roulette's aims was to ensure that the platform was available to the whole University community, resulting in each persona having the same feature set available to them.

Staff Member

- *As a* staff member, *I want to* be able to find seminars that I like and that fit around my timetable *so that* I can attend seminars that I might not have thought about attending before.
- *As a* staff member, *I want to* be able to login using the University's single sign-on service *so that* I can have a personalised experience on the platform.
- *As a* staff member, *I want to* be able to enter my personal interests *so that* the application can learn what I like and suggest seminars accordingly.
- *As a* staff member, *I want to* be able to choose the time at which I want to attend a seminar *so that* I can find seminars that are happening today, this week or this month.
- *As a* staff member, *I want to* be notified when a seminar is coming up *so that* I ensure that I do not miss it by mistake.
- *As a* staff member, *I want to* be able to choose the amount of randomness within my seminar suggestions *so that* I can find seminars that I like with an element of serendipity.

Student

- *As a* student, *I want to* attend seminars that are available to staff members *so that* I build upon my knowledge that I have learnt within my university courses.

Researcher

- *As a* researcher, *I want to* attend seminars that are relevant to my area of research *so that* I improve the work I undertake at the University.

3.1.2 Initial User Survey

An initial user survey was developed (see Appendix B) to elicit requirements from seven different stakeholders, including avid seminar attendees and the project's supervisor. The questionnaire consisted of 11 questions/statements designed to target each of the project's aims. Its goal was to act as a preliminary method of requirement gathering with a small subset of users to build upon the project supervisor's initial specification. The survey's intention was not to collect large amounts of data. Following is a summary of the responses received.

1. **On a typical week, how many seminars do you attend at the University?**
   The average number of seminars attended per week was 1.5, showing that the group of participants are frequent seminar attendees.

2. **What is your main area of research?**
   The majority of responses were either Computing Science or Software Engineering, which shows that there might be some bias in this survey towards the University of Glasgow's School of Computing Science. It should be noted that the list of requirements devised in Sections 3.2.1 and 3.2.2 may not represent the opinions of the whole University.

3. **Currently, we have two potential data sources for Seminar Roulette - Samoa Events and Eventbrite. Are there any other data sources which you are aware of that the university use for advertising seminars? If so, what are they?**
   Most of the answers to this question either stated the University of Glasgow website or mailing lists. Emails would be challenging to scrape data from, so it was decided that this source would not be utilised within Seminar Roulette.

4. **We would like to connect your calendar so that the system can suggest seminars that fit around your schedule. Which calendar service(s) do you use for work?**
   57% of respondents stated that they use Microsoft Exchange as their daily calendar application whilst working at the University of Glasgow.

5. **I think that it is important for the system to take into account my preferences/interests so that it can suggest seminars that I like.**
   86% of participants either *agreed* or *strongly agreed* with this statement. No participants *disagreed* or *strongly disagreed* with it.

6. **I would like there to be a feature that randomly suggests seminars to me.**
   There were some conflicting views with this statement. 14% of respondents would not like to see a random seminar feature implemented into the project.

7. **Once relevant seminar(s) have been suggested to you, would you like to be able to add them to your respective calendar?**
   The majority of participants said that they would like to be able to add a seminar to their calendar.

8. **Once relevant seminar(s) have been suggested to you, would you like to be notified before they start by email or otherwise?**
   Opinions on this question were split; 43% said *no*, and another 43% responded with *yes* they would like to be notified before a seminar starts. To resolve this, a setting could be implemented on the platform, allowing users to decide whether they would like to receive reminders.

9. **We endeavour to integrate the University's single sign-on service (Shibboleth) into the web application. Would you like to be able to use the system anonymously too?**

57% of people thought that it was essential to be able to use the system anonymously.

10. **Are you interested in attending seminars outwith your school/research area? For instance, would you be interested in attending Engineering seminars if you are a Computing Science academic?**
    71% of participants responded with *yes* to this question, and 29% were a *maybe*. Using this data, it was clear that Seminar Roulette must pull in data from varying academic subjects.

11. **Are there any additional features that you would like to see implemented into Seminar Roulette?**
    Many feature suggestions shaped the functional and non-functional requirements of the project. The most noteworthy being that the system should learn what a user likes from the seminars they attend and also the ability to automatically populate a user's calendar with events that fit around their schedule.

3.2 Prioritisation

Following on from the requirements elicitation process, a set of functional and non-functional requirements were formed. Once devised, they acted as a reference to ensure that Seminar Roulette was on track to achieve its initial goals. The outcome of this is evaluated in Chapter 6. To prioritise the requirements, the MoSCoW method was utilised, enabling requirements to be grouped by importance (Khan et al. 2015) and have one of the following meanings:

- *Must Have:* requirements that are imperative to deliver a viable software solution – the system would be unusable without them.
- *Should Have:* requirements that are important but not essential to ensure the success of the project.
- *Could Have:* requirements that are desirable but would not impact the final system much if not included.
- *Won't Have This Time:* requirements that will not be included in this prototype but are helpful to record so that the scope of the project in future iterations can be envisaged.

Each requirement was grouped, labelled and abbreviated. The groups mirror the different components of the web application. The label will be used to refer back to the requirement throughout this dissertation. The abbreviations; **MH**, **SH**, **CH** and **WH**, represent the priority levels; *Must Have*, *Should Have*, *Could Have* and *Won't Have This Time*.

3.2.1 Functional Requirements

Functional requirements define how the system should behave, and if not met, Seminar Roulette would be classed as unusable. They target product features and user requirements (Kurtanović and Maalej 2017).

System

S1 **MH** – The system must display information to users about research seminars taking place at the University of Glasgow.

S2 **MH** – The system must have an *I'm Feeling Lucky* button that displays a random seminar that a user has not attended before.

S3 **SH** – The system should have a *serves food* filter so that users can find seminars that provide refreshments.

S4 **CH** – The system could connect to a user's Microsoft Exchange calendar so that they can see seminars that fit around their schedule.

S5 **CH** - Each seminar could have a map showing its location.

S6 **WH** - The system will include a messaging/discussion feature where users can talk to each other about past and upcoming seminars.

User Experience

U1 **MH** - Users must be able to rate past seminars which they have attended.

U2 **MH** - Users must be able to view seminar recommendations based on how they have rated past seminars.

U3 **MH** - Users must be able to gain access to Seminar Roulette via a web browser.

U4 **MH** - Users must be able to filter seminars by a specific time frame, such as today or this week.

U5 **SH** - Users should be able to add seminars to their icalendar.

U6 **SH** - Users should be able to enter their personal interests and be shown a percentage match for each seminar based on how similar an event is to their interests.

U7 **SH** - Users should be able to search for a seminar using a search bar.

U8 **CH** - Users could be able to view seminars that their peers have attended in the past.

U9 **CH** - Users could be notified before a seminar starts so that they do not forget to attend.

U10 **CH** - When a user looks at a specific seminar, related events could be displayed underneath.

Authentication

A1 **MH** - Users must be able to login to the system using their University GUID and password through the single sign-on service.

A2 **SH** - Users should have the ability to use the system anonymously with a limited feature set.

A3 **WH** - Users outwith the University of Glasgow community will be able to login to the system.

Database

D1 **MH** - The database must have the ability to pull seminar data from a variety of sources.

D2 **MH** - The database must update seminar data frequently so that the most up to date information is shown on the system.

D3 **WH** - There will be a way for users to add, edit or remove seminars on the platform.

3.2.2 Non-Functional Requirements

The requirements outlined in this section are non-functional, having no direct effect on the platform. They are not essential to the functionality of the application. Instead, they describe how the system should behave, focusing on product properties and user expectation (Kurtanović and Maalej 2017).

System

S7 **MH** - The system must be responsive to all screen sizes, including mobile and tablet, without compromising on features.

S8 **MH** - The system must be able to scale to allow 40 000 people to use the platform. As of 2020, there are 8640 research/teaching staff (University of Glasgow 2020a) and 29837 students (University of Glasgow 2020b) at the University. Thus, 40 000 users is an upper bound, which the application must be able to handle to ensure that Seminar Roulette is available to the entirety of the University of Glasgow academic community.

S9 **MH** - The system must alert users to errors and how they can resolve them.

S10 **MH** - The system must be efficient and respond within two seconds. Research conducted by Google in 2017 showed that websites with load times longer than three seconds result in 53% of users being lost (business.com Member 2020).

S11 **MH** - The system must be able to function on the most common web browsers - Chrome, Safari, Firefox and Edge (Liu 2021).

User Experience

U11 **MH** - Users must be able to use the system without any prior training.

U12 **MH** - The user interface/experience must be intuitive and visually pleasing to drive user engagement.

Database

D4 **MH** - Personal data kept in the database must be stored in a GDPR compliant way. To adhere to this, the University's *GDPR compliance checklist for databases storing personal data* must be met (University of Glasgow 2021c).

3.3 Summary

This chapter described the requirements gathering process through various techniques, such as an initial user survey. User stories were developed to understand the stakeholders' wishes further. It also detailed how a set of functional and non-functional requirements were formed. These were then prioritised in terms of importance and grouped by system component using the MoSCoW method.

4 Design

This chapter will outline the high-level design of the product, discuss the system's architecture, how the user interface was formed, and investigate the crucial components of the platform. The list of requirements devised in Sections 3.2.1 and 3.2.2 will be referenced throughout, explaining how some of the requirements were satisfied through the application's design.

4.1 System Architecture

Seminar Roulette was designed using a three-tier architecture - a common approach when constructing a client-server web application. In this architectural pattern, components are structured into three logical and physical layers - presentation, application and data (Eckerson 1995). The main benefit of using this approach is that each tier can be developed independently, resulting in each layer being able to be run on a separate operating system or server platform. Other benefits of this architecture include improved scalability and reliability as each layer can be scaled without relying on another. Furthermore, if the development of Seminar Roulette was undertaken by a team, each tier could be implemented simultaneously by different groups of developers.

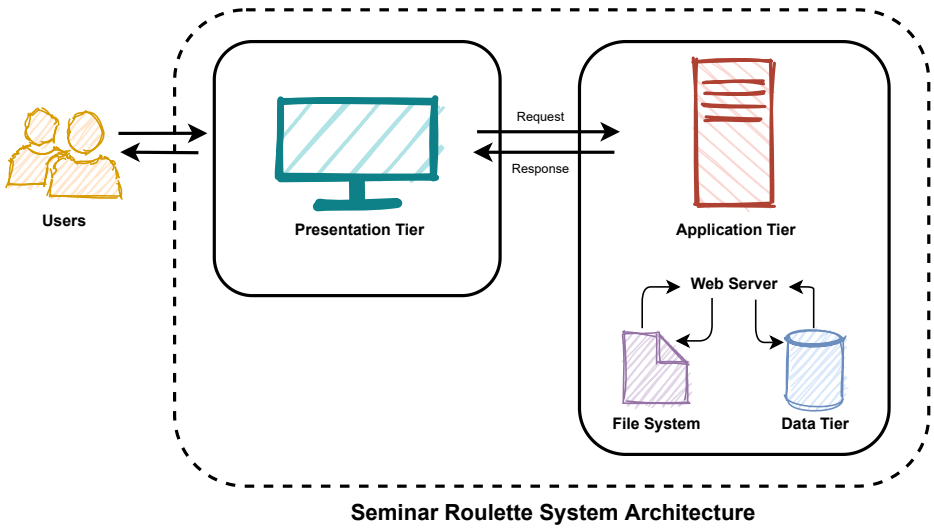


Figure 4.1: *System architecture diagram of Seminar Roulette, showing the three-tier approach. Figure adapted from Dabbs (2019).*

Prior to implementation, a visualisation of this architecture was devised (see Figure 4.1). The figure describes the three different tiers and how they interact with each other. The first of these being the presentation layer which was responsible for the user interface of the web application and allowed users to interact and communicate with the system's components. Next, the application layer acted as the core of Seminar Roulette and contained the platform's fundamental logic. This

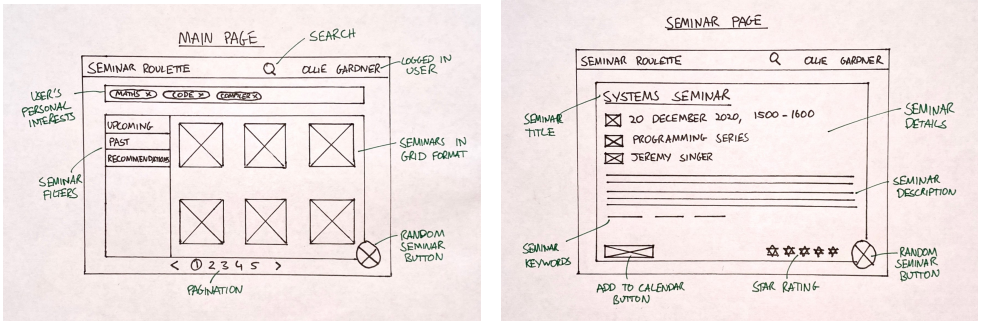
tier was responsible for communicating with the data tier through the use of RESTful API calls. Finally, the data layer contained the information stored by the application through the use of a database. The design of this layer could have been adopted using either a relational or NoSQL database approach. For Seminar Roulette, a relational database was chosen due to the large amount of data that had to be stored, identified and related to other models. In an application of this nature, data passes through the application tier, with the presentation and data tiers being unable to communicate directly.

4.2 User Interface

User interface design was a critical stage of the project, with 12 requirements focusing on the platform's user experience. Non-functional requirements *U11* and *U12* state that the application must be able to be used without prior training and have a visually pleasing user interface. To meet both of these requirements, the system had to be attractive and straightforward to use. The creation of the user interface went through various iterations before reaching the final high-fidelity prototype illustrated in Figure 4.3.

4.2.1 Low-Fidelity Prototypes

Before implementation commenced, low-fidelity prototypes were drawn in the form of paper sketches. The aim of these prototypes was to showcase a tangible representation of the final software product. The advantage of creating these first was that it enabled the app's different components to be visualised quickly onto paper. This stage's particular focus was the seminar cards that were responsible for displaying lots of information about events within a compact space. Paper prototypes enabled multiple iterations of these designs to be created with ease. In the first version of these sketches (see Figure 4.2), users could click on seminars and be taken to a new page. However, this was quickly scrapped for reasons outlined in Section 4.2.2.



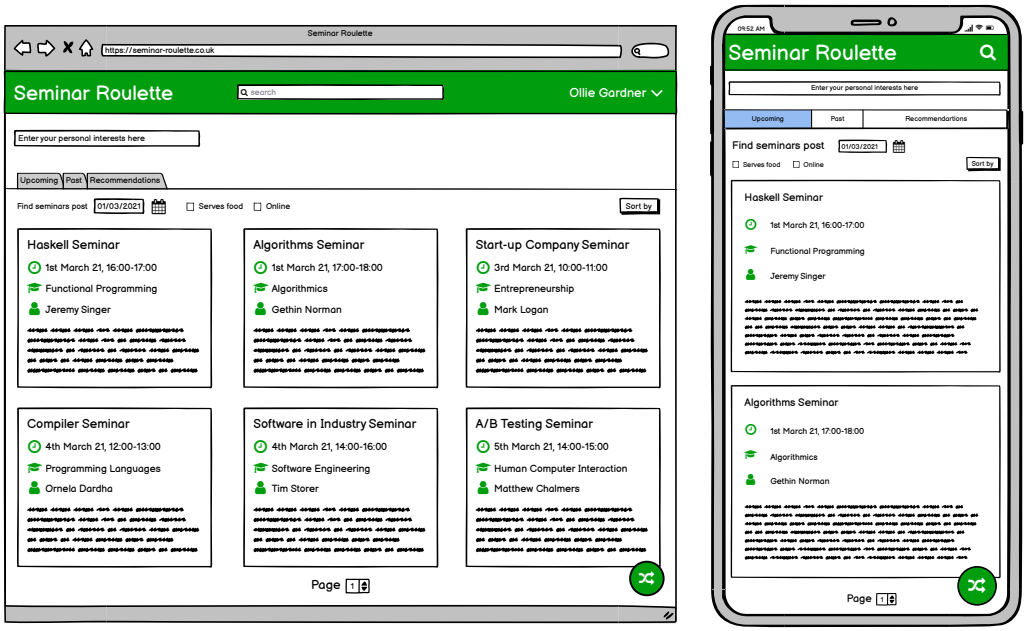
(a) *Paper prototype of the main page.* ***(b)*** *Paper prototype of the seminar page.*

Figure 4.2: *Low-fidelity paper prototypes of Seminar Roulette. (a) shows the main page of the platform where seminars can be viewed and filtered. (b) shows an individual seminar page, showing more details about the event. This page can be accessed by clicking on a seminar from the main page.*

Figure 4.2b displays a seminar's page that houses various components, such as an add to calendar button and the ability to leave a star rating. By including these details on this page, requirements *S1*, *U1* and *U5* were satisfied. Furthermore, the initial design of seminar filters and a search bar are shown, meeting requirements *U4* and *U7*.

4.2.2 High-Fidelity Prototypes

Following the creation of low-fidelity sketches, high-fidelity prototypes of Seminar Roulette were developed using Balsamiq, a wireframing tool (Balsamiq 2021). They intended to show the platform in much greater detail than the paper drawings and displayed an accurate representation of the final user interface design prior to implementation. Requirement *S7* was integral to this process as the platform had to be responsive to varying screen sizes. This resulted in wireframes being created for both desktop and mobile, allowing the differences between these views to be understood. As pictured in Figure 4.3, the two prototypes have the same feature set but are laid out differently. This was to ensure that the mobile version of Seminar Roulette did not compromise on the features available on the desktop site.



(a) *Desktop high-fidelity prototype.* ***(b)*** *Mobile high-fidelity prototype.*

Figure 4.3: *High-fidelity prototypes of Seminar Roulette created with Balsamiq. (a) shows the new platform on desktop and large tablets. (b) shows Seminar Roulette on a mobile device, ensuring that the site is responsive to small screen sizes whilst not compromising on features.*

By creating these prototypes, development time was decreased as the final look and feel of the platform was always clear. During the design process, it was decided that the system should have a minimal number of pages due to the target users being time-poor and finding navigating through multiple pages potentially frustrating. Furthermore, any components which asked for user input used pop-up modals to ensure that the page remained unchanged in the background. Therefore, a new page did not have to be loaded often. This aided requirement *U11*, where it was stated academics must be able to use the platform without any prior training.

4.2.3 Colour Scheme

To satisfy requirement *U12*, the colour scheme was a significant design decision to ensure that the user interface was visually pleasing. As shown in the high-fidelity wireframes in Figure 4.3, green was used as it has connotations of being soothing and relaxing (Won and Westland 2017). It was important for time-poor users to feel at ease when using the software, and hence green was chosen as the primary colour.

Post design, it was discovered that the University has its own brand toolkit and colour scheme (University of Glasgow 2021b). They encourage University systems, prints and digital media to use their colour palette to ensure consistency between their products. The colour *University Blue* is recommended as the primary colour. Seminar Roulette was designed using green, but if it were to be deployed for public use, the colour scheme would be changed to use *University Blue* to enable users to recognise that it is a University platform. As mentioned in Section 1.1, there is potential for Seminar Roulette to become a staple system in the University's software portfolio.

4.2.4 Accessibility

Although none of the requirements devised in Sections 3.2.1 and 3.2.2 stated that the application must be accessible to impaired users, it was still necessary to consider these types of users during the design process. For example, some users find that distressing motion on screens can trigger seizures, known as photosensitive epilepsy (Harding et al. 1994). To address this, Seminar Roulette was designed in such a way as to limit the amount of animation and flickering on the platform to reduce the likeliness of seizures.

It is estimated that around 253 million people in the world live with some sort of vision impairment (Tuchkov 2018). Colour blind users often find it difficult to distinguish between similar colour tones, resulting in a diminished user experience. The system was designed with an alternative dark colour scheme consisting of black and white to address this. People who are colour blind benefit immensely from contrasting colours with black and white creating one of the highest contrasts available. It was decided that users would be able to switch between the light and dark mode through a button located in the top application bar.

4.3 Database Design

As stated in Section 4.1, a relational database was chosen as the data collected and stored by Seminar Roulette required relationships between entities to be tracked. This allowed for personalised recommendations to be made to users through collating data on how they have rated past seminars. During the design phase, a database schema was created, and as the project matured, it was amended multiple times. The final iteration of this schema is illustrated in Figure 4.4.

The database contained seven tables. The purpose of each has been summarised briefly below:

- **Seminar** keeps track of a seminar's details and metadata, such as title, description, date and time.
- **Seminar Group** holds data about the department or group of people who are hosting the seminar. This table allows all seminars hosted by a particular organisation to be found.
- **Location** stores details about where the seminar is taking place and whether it is online or in-person.
- **Speaker** stores information on the person who is talking at the seminar. The speaker's website is kept to allow users to find out more about the individual prior to the seminar commencing.
- **Seminar History** allows for a user's seminar ratings to be stored within this model. This information is then used by the recommender system to make event suggestions to academics.
- **University User** holds information about the user who has logged into the system through the University's single sign-on.
- **Cronjob** is a model which keeps track of whether data has been pulled in successfully from Samoa Events and Eventbrite. This script was run nightly, and if it failed, the Cronjob table would store the error message so that it could be fixed swiftly.

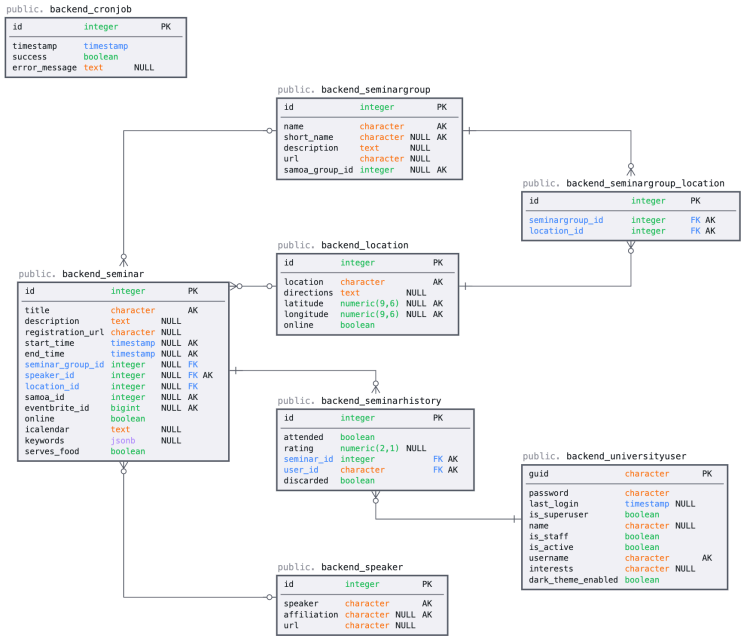


Figure 4.4: *Seminar Roulette's database schema, created by reverse engineering the final database. Entities and relationships are depicted using Crow's Foot Notation.*

4.4 Information Architecture

Organising and displaying the information on Seminar Roulette was imperative to provide a good user experience on the site. Without a logical arrangement of information, user experience could have diminished and resulted in the platform becoming cumbersome to use. Seminar Roulette was designed so that the user rarely had to leave the page they were currently on through pop-ups and expanding information on click. The application was simple in terms of information architecture; a sitemap is visualised in Figure 4.5. Structuring the information in this manner enabled the flow of the anonymous functionality to be pictured - users had to login using the University's single sign-on to reap all the benefits of Seminar Roulette.

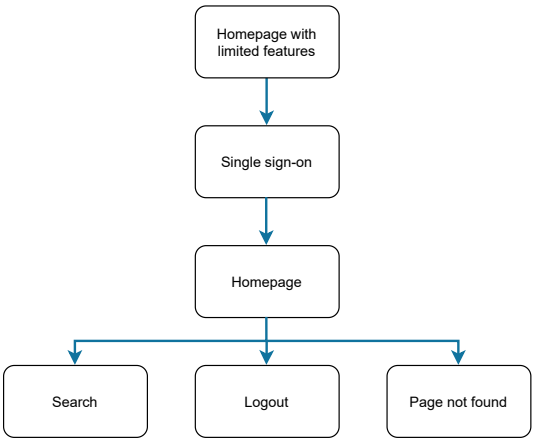


Figure 4.5: *Sitemap of Seminar Roulette, showing how the system was organised and arranged.*

4.5 Algorithms

An integral part of Seminar Roulette was its algorithms which enabled the system to make accurate seminar recommendations to users. In this section, two algorithms will be explored - Matrix Factorisation via Singular Value Decomposition and WuPalmer Similarity. Before implementation commenced, both algorithms had to be researched and chosen as the most suitable for their task. Seminar Roulette contained two levels of recommendation, and hence two algorithms were necessary. The first of these was the ability to suggest seminars based on user ratings, and the second allowed seminar suggestions to be made based on an academic's interests.

4.5.1 Matrix Factorisation via Singular Value Decomposition

Matrix Factorisation via Singular Value Decomposition is a type of collaborative filtering suitable for large data sets containing sparse matrices (Carleton College 2007). In Chapter 2, collaborative filtering techniques used by Netflix and Amazon were investigated. In these examples, neighbourhood algorithms were used to determine the distance between items and make product recommendations to users. The new system was designed so that users could rate a seminar from one to five. Using their ratings, future seminars could be suggested to the user similar to events they have previously rated highly.

The ideology behind matrix factorisation is that one matrix can be broken down into products of other matrices. In Singular Value Decomposition, an m by n matrix X is factorised into three matrices - U, S and V^T. The computational complexity of this calculation is $O(min(mn^2, m^2n))$. This process is shown in Figure 4.6.

$$\overset{X}{\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix}} = \overset{U}{\begin{pmatrix} u_{1,1} & \cdots & u_{1,r} \\ \vdots & \ddots & \vdots \\ u_{m,1} & \cdots & u_{m,r} \end{pmatrix}} \overset{S}{\begin{pmatrix} s_{1,1} & 0 & \cdots \\ 0 & \ddots & \vdots \\ \vdots & \cdots & s_{r,r} \end{pmatrix}} \overset{V^T}{\begin{pmatrix} v_{1,1} & \cdots & v_{1,n} \\ \vdots & \ddots & \vdots \\ v_{r,1} & \cdots & v_{r,n} \end{pmatrix}}$$

Figure 4.6: *Singular Value Decomposition process, adapted from Carleton College (2007).*

Matrix X represents all user seminar ratings, U is the users matrix containing a reference to all of the users in the database, S consists of singular values within a diagonal matrix and V^T stores information on the seminars. In other words, the U matrix represents whether a user rated a seminar highly or not, whilst a mapping of rating relevance to a seminar is stored in matrix V^T.

Table 4.1: *Example seminar ratings if there were three users and five seminars where five is the best possible rating that could be given. A rating of zero indicates that the user has not given that seminar a rating.*

Seminar / **User**	**1**	**2**	**3**	**4**	**5**
A	5.0	0.0	0.0	0.0	0.0
B	4.0	5.0	0.0	0.0	0.0
C	0.0	0.0	0.0	2.0	4.0

For example, if there are three users (A, B and C) who have rated seminars according to Table 4.1, the Matrix Factorisation via Singular Value Decomposition process is calculated as shown in Figure 4.7.

$$
\underset{X}{\begin{pmatrix} 5.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 4.0 & 5.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 2.0 & 4.0 \end{pmatrix}}
$$

$$
= \underset{U}{\begin{pmatrix} 5.1e^{-17} & -1.1e^{-16} & -2.8e^{-1} & -7.8e^{-1} & 5.6e^{-1} \\ -5.1e^{-17} & 4.1e^{-17} & 5.7e^{-1} & 3.3e^{-1} & 7.5e^{-1} \\ -5.1e^{-17} & 7.4e^{-17} & 7.7e^{-1} & -5.3e^{-1} & -3.5e^{-1} \end{pmatrix}}
$$

$$
\underset{S}{\begin{pmatrix} 6.4 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 3.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 3.0 & 0.0 & 0.0 \end{pmatrix}}
\underset{V^T}{\begin{pmatrix} 0.3 & 0.3 & 0.1 & 0.8 & 0.1 \\ -0.2 & -0.2 & -0.8 & 0.3 & -0.5 \\ -0.2 & 0.5 & -0.5 & 0.0 & 0.6 \end{pmatrix}}
$$

Figure 4.7: *Matrix Factorisation via Singular Value Decomposition using the values shown in Table 4.1. All values have been rounded to one decimal place.*

To calculate the seminar recommendations, predictions are made from the decomposed matrices by matrix multiplying U, S and V^T. This calculation returns a matrix of scores for seminars that users have not previously rated. These predictions are shown in Table 4.2. As shown, user A's most relevant suggestion would be seminar three as it contains the highest value out of the seminars they have not attended before. This results in seminar three being recommended to user A.

Table 4.2: *Seminar rating predictions following the Singular Value Decomposition process.*

Seminar User	1	2	3	4	5
A	$5.0e^{0}$	$-10.0e^{-16}$	$-1.1e^{-16}$	$-7.8e^{-16}$	$-1.2e^{-15}$
B	$4.0e^{0}$	$5.0e^{0}$	$6.7e^{-16}$	$-1.3e^{-15}$	$-2.2e^{-16}$
C	$-6.7e^{-16}$	$3.3e^{-16}$	$-8.9e^{-16}$	$2.0e^{0}$	$4.0e^{0}$

By implementing this algorithm, requirement *U2* was satisfied. It stated that users must be able to view seminar recommendations based on how they have rated past seminars.

4.5.2 WuPalmer Similarity

Seminar Roulette was designed to allow users to enter their personal interests into a text input box. In response, seminars displayed a percentage of how relevant the event is to their interests. To calculate this value and satisfy requirement *U6*, WuPalmer was used. WuPalmer was created by Zhibiao Wu and Martha Palmer in 1994, responsible for calculating the similarity between two words by looking at the depths of the words' synsets and lastly looking at the depth of the Least Common Subsumer (Wu and Palmer 1994). The equation is shown in Figure 4.8.

$$
score = 2 * \frac{depth(LCS)}{depth(s1) + depth(s2)}
$$

Figure 4.8: *WuPalmer Similarity where s1 and s2 are the synsets of the comparison words and LCS is the Least Common Subsumer. Equation adapted from Wu and Palmer (1994).*

In the application, $s1$ represented one of the user's interests which they had entered. On the other hand, $s2$ was the seminar's top keyword. An event's keywords were devised by taking its

description, removing the stop words, and calculating each word's frequencies. The word that appeared the most in the description acted as $s2$.

The calculated score was used to determine how similar the words were. For example, if the score was 0.27, then the words were 27% related. A low score like this signified that the seminar was a poor match for that specific user.

4.6 Database Population Script

To satisfy requirements *D1* and *D2*, a script was developed to pull in data from some of the sources explored in Chapter 2 - Samoa Events and Eventbrite. This code was responsible for ensuring that Seminar Roulette always stored and displayed the most up-to-date seminar information on the platform. This was achieved by running the script nightly using a time-based job scheduler. The script made use of the Samoa Events API and Eventbrite API. The following GET requests were used to retrieve data from these sources:

- `https://samoa.dcs.gla.ac.uk/events/rest/Event/searchtext?search=` returns all upcoming research seminars stored in the Samoa Events database.
- `https://www.eventbriteapi.com/v3/organizers/:id/events/` returns all of the events that an organiser has hosted on Eventbrite. The *:id* field was provided using a configuration file that allowed site administrators to configure which Eventbrite organisers they wanted to track. For example, the organiser ID *6830828415* relates to the School of Computing Science's Eventbrite page.

After executing these API calls, both requests returned an array of objects in the JSON format. The objects were then looped over and populated in the Seminar Roulette database. Some of the data provided in the JSON response had to be manipulated so that it conformed to the models outlined in Section 4.3. An extract of the JSON returned by the Samoa Events API is shown in Appendix C. At this stage, no personally identifying information was being stored. So this data collection script did not have to conform to the University's GDPR checklist mentioned in requirement *D4*.

It was imperative to the design of Seminar Roulette that more data sources could be added to the script with ease. This was ensured by documenting the code to allow another developer to add a new data feed in the same way Samoa Events and Eventbrite were implemented. Examples of other data sources that could be added in the future are outlined in Section 7.2.

4.7 Summary

This chapter provided an overview of the project's design phase, enabling another developer to create Seminar Roulette in the same manner but using different technologies than outlined in Chapter 5. Various aspects of the design process were investigated, ranging from the system architecture to the database population script. Throughout this chapter, requirements were referenced to show how the design satisfied them and met their criteria.

5 Implementation

This chapter will detail the implementation of Seminar Roulette, reviewing the various stages of development that led to the creation of a working prototype. In Section 5.1, the software engineering methodologies and practices applied to the project will be explored. Next, Sections 5.2 and 5.3 will examine the tools, technologies and algorithms selected to build the web application and finally, the deployment process will be discussed in Section 5.4.

5.1 Software Engineering Practices

For the duration of this work, software was developed in an Agile manner to aid the project's organisation and management. Following Agile methodologies enabled the product to be split into small incremental builds and working executable prototypes to be released often. A supervisor meeting was organised each week and acted as a stand-up where tasks to be completed for next time were agreed. Upon fulfilment of these goals, the next incremental build of the software product was released. These processes enabled swift delivery of the final prototype and ensured that there was enough time for an adequate evaluation to be conducted.

5.1.1 Version Control

Version control and source code management were integral factors in the project's success. GitHub was chosen to host the codebase online, allowing for there always to be an online backup of the repository if there were any issues locally (GitHub 2021a). Feature branching was utilised within the project to allow for various components to be developed simultaneously. A development branch was created to aid the project's workflow - as features were completed, they were merged into development and tested. Once all of the code in development was fully functioning, it was merged into master and deployed to production. This ensured that if there were ever a problem with the code on one of the feature branches, there would always be a working copy present on the master branch.

5.1.2 Continuous Integration

Early on in the project, continuous integration was introduced in the form of GitHub Actions, allowing workflows to be triggered automatically when new changes were pushed to the repository (GitHub 2021b). A workflow was created and configured to execute when features were merged into the development and master branches. This pipeline was set up to run the suite of unit tests, preventing broken code from being pushed and deployed to production. It was decided that a linting workflow would not be implemented as code was formatted as it was written using YAPF (Google 2021b) and Prettier (Prettier 2021). The continuous integration pipeline was essential in the project's implementation as it successfully identified, on multiple occasions, incorrect code that could have caused significant problems if released.

5.1.3 Issue Management

In the first week of the project, issue management and time tracking were identified as important aspects to ensure deadlines were met and self-organisation was of the highest standard at all times. A Trello board (Trello 2021) was created (see Figure 5.1) to aid with project and task management. It was divided into five columns, each indicating a different stage of progress - *todo*, *blocked*, *doing*, *review* and *done*. This gave a high-level overview of the project in its current state quickly and easily. The board was integrated with the GitHub repository to allow an issue's corresponding feature branch to be displayed on the relevant Trello card. Furthermore, each task was labelled so that the system's component that the issue related to could be viewed at a glance. The labels used were *documentation*, *bug*, *backend*, *frontend* and *dissertation*.

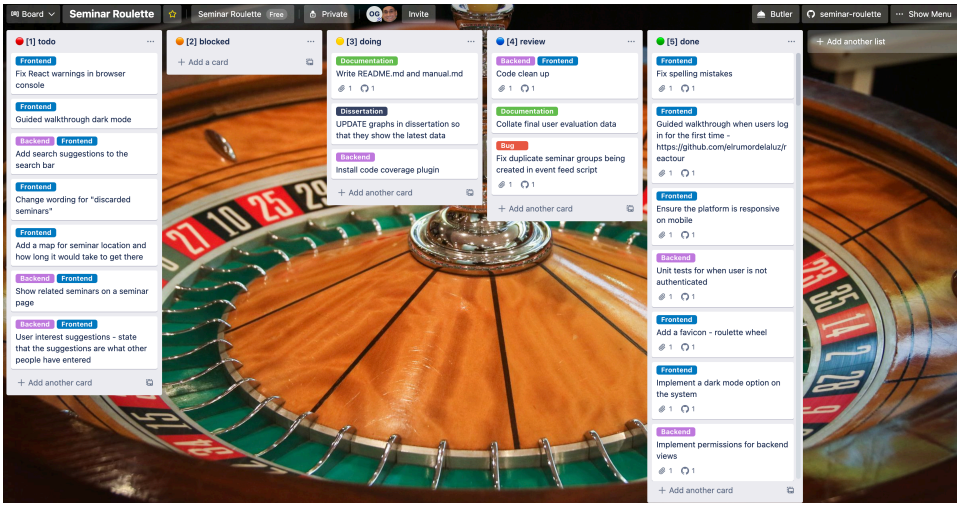


Figure 5.1: *Seminar Roulette's Trello board, used to split up the project's tasks into various lists corresponding to their current progress.*

5.1.4 Kanban Development

As outlined in Section 5.1.3, a Kanban board was created using Trello. Due to the development team only consisting of one person, a Kanban approach was taken towards the development methodology. One of the advantages of using Kanban was that the project could quickly adapt to changing priorities as tasks could be moved effortlessly between the different progress lists when required. This allowed for the list of functional and non-functional requirements devised in Sections 3.2.1 and 3.2.2 to evolve naturally and be re-prioritised if required. The release methodology of this technique was to deploy new features whenever they were ready, without the need for fixed timescales. Due to there being no prescribed release times or dates, documentation was created weekly, identifying the issues that had been completed since the previous week.

5.2 Tools and Technologies

Seminar Roulette was built using many tools and technologies. This section will explore these in detail, explain why they were chosen and reflect on whether they were fit for purpose. The chosen technologies build on top of the design work in Chapter 4, showing how Seminar Roulette was implemented using one set of technologies.

5.2.1 Frontend

From the outset, it was established that the user interface must be efficient, user friendly and visually pleasing. To satisfy these requirements, React, a JavaScript framework for building user interface components was used (React 2021). React was chosen as it is open-source and has a large community backing it. It was developed by Facebook and is highly efficient with updating its components when data is changed, ensuring that the latest information was always rendered on Seminar Roulette. React allowed for reusable components to be created - essential in developing the system to the standard illustrated by the high-fidelity prototypes in Section 4.2.2. Components were reused multiple times throughout the platform, for example, the seminar card, loading spinners, filters and pagination.

Another advantage of using React was how it handles state management. It allows for variables to be stored as states, resulting in components having the ability to be updated without reloading the entire page. For example, in Seminar Roulette, if a user chooses to filter or sort seminars, the React components are dynamically updated without a page refresh. This was key as the platform displayed dynamic content that changed frequently. A diagram illustrating React's component lifecycle is shown in Figure 5.2.

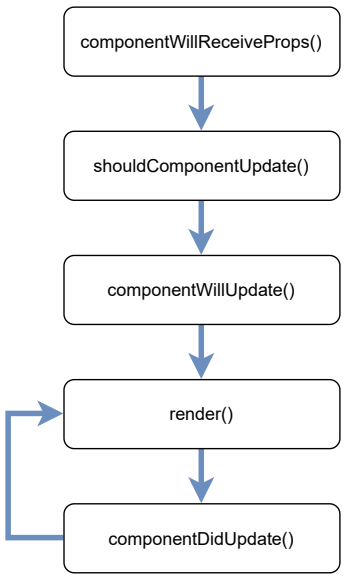


Figure 5.2: *Lifecycle of React's components, adapted from Kef (2019).*

To create a consistent user interface and experience, Material-UI, an open-source React UI framework was used to style the site (Material-UI 2021). Material-UI was built using Material Design principles (Material Design 2021) - a set of guidelines created by Google to help developers build consistent and high-quality applications. Using pre-made frontend components sped up development time as custom components did not have to be designed and implemented. Furthermore, Material-UI contained in-built functionality for handling varying screen sizes, imperative to ensure that the system was compatible on desktop, tablet and mobile. The framework handled this by using responsive user interface elements that adapted to any screen size.

Overall, there were no significant issues with the chosen frontend technologies, and all were fit for purpose. However, in future iterations of Seminar Roulette, a different frontend framework may have to be considered to ensure that the web application can be developed with ease. React has a high development pace, and its environment is constantly being updated. Whilst this could be perceived as an advantage, it impacts the learning curve of React negatively. Furthermore,

documentation is sparse and not kept up to date with the constant updates, which could cause problems if Seminar Roulette's development team was to be expanded to multiple developers.

5.2.2 Backend

Django was used to develop the backend of the web application. Django is an open-source Python-based web framework that is free to use (Django 2021). It was chosen due to it being fast, secure, scalable and versatile. It can scale to large amounts of traffic, which was imperative in satisfying requirement *S8*, enabling the system to be used by tens of thousands of people. Furthermore, Django is highly secure and prevents developers from being susceptible to SQL injections, cross-site request forgery and cross-site scripting attacks.

Django's architecture follows a model-view-controller styled pattern. In this project, this architecture was inappropriate due to the system being designed using a three-tier architecture as outlined in Section 4.1. This required the frontend and backend to communicate with each other using RESTful API calls. To address this problem, the Django REST Framework (DRF) was used (Django REST Framework 2021), enabling a web API to be created with Python and Django.

The chosen backend technologies were fit for purpose, resulting in a smooth implementation stage and a final prototype to be built successfully. However, if the application were to be built again, Flask would be used. Flask is another Python web framework and is referred to as a microframework as it does not require any external libraries (Flask 2021). It is more lightweight than monolithic Django, and web API's are supported straight out of the box, whereas Django requires the Django REST Framework to be installed. In addition, Flask is used by reputable companies such as Netflix and Reddit (StackShare 2021), making it a solid choice for the backend framework.

5.2.3 Database

As explained in Section 4.3, a relational database was more appropriate than a NoSQL database to store site and user information. PostgreSQL was chosen as Seminar Roulette's database management system and is also free and open-source (PostgreSQL 2021). PostgreSQL was favoured over MySQL due to it being better for production-ready systems, crucial if Seminar Roulette was to be deployed as a web service to the University's seminar community. In addition, it has faster read/write speeds which were crucial to the success of system requirement *S10*.

As data was collected, it was stored in a GDPR compliant way, ensuring that a minimal amount of personally identifying information was kept and that database requirement *D4* was met. The University's checklist for databases storing personal data (see requirement *D4*) was strictly adhered to. If Seminar Roulette were to be deployed for public use, a privacy policy would be created so that users are aware of what personal data is being kept on them. Furthermore, there would be an option for academics to have their information removed from the database if they wish. Finally, data would not be stored longer than necessary.

PostgreSQL did not cause any issues throughout the project and was suitable for storing relational data. In future iterations of this work, a hybrid structure could be introduced, consisting of both a relational and NoSQL database in a single instance. This would benefit the application as the system would become highly flexible and perform better when retrieving and storing data.

5.2.4 User Authentication

To meet requirement *A1*, the University's single sign-on was integrated to allow users to authenticate themselves using their GUID and password (see Figure 5.3). The authentication used Shibboleth, a single sign-on architecture that allows people to login to various systems using

one set of credentials (Shibboleth 2021). Shibboleth was only present on the deployed version of Seminar Roulette as it needs to run on a Linux virtual machine to function. Further details regarding the deployment process are discussed in Section 5.4. Within the local environment, a fake user was created so that the system could be robustly tested.

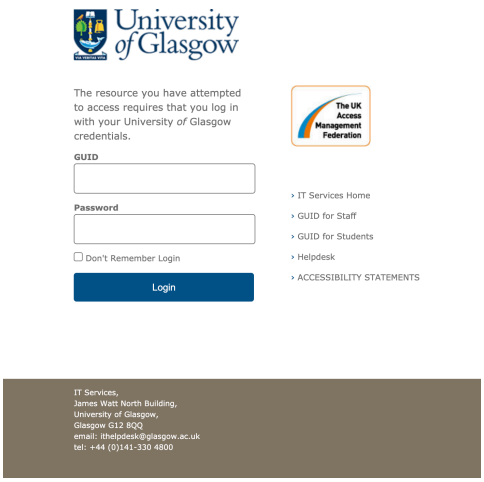


Figure 5.3: *University's single sign-on, enabling academics to sign in using one identity.*

Requirement *A2* states that the platform should be able to be used by users who are not authenticated. This was developed on Seminar Roulette to allow academics to browse and search for events happening at the University casually. If users decided to use the system anonymously, their feature set was limited, and access to recommendation features was restricted. Furthermore, they were unable to rate past seminars. This was due to these features all requiring the user's personal information to be stored; for example, if a user rated a seminar five stars, then the database had to store who rated that event.

5.3 Recommender Algorithms

This section outlines the implementation of the algorithms described in Section 4.5, showing how they were written with the chosen programming language, Python. External libraries used will also be outlined and shown how they were integrated to ensure that the algorithms produced the correct results.

5.3.1 Seminar Recommendations

Seminar recommendations were provided to users through the Matrix Factorisation via Singular Value Decomposition (SVD) algorithm discussed in Section 4.5.1. To achieve this, a Python data analysis and manipulation library called Pandas was used (Pandas 2021). This enabled seminar data and user ratings to be turned into a DataFrame, a two-dimensional and size-mutable data structure exclusive to Pandas. DataFrames can be manipulated into matrices to allow the SVD to take place. To handle the SVD, the pre-written *svds* function from SciPy was used. SciPy is a Python package used to solve scientific and mathematical computing problems (SciPy 2021).

5.3.2 Similarity Matching

This section provides a visualisation of the WuPalmer similarity algorithm explored in Section 4.5.2. Figure 5.4 displays a seminar titled *The Road to a Sufficiently Smart Compiler*. The system

successfully identified the keywords for this talk and displayed them in the bottom right-hand corner - *compilers*, *work* and *specialisation*. The WuPalmer algorithm has calculated that between these keywords and the user's personal interests, there is an 82% match. In this example, the user's interests are *maths*, *code* and *machine learning*. As expected, there is a reasonably high similarity between these sets of words as *code* and *machine learning* are highly related to *compilers*.

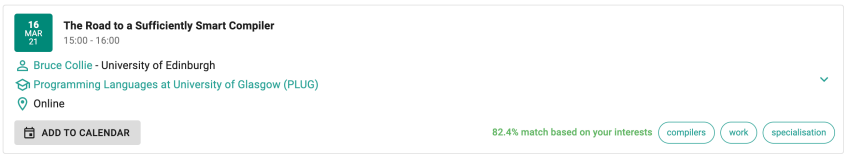


Figure 5.4: *A seminar card displaying its keywords and the similarity match based on a user's interests.*

The WuPalmer algorithm was implemented using the Python Natural Language Toolkit or NLTK for short (NLTK 2020). NLTK is a Python package that contains a suite of libraries for semantic natural language processing. Within NLTK's wordnet library, the pre-written function *wup_similarity* was used to compute a similarity score between two words.

5.3.3 Performance Regressions

Throughout the implementation stage of the project, a struggle was ensuring that the recommender algorithm performed efficiently and returned data within an acceptable timeframe. Due to the algorithm being extremely accurate and calculating new seminar suggestions each time the user loaded the main page, it was sometimes slow to execute. It was decided that these inefficiencies were sound for a prototype but would not be suitable for a production system. In the prototype version of the system, accuracy was favoured over performance because the project was limited in time and resources. In future iterations of Seminar Roulette, performance regressions would have to be addressed. A solution to this would be to cache some of the recommendation data, resulting in event suggestions being retrieved from memory if the user has not rated any new seminars when the page is loaded. Another workaround would be to use asynchronous calls so that the user can still browse the rest of the application whilst their recommendations are being calculated. Alternatively, seminar suggestions could be devised using precomputation and updated infrequently. For example, a time-based job scheduler could be set up to calculate event recommendations on a nightly basis, resulting in the system becoming more efficient as it would not have to determine recommendations each time a user loads the site.

5.4 Deployment

As a Kanban style of development was undertaken for the project's duration, it was imperative that Seminar Roulette was deployed early on and new updates released frequently. The system was deployed to the CentOS distribution of a Linux virtual machine with an Apache web server. The virtual machine used belonged to the University, so the application sat behind the University's firewall. CentOS was chosen as it was one of the operating systems compatible with the University's single sign-on. This caused some significant difficulties, ensuring that the correct version of the requirements was installed and did not conflict with each other. Furthermore, the firewall slowed development time immensely as workarounds had to be found to bypass the firewall at specific points. For example, an SSL certificate could not be signed by an external party to ensure that user data was encrypted during transit when using the system.

When a feature was completed and tested, it was released immediately as per the Kanban development methodology. This process was tedious as it was not automated and had to be

done manually. This involved logging into the virtual machine, pulling the latest commit from GitHub, installing any new requirements, migrating the database models, collecting the static files and lastly, restarting the web server. In future, a new continuous integration pipeline would be configured to handle automatic deployment whenever code is merged into the master branch.

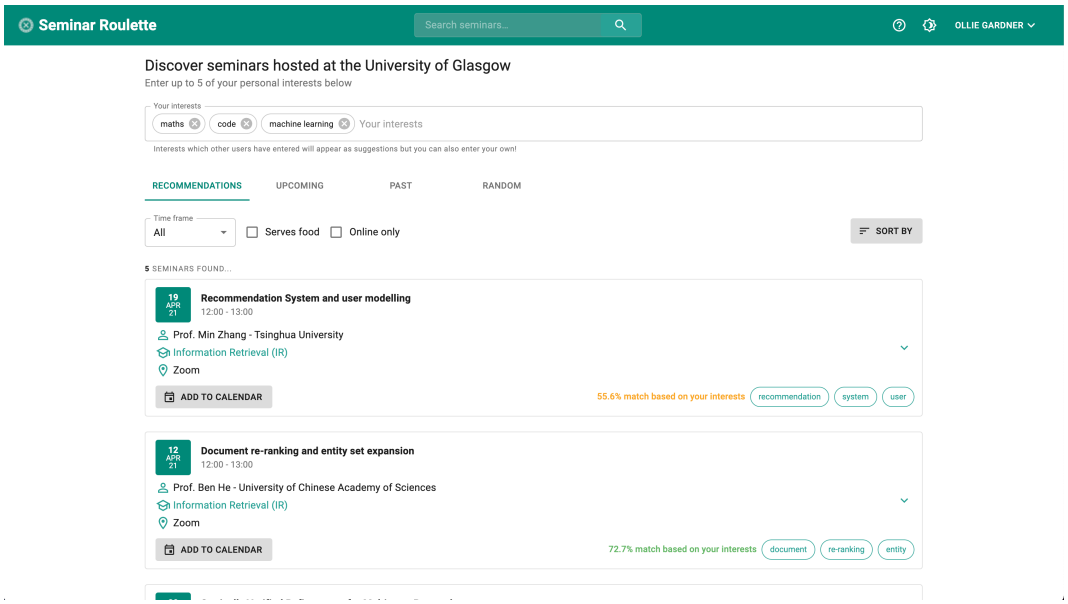
In the weeks leading up to the completion of this work, the University had an unexpected power outage in their server room. Unfortunately, this affected Seminar Roulette's virtual machine and resulted in it becoming corrupted and unrecoverable. Due to the upcoming project deadline, there was no time for Seminar Roulette to be set up on a new virtual machine as the deployment process was lengthy. To ensure fast and efficient deployment in the future, Docker (Docker 2021) containers could be set up to ensure that Seminar Roulette can be successfully deployed on any machine. This would allow for all of the project's software, libraries and configuration files to be bundled into one and deployed from anywhere in the world, removing the need to install packages and requirements whenever Seminar Roulette has to be installed onto a new virtual machine.

5.5 Final Product

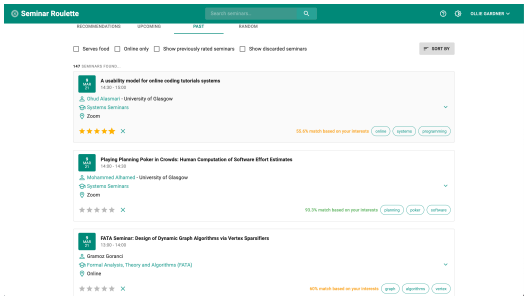
Figure 5.5 on page 30 illustrates the Seminar Roulette system after implementation had been completed. The look of the user interface differs slightly from the high-fidelity prototypes created in Section 4.2.2 due to feedback received in a think-aloud evaluation. This will be explored further in Chapter 6. Pictured are the various components of Seminar Roulette, allowing seminar attendees to find events with ease that they have not thought about attending before.

5.6 Summary

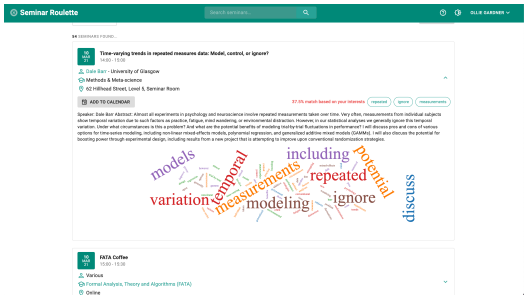
This chapter described the implementation stage of the project, highlighting the various steps that were undertaken to develop the prototype application. The project's organisation was detailed through the use of various Agile software engineering practices. The technologies and libraries used to implement the system were discussed in detail and evaluated based on their suitability. Finally, a high-level overview of the deployment process was provided, enabling another developer to release Seminar Roulette if desired.



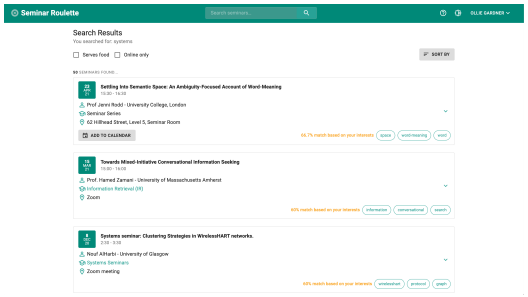
(a) Seminar Roulette's main page, showing a user's seminar recommendations.



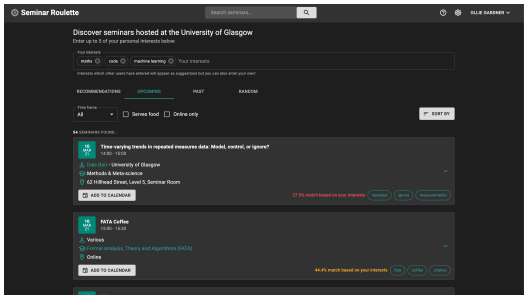
(b) Past seminars, showing the ability to give an event a star rating.



(c) Expanded seminar, displaying more information about the event.



(d) Seminars returned following a search for "systems".



(e) Upcoming seminars in dark mode.

Figure 5.5: *Screen captures of the final product after implementation was completed. Figures (a) to (e) show the different components of the web application. The layout of the page stays consistent throughout the various views. The user has the option of filtering down the returned seminars using filters. One can filter by time frame, whether the seminar serves refreshments or if the seminar is online or in-person. The list of seminars can also be sorted alphabetically or by start date and time.*

6 Evaluation

This chapter provides an evaluation of the Seminar Roulette web application, examining the various complementary approaches undertaken to determine the project's success. These approaches range from assessing the performance of the system and codebase in Sections 6.1 and 6.2 to evaluations conducted with human participants towards the end of the project in Sections 6.3 and 6.4. To conclude, the functional and non-functional requirements devised in Sections 3.2.1 and 3.2.2 will be analysed to determine whether the final prototype satisfied each.

Evaluations that involved human participants adhered to the ethics checklist created by the University's School of Computing Science. Before these took place, the ethics form was signed by both the evaluator and project supervisor, as shown in Appendix A.

6.1 Unit Testing

As discussed in Section 5.1.2, a continuous integration pipeline was configured at the start of the project. It was responsible for running a suite of 42 unit tests whenever code was merged into either the development or master branch. This test suite consisted of 34 backend and eight frontend tests. These tests were created to validate that each of Seminar Roulette's components was performing as intended. The backend unit tests ranged from testing the connection to the Samoa Events API to determining if a seminar filter worked correctly. The Python Coverage library (Coverage 2021) was used to produce a code coverage report, as shown in Table 6.1. The report indicates that an overall coverage of 87% was achieved by the backend unit tests. Research conducted by George and Williams (2004) states that a code coverage of between 80% and 90% is accepted as the industry standard. Therefore, Seminar Roulette's backend test suite was effective in ensuring that the application performed as expected.

Table 6.1: *Code coverage report generated from Python's Coverage library, indicating a total coverage of 87%.*

Module	Statements	Missing	Excluded	Coverage
backend/__init__.py	1	0	0	100%
backend/admin.py	29	0	0	100%
backend/apps.py	13	0	0	100%
backend/managers.py	28	5	0	82%
backend/models.py	87	3	0	97%
backend/serializers.py	32	0	0	100%
backend/tests.py	156	0	0	100%
backend/urls.py	7	0	0	100%
backend/views/views_misc.py	31	1	0	97%
backend/views/views_seminar.py	142	32	0	77%
backend/views/views_user.py	107	40	0	63%
Total	**633**	**81**	**0**	**87%**

Along with the backend unit tests, a small suite of eight frontend tests was developed to verify that Seminar Roulette's React components were being rendered correctly to the user. Only a small number of frontend tests were created due to the time constraints of the project. In addition, priority was given to the backend unit tests as these were responsible for testing the underlying logic of the platform and ensuring no broken code was ever pushed to production. The frontend test suite used smoke tests to ensure that the production version of Seminar Roulette was stable enough to be released. An example of an integrated smoke test is shown in Listing 6.1 to ensure that the application rendered without throwing any errors when the app component was mounted.

```
import React from "react";
import { render, unmountComponentAtNode } from "react-dom";
import App from "../App";

it("Renders application without crashing", () => {
  const root = document.createElement("div");
  render(<App />, root);
  unmountComponentAtNode(root);
});
```

Listing 6.1: *Frontend smoke test verifying that the app component rendered without crashing or throwing any errors.*

Along with the eight frontend tests, the frontend was also tested manually through the final user evaluation described in Section 6.4. Although this hybrid of automated and manual frontend testing was sufficient for this project, it was not the most optimal method of testing the user interface. This would have to be addressed in future iterations of the system to ensure that the entirety of the application is tested robustly and automatically.

6.2 System Performance

In Section 3.2.2, system requirement *S10* states that the application must be efficient and respond within two seconds. To evaluate this metric, the quality of the application was measured through Lighthouse (Google 2021a), a tool developed by Google that evaluates the performance of websites. Lighthouse is commonly used on commercial websites and measures various metrics, of which five were tracked for this evaluation. These were *performance*, *accessibility*, *best practices*, *time to interactive* and *speed index*. *Time to interactive* was the number of seconds taken for a page to load completely, whilst *speed index* measured the number of seconds it took for a page's contents to be populated visibly. The aforementioned metrics were tracked on each of Seminar Roulette's four pages:

- **Anonymous:** the homepage with a limited feature set available due to the user not being logged in.
- **Single sign-on:** the University's single sign-on page integrated with Shibboleth.
- **Homepage:** the main page of the site when the user has logged in, resulting in all seminar recommendation features being available to them.
- **Search:** the page that appears after a user performs a free-text search for an event.

The results from the system performance evaluation are shown in Table 6.2. To retrieve these results, the system performance evaluation was carried out three times on the same device and network, with the average score of each metric then being calculated.

Table 6.2: *System performance results from Lighthouse's evaluation. The performance, accessibility and best practices columns have a maximum score of 100. Red indicates a poor score. Orange represents that the metric needs improved, and green demonstrates a good score.*

Page	Performance	Accessibility	Best Practices	Time to Interactive (s)	Speed Index (s)
Anonymous	90	82	93	1.5	1.3
Single sign-on	100	68	100	0.4	0.4
Homepage	59	84	93	2.7	3.0
Search	60	84	93	3.1	1.3

Overall, Seminar Roulette functioned well in the performance evaluation. However, there were a few areas where the system scored poorly and will need to be improved. A common trend in the results was that the application performed worse on pages that contained lots of dynamic content. Similarly, pages with fewer features were much more efficient and responded quicker. To determine the success of requirement *S10*, the *time to interactive* metric was analysed as this represented when the page was able to be fully interactive to the user. Based on this, two out of the four pages failed to meet requirement *S10* as they took longer than two seconds to load completely. The pages likely took longer than two seconds to load due to the performance regressions discussed in Section 5.3.3. The anonymous homepage and single sign-on both passed this test. However, those pages were limited in features and did not represent the system as a whole. Therefore, requirement *S10* was not met, potentially resulting in users becoming deterred from the system and never using it again. Moving forward, the performance regressions would need to be addressed, aiming to reduce page load times to less than two seconds.

6.3 Think-Aloud Evaluation

At the end of semester one, a think-aloud evaluation was conducted with six participants: the project supervisor's postgraduate students. All were regular research seminar attendees, and so were a suitable target group to participate. This evaluation aimed to determine the features that they thought were missing from the system at that stage in time. Furthermore, feedback was sought on the user interface and experience to ensure that Seminar Roulette was easy to use without any prior training.

During the evaluation, participants were each interviewed on Zoom for 30 minutes. They were given access to Seminar Roulette and allowed to freely roam the system with no formal tasks to complete. They were asked to speak their thoughts out loud and comment on the usability of the application and whether there were any features that they thought were missing. The feedback received from the think-aloud evaluations is summarised below:

- Participants generally found the grid structure of seminars confusing and felt that there was too much information crammed onto the page. This design is illustrated by the high-fidelity prototypes in Figure 4.2.2. Users stated that they would prefer to be greeted with a list of seminars.
- Add the ability to filter out seminars that are titled *TBA*, *TBC* or *TBD*.
- Be able to see which seminars friends are clicking on and the ones that they are rating highly.
- Participants suggested adding a legend or tooltip for the icons to explain what they were representing.
- At first glance, the random seminar floating action button was not prominent and in an unexpected place.

- Implement the ability to add a seminar to one's calendar.
- Some participants reported that they did not understand what the aim of the application was at first and suggested adding some helper text somewhere near the top of the page.
- When a seminar was clicked, participants suggested opening the seminar information as a pop-up would be better than taking the user to a new page.

The feedback gathered was used to determine how the second semester of the project should be spent. Following the completion of the think-aloud evaluations, attention was turned to the frontend and layout of the system. Action was taken to improve the user interface and experience in semester two.

6.4 Final User Evaluation

Throughout this work, feedback from human subjects was imperative to ensure that Seminar Roulette fulfilled all of its requirements. Towards the end of the project, a final user evaluation was carried out to measure the suitability and usability of the system. The evaluation collected both quantitative and qualitative data intending to improve the prototype in future iterations further. The final user evaluation was completed by 34 participants.

6.4.1 Evaluation Procedure

As shown in Appendix D, a questionnaire was created using Google Forms and was split into three sections - tasks to complete, quantitative data and qualitative data.

Tasks to Complete

Participants were given a list of 10 tasks to perform on Seminar Roulette, enabling them to become familiar with the platform and learn how it functioned. The 10 tasks are listed below:

1. Navigate to `https://howard.dcs.gla.ac.uk/` in your web browser.
2. The platform can be used without signing in but has a limited feature set. Please take a look at upcoming and random seminars.
3. Login using your University of Glasgow GUID and password by clicking the LOGIN button in the navigation bar.
4. Enter up to five of your personal interests.
5. Find an upcoming seminar and click on it to discover more information about it.
6. Filter and sort the seminars to your desired needs.
7. Find at least two past seminars and rate them.
8. Navigate to your seminar recommendations.
9. Find a random seminar.
10. Search for a seminar.

Following the completion of these tasks, participants were asked to move onto the quantitative data stage of the form.

Quantitative Data

The quantitative data section of the questionnaire was based on a Post-Study System Usability Questionnaire (PSSUQ), a standard method of measuring user satisfaction with a software product (Will T 2021). The PSSUQ survey consisted of 16 questions with a seven-point Likert scale ranging from *strongly agree* to *strongly disagree* as the options for responses. Participants were also able to leave a question blank to indicate that it was not applicable. The questions in the PSSUQ survey are shown in Figures D.2 to D.6.

To determine the overall result of the PSSUQ survey, the scores from the seven-point Likert scale were averaged. The aim was to have as low a score as possible as this indicated good user performance and satisfaction. The advantage of using PSSUQ was that the results could be split up into three subsets, measuring *system usefulness*, *information quality* and *interface quality*. This enabled the usability of each system component to be assessed independently of the other.

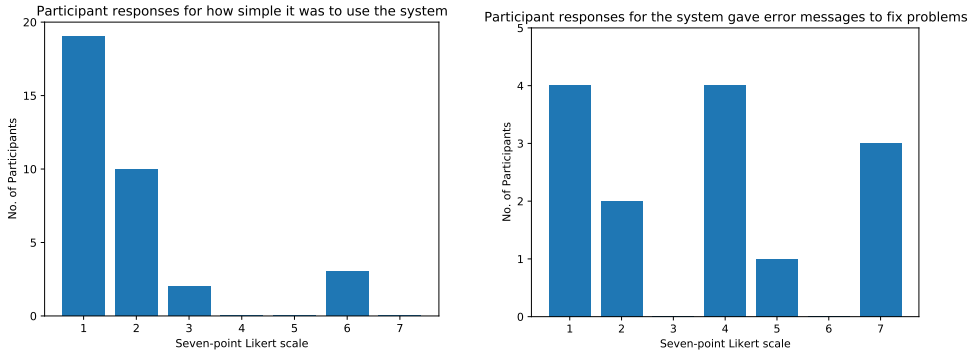
Qualitative Data

After completing the PSSUQ survey, participants were asked to fill in a series of open-ended questions. These were designed to elicit qualitative feedback from participants. The questions asked revolved around discovering how users felt Seminar Roulette could be improved in the future or if there were any additional features that they would like added to the system. Furthermore, the qualitative section allowed participants to report bugs and errors that they encountered whilst testing the platform.

6.4.2 Quantitative Results

The PSSUQ aimed to achieve the lowest score possible, where a score of one indicated that the system was easy to use, and seven implied that the platform performed poorly. Seminar Roulette scored 1.9 in the overall PSSUQ, showing that the system was highly usable. As mentioned in Section 6.4.1, the advantage of using a PSSUQ survey was that the overall score could be split into three sub-scores; system usefulness, information quality and interface quality. The results of these were 1.7, 2.0 and 2.0, respectively.

As all three sub-scores were low, the platform was labelled highly usable in usefulness, information quality and interface quality. Based on these scores, non-functional requirements *U11* and *U12* were satisfied as the system was easy to use without any prior training and provided an intuitive user interface. This is further backed up by Figure 6.1a, where a column chart is shown of user responses to the statement *it was simple to use this system*. 56% of participants stated that they *strongly agree* that the platform was simple to use.



(a) *PSSUQ quantitative results for question two.* ***(b)*** *PSSUQ quantitative results for question seven.*

Figure 6.1: *Column charts showing participant responses on the seven-point Likert scale where 1 represents "strongly agree", and 7 indicates "strongly disagree". (a) shows responses to the statement "it was simple to use this system", whilst (b) shows responses to the statement "the system gave error messages that clearly told me how to fix problems".*

The PSSUQ resulted in some low usability scores. While it can be concluded that Seminar Roulette was a success in being simple to use, it should also be noted that there might be some bias in the data collected in the final user evaluation. Due to the restrictions imposed by COVID-19, many of the participants in the evaluation were friends or people who personally knew the evaluator. This may have caused some results to be skewed in favour of the evaluator, and this

should be taken into account if Seminar Roulette was to be deployed. Post COVID-19, a further evaluation could be conducted with a new group of participants who do not know the evaluator to determine whether the platform would be suitable in the University's portfolio of event software products.

The most varied results from the PSSUQ came from statement seven; *the system gave error messages that clearly told me how to fix problems* (see Figure 6.1b). There were only 14 responses meaning that 20 participants left this statement blank to indicate that it was not applicable. In addition, three participants stated that they *strongly disagree* with the platform displaying error messages to them. This implies that Seminar Roulette could have been better in showing descriptive error messages to users when they encountered problems. By not improving upon this, users could become confused and these issues could deter them from using the application ever again. As only four participants disagreed with this statement, it was deemed that non-functional requirement *S9* was still met as 10 participants gave positive feedback.

6.4.3 Qualitative Results

After finishing the PSSUQ survey, participants were asked to complete five open-ended questions to elicit qualitative feedback. These questions were designed to further assess the suitability of Seminar Roulette and shaped the future work outlined in Section 7.2 of Chapter 7. The responses to these five questions have been summarised below.

Whilst using Seminar Roulette, did you encounter any errors? If so, please describe the error and how it arose.

59% of respondents reported that they did not encounter any errors whilst using the platform, showing that the system is bug free and can reliably provide seminar recommendations to University academics. Bugs that were reported were not detrimental to the functionality of the system. These included minor spelling mistakes and lots of users experiencing slow loading times, further backing up the performance issues discussed in Section 5.3.3.

Do you think that Seminar Roulette could be a useful platform to provide seminar recommendations to university academics? If no, please specify why not.

All 34 participants said that Seminar Roulette would be a suitable platform to provide seminar recommendations to academics, showing that a desire exists for a system of this nature within the University's software portfolio.

Do you have any suggestions on how the Seminar Roulette platform could be improved?

Many different suggestions were given for how the platform could be improved in the future. The most noteworthy of these were:

- Fix the performance problems so that users do not have to wait a long time to receive their recommendations.
- Make the four tabs, *recommendations*, *upcoming*, *past* and *random* clearer as there was occasionally some confusion as to what these were representing.
- Some participants did not understand the difference between seminar recommendations and the personal interests percentage matching. Users suggested adding some helper text on the homepage to describe the differences between the two features.

Are there any additional features that you would like to see in Seminar Roulette?

There was some overlap between features suggested by participants and features outlined in the functional requirements that were not implemented due to time constraints. The most common future feature suggestions were:

- Implement a discussion area where attendees could exchange messages pre and post-event.
- Add graphics for the seminar speaker to make it easier to network with people at the event.

- Integration with Microsoft Exchange calendar and only recommend seminars that fit around an academic's schedule.

In five words, how would you describe the overall design and functionality of Seminar Roulette?

The final question of the user evaluation form asked participants to describe the overall design and functionality of Seminar Roulette in five words. Using their responses, a word cloud was created (see Figure 6.2) to show a weighted list of the most common feedback received. The most common words received were *easy*, *clean*, *simple* and *intuitive*, solidifying the aims set out in Section 1.2. These words were used by respondents 16, 13, 9 and 8 times, respectively.

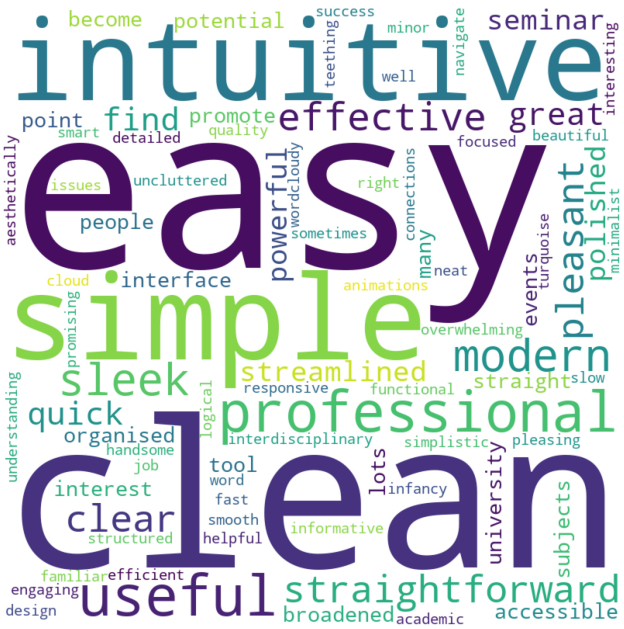


Figure 6.2: *Word cloud displaying words that participants used to describe their experience with Seminar Roulette. The size of each word correlates with the number of times that word was used by participants.*

6.5 Requirement Validation

To complete the evaluation of Seminar Roulette, the functional and non-functional requirements devised in Sections 3.2.1 and 3.2.2 were analysed to determine whether they had been satisfied or not. Within these sections, three requirements were assigned the priority *Won't Have This Time* and were out of the project's scope from the outset. As a result, these requirements will be excluded from this section of the evaluation.

In total, 27 functional and non-functional requirements were considered. Of these, 21 were met or 78%, as shown in Table 6.3. All *Must Have* requirements were satisfied apart from *S10* for reasons outlined in Section 6.2. All five *Should Have* requirements were successfully implemented. None of the five *Could Have* requirements were met due to the time constraints of the project. Not fulfilling these requirements was not detrimental to the final evaluation outcome as they were only going to be implemented if time allowed.

Whilst the final prototype did not deliver on all of the requirements that it set out to achieve, the project was still regarded as a success due to 78% of all requirements, and 94% of *Must*

Have requirements being satisfied successfully. Future iterations of Seminar Roulette will aim to improve upon this figure, with 100% requirement satisfaction being the ultimate goal.

Table 6.3: *List of functional and non-functional requirements, showing whether the final prototype satisfied the requirement or not and its corresponding priority. A ✓ signifies that the requirement was met whilst a × shows that it was not.*

(a) *Functional requirements.*

Requirement	Priority	Satisfied
S1	MH	✓
S2	MH	✓
S3	SH	✓
S4	CH	×
S5	CH	×
S6	WH	×
U1	MH	✓
U2	MH	✓
U3	MH	✓
U4	MH	✓
U5	SH	✓
U6	SH	✓
U7	SH	✓
U8	CH	×
U9	CH	×
U10	CH	×
A1	MH	✓
A2	SH	✓
A3	WH	×
D1	MH	✓
D2	MH	✓
D3	WH	×

(b) *Non-functional requirements.*

Requirement	Priority	Satisfied
S7	MH	✓
S8	MH	✓
S9	MH	✓
S10	MH	×
S11	MH	✓
U11	MH	✓
U12	MH	✓
D4	MH	✓

6.6 Summary

This chapter described various evaluation methods in order to determine the success of the final prototype. The range of evaluative techniques measured the robustness of the codebase, time taken for the system to become interactive, multiple user evaluations with human participants and finally validation of the functional and non-functional requirements set out in Sections 3.2.1 and 3.2.2.

7 Conclusion

This chapter begins with a reflection in Section 7.1 of what went well and what did not within the project, investigating future work that could be undertaken if the project was to be continued for another six months in Section 7.2. Finally, a summary of the entirety of this work is provided in Section 7.3.

7.1 Reflection

This project has allowed the developer to gain a greater insight into the inner workings of University systems and learn how to build clean, responsive and functional user interfaces for deployed software. Another key takeaway from this work was the self-organisation required to develop a large-scale software product single-handedly. Prior to this project, work of this size had only been undertaken by the author in a group setting with multiple developers on board. The iterative nature of this project was new to the developer and allowed for the idea of Seminar Roulette to come to fruition. This took Seminar Roulette from paper prototypes to a robust Django and React web application. The evaluation process of the project was mainly new to the developer, and it allowed them to realise how large-scale software products might be tested and evaluated within industry.

Seminar Roulette acted as the author's first time developing a recommender system within a software product. The developer had to conduct extensive research into the most suitable algorithm for the task and learn how to implement it using the chosen technologies. This was a huge learning curve and took a significant amount of time during the design and implementation stages.

If Seminar Roulette were to be built from scratch again, further initial research would be conducted into how to ensure that the system is accessible to users with impairments, as this is something that was overlooked within the design stage of the project. Furthermore, more attention would be given to the colour scheme and design, ensuring that the platform looks and feels like a University software product.

7.2 Future work

A multitude of work could be undertaken to extend the functionality of Seminar Roulette in the future. However, before additional features could be implemented, the short-term goal would be to address the current performance issues to ensure that the platform is efficient as possible. In turn, this would improve usability and reduce the likeliness of the platform becoming cumbersome to use.

For future implementation, the functional and non-functional requirements identified in Sections 3.2.1 and 3.2.2 that were not achieved would be prioritised. As none of the *Could Have* requirements were implemented, these would be acted upon first. The main focus would involve extending the functionality of the recommendation feature, allowing academics to connect their Microsoft Exchange calendar to Seminar Roulette to enable seminar suggestions to be made that

fit around their timetable. This would allow users to have an even more personalised experience with the system.

Although the prototype in its current state scrapes data from Samoa Events and Eventbrite, the University of Glasgow use other services to advertise seminars taking place on campus. These feeds were not included in the database population script discussed in Section 4.6 due to the time constraints of the project. Other data feeds that could be integrated into Seminar Roulette are Bookitbee (Bookitbee 2021), Yammer (Yammer 2021) and Meetup (Meetup 2021). Bookitbee is an online booking system for events. Yammer is a social networking platform used within large organisations, and Meetup is an online service to organise events. Whilst Bookitbee and Yammer both have their own pages exclusive to the University, Meetup is an external platform that would allow Seminar Roulette to extend its functionality to show events happening in Glasgow but outwith the University.

In future iterations of Seminar Roulette, the platform could be re-targeted for other academic institutions, allowing other Universities to provide their academics with event recommendations. The current application is closely coupled to University of Glasgow business systems. A new version of the platform would have to re-target the single sign-on and database population script to the relevant data feeds. This process is better known as *white labelling*, where a product is produced that has the capability to be re-branded to suit any requirements. For example, if the University of Strathclyde wanted to add Seminar Roulette to their software portfolio, the database population script would need to include their event data feeds. Strathclyde advertises seminars through their website and their own custom seminar system, sbs.strath (University of Strathclyde 2021).

7.3 Summary

In summary, Seminar Roulette is a web-based platform designed to provide event recommendations to University of Glasgow academics. The system has successfully managed to pull in data from both Samoa Events and Eventbrite, enabling it to become a centralised hub for seminar attendees. Users can rate past seminars and enter keywords that relate to their personal interests. In response, users are provided with a list of event recommendations and a percentage value for how similar a seminar is to their interests. The system made use of the University's single sign-on, allowing academics to login to the application using their University GUID and password.

Seminar Roulette was evaluated using various techniques, including unit testing, system performance, two end-user evaluations and requirement validation. Using the aforementioned methods, it was concluded that Seminar Roulette effectively provides valuable seminar suggestions to members of the University's academic community. Although the system had some significant performance issues, the platform was still deemed highly usable and able to be used without any prior training. By achieving both of these goals, the original aims set out in Section 1.2 were validated.

With some minor adjustments, Seminar Roulette, in its current state, could be deployed as a web service within the University of Glasgow, allowing academics to discover seminars and events that they have not thought about attending before. By successfully implementing a working prototype, the platform is the first University system to adopt a unified approach to seminar management. This solidifies the speculation outlined in Section 1.2, where it was stated that *there is potential for the new system to become a staple piece of software in the University's seminar community*. Therefore, the entirety of this work can be regarded as a success.

A Ethics Checklist

This appendix contains a copy of the ethics checklist, required to be signed before end-user evaluations with participants could take place.

Department of Computing Science
University of Glasgow

Ethics checklist form for 3rd/4th/5th year, MSc IT/CS/ACS 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 submit an ethics approval form to the Department Ethics Committee.

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, mobile phones, and PDAs 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 A.1: *Signed ethics checklist - page 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.

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 Seminar Roulette

Student's Name Ollie Gardner

Student's Registration Number 2310049g

Student's Signature Ollie Gardner

Supervisor's Signature D.O. Sugden

Date 14/12/20

Figure A.2: *Signed ethics checklist - page 2 of 2.*

B Initial User Survey

This appendix provides the initial user survey, which was distributed amongst key stakeholders to build a list of requirements.

Seminar Roulette Requirements Survey

Every week, tens (if not hundreds) of research seminars take place at Glasgow University. I try to get along to a few of these, but I don't always have time to look for the most interesting events to attend. Sometimes I don't even know whether a seminar will be fun or not, so I just give it a miss. Another complication is that my diary schedule changes every week so I can't always make regular seminar times. This project involves ingesting data from seminar RSS feeds and other sources to determine what's happening at the University, then suggesting seminars that I might like to attend - based on my diary availability combined with some element of randomness.

The key problems are:
1) determining what I like and making appropriate recommendations - this might involve some machine learning.
2) adding an element of serendipity so I attend new events I might not have thought about attending before
3) scaling this system up so it works for all academics at the university

If the system works well, it could be deployed as a web service for the wider university community.

* Required

1. On a typical week, how many seminars do you attend at the university? *

Mark only one oval.

0	1	2	3	4	5	6	7	8	9	10
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

2. What is your main area of research? *

_____

Figure B.1: *Initial user survey - page 1 of 4.*

3. Currently, we have 2 potential data sources for Seminar Roulette - Samoa (https://samoa.dcs.gla.ac.uk/events/) and EventBrite. Are there any other data sources which you are aware of that the University use for advertising seminars? If so, what are they? *

4. We would like to connect your calendar so that the system can suggest seminars which fit around your schedule. Which calendar service(s) do you use for work? *

Check all that apply.

- [ ] Google Calendar
- [ ] Microsoft Outlook / Exchange
- [ ] Apple Calendar / iCloud
- [ ] Other: _____

5. I think that it is important for the system to take into account my preferences/interests so that it can suggest seminars which I like. *

Mark only one oval.

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly agree

Figure B.2: *Initial user survey - page 2 of 4.*

6. I would like there to be a feature which randomly suggests seminars to me. *

   Mark only one oval.

   - ◯ Strongly disagree
   - ◯ Disagree
   - ◯ Neutral
   - ◯ Agree
   - ◯ Strongly agree

7. Once relevant seminar(s) have been suggested to you, would you like to be able to add them to your respective calendar? *

   Mark only one oval.

   - ◯ Yes
   - ◯ No
   - ◯ Maybe

8. Once relevant seminar(s) have been suggested to you, would you like to be notified before they start by email or otherwise? *

   Mark only one oval.

   - ◯ Yes
   - ◯ No
   - ◯ Maybe

Figure B.3: *Initial user survey - page 3 of 4.*

9. We endeavour to integrate the University's single sign-on system (Shibboleth) into the web application. Would you like to be able to use the system anonymously too? *

   Mark only one oval.

   - ◯ Yes
   - ◯ No
   - ◯ Maybe

10. Are you interested in attending seminars outwith your school/research area? For instance, would you be interested in attending Engineering seminars if you are a Computing Science academic? *

    Mark only one oval.

    - ◯ Yes
    - ◯ No
    - ◯ Maybe

11. Are there any additional features which you would like to see implemented into Seminar Roulette?

Figure B.4: *Initial user survey - page 4 of 4.*

C Samoa Events API Response

This appendix contains an extract of the JSON response returned when the Samoa Events API was called.


```
{
  "id":17712,
  "title":"A usability model for online coding tutorials systems",
  "startTime":"2021-03-09T14:30",
  "endTime":"2021-03-09T15:00",
  "speaker":"Ohud Alasmari",
  "speakerAffiliation":"University of Glasgow",
  "speakerUrl":"https://www.gla.ac.uk/pgrs/ohudalasmari/#supervisors",
   "description":"<p><span>Online Coding Tutorial systems provide a
       ↪ basis for free and open interactive programming education at
       ↪ scale. Such browser-based systems featuring automated feedback
       ↪ are increasingly popular as remote learning has become
       ↪ normalized. Programming students with different social and
       ↪ cultural backgrounds from all over the world can access these
       ↪ platforms. In addition, Online Coding Tutorials facilitate
       ↪ practical software development experiences that form an
       ↪ integral part of the learning process for novice programmers.
       ↪ However, such systems will only be truly effective if they meet
       ↪ diverse programming learner requirements. In this paper, we
       ↪ argue that these requirements must be informed by a range of
       ↪ disciplines, including system usability, computing pedagogy,
       ↪ and internationalization. We conducted a wide-ranging survey of
       ↪ partially relevant usability models; from these studies we
       ↪ synthesized a new and specialized hierarchical usability model
       ↪ for Online Coding Tutorial systems. This new model has four
       ↪ dimensions: pedagogy, platform, culture and cognition. We claim
       ↪ that, in relation to previous models, our multi-dimensional
       ↪ framework covers a more comprehensive range of requirements for
       ↪ online programming language learners. This framework can be
       ↪ used to characterize and compare existing online tools, as well
       ↪ as to inform good design practice for new tools. We provide
       ↪ initial evidence of the potential utility of our model by
       ↪ applying it to three mainstream programming learning tools:
       ↪ LearnPython, TryRuby, and TryJavaScript.</span></p>",
  "location":{
    "id":200,
    "location":"Zoom",
    "directions":"",
    "mapUrl":null,
    "latitude":0.0,
    "longitude":0.0
  },
  "locationId":200,
```

```
19    "owningOu":{
20      "id":27,
21      "parent":5,
22      "name":"Systems Seminars",
23      "url":"https://www.gla.ac.uk/schools/computing/research/
24      researchsections/systems-section/",
25      "location":1,
26      "description":"<p>The GLAsgow Systems Section (GLASS) researches
            ↪ parallel and distributed systems, networked systems and
            ↪ (safety-critical) software systems. We have a strong focus on
            ↪ real-world systems, and cover all scales and across the
            ↪ hardware-software spectrum. We contribute to, develop and
            ↪ release open source research software. There are several
            ↪ research groups within the
            ↪ section:</p>\r\n<ul>\r\n<li><span><a
            ↪ href=\"http://www.dcs.gla.ac.uk/research/gpg/\">GPG: The
            ↪ Glasgow Parallelism Group</a></span><span> (led by Phil
            ↪ Trinder)</span></li>\r\n<li><span><a
            ↪ href=\"http://www.dcs.gla.ac.uk/research/networked-systems/\">
27      Networked Systems</a></span> (led by Colin
            ↪ Perkins)</li>\r\n<li><span><a
            ↪ href=\"../../schools/computing/research/researchoverview/
28      systemsengineeringresearchgroup/#d.en.481939\">Systems
            ↪ Engineering</a></span><span> (led by Tim
            ↪ Storer)</span></li>\r\n</ul>\r\n<p>Much of the research we
            ↪ undertake is collaborative and has industrial partners. We
            ↪ work closely with other groups in Computing Science as well
            ↪ as other schools including Engineering. We also work closely
            ↪ with other world leading Universities and many private and
            ↪ public sector organisations (recently: Airbus, Cisco Systems,
            ↪ EDF, Ericsson, GCHQ, IETF, Microsoft Research,
            ↪ NASA).</p>\r\n<p>Systems seminars are usually held on
            ↪ Wednesdays. Everyone from the University of Glasgow and
            ↪ beyond is welcome to attend these talks - see the Events tab
            ↪ for more details. We are happy to hear from anyone that would
            ↪ like to visit us to give a talk.</p>\r\n<p>The Systems
            ↪ seminar coordinators are Natalia Chechina and Magnus
            ↪ Morton.</p>",
29      "shortname":"socs-systems-seminars",
30      "status":"ACTIVE",
31    },
32    "owningOuId":27,
33    "owningGuid":"2426493a",
34    "lastModified":"2021-03-03T20:05:35Z",
35    "lastModifiedGuid":"2426493a",
36    "registrationUrl":"",
37    "intendedAudience":0,
38    "userActionsAllowed":0,
39    "userTagList":[
40      "public"
41    ],
42    "marked":false,
43    "extraLocation":null,
44    "valid":true
45  }
```

Listing C.1: *Extract of JSON response from Samoa Events API.*

D | Final User Evaluation Questionnaire

This appendix provides the final user evaluation questionnaire that was used to collect participant's feedback.

Seminar Roulette Evaluation

The aim of this evaluation is to investigate the suitability and usability of the Seminar Roulette web application. We are performing this evaluation to gather your feedback with the aim to further improve the prototype which has been developed. Interactions on the platform will be associated with your University of Glasgow GUID.

You will be asked to perform a number of tasks and then answer some questions afterwards. The data collected from the questionnaire will be anonymous. Please remember that it is the system, not you, that is being evaluated.

You are welcome to withdraw from the evaluation at any time. If you do, then it will not be possible for you to be debriefed about the purposes of the evaluation.

If you have any further questions regarding this evaluation please contact the system developer, Ollie Gardner (2310049g@student.gla.ac.uk), or the project supervisor, Dr Jeremy Singer (jeremy.singer@glasgow.ac.uk).

* Required

1. Ethical consent *

Check all that apply.

☐ I confirm that I have read the above information, agree to voluntarily take part in this evaluation and that I am over the age of 16.

Figure D.1: *Final user evaluation questionnaire - page 1 of 8.*

Part 1: Background & Tasks to Perform

Seminar Roulette is a web application which allows University of Glasgow academics to discover research seminars taking place at the university. It pulls in seminar data from a variety of web sources and aims to be the primary source for finding seminars which are of interest to you. The system contains a recommender system algorithm which will provide recommendations to you based on how you have rated past seminars.

The platform is currently hosted on a virtual machine internal to the School of Computing Science. It has been made public for the purpose of this evaluation and can be found by navigating to https://howard.dcs.gla.ac.uk/ in your favourite web browser. Please note, you may need to accept the self-signed SSL certificate in order to gain access to the website.

Please carry out the following tasks before continuing to the next stage of this form:

1) Navigate to https://howard.dcs.gla.ac.uk/ in your web browser.
2) The platform can be used without signing in but has a limited feature set. Please take a look at upcoming and random seminars.
3) Login using your University of Glasgow GUID and password by clicking the LOGIN button in the navigation bar.
4) Enter up to 5 of your personal interests.
5) Find an upcoming seminar and click on it to discover more information about it.
6) Filter and sort the seminars to your desired needs.
7) Find at least 2 past seminars and rate them.
8) Navigate to your seminar recommendations.
9) Find a random seminar.
10) Search for a seminar.

After completing the above tasks, please continue to the next stage of this form.

Part 2: Quantitative Data

This section is based off a Post-Study System Usability Questionnaire (PSSUQ).

On a scale between Strongly Agree to Strongly Disagree, please rate the following statements. Please leave blank if the statement is Not Applicable (N/A).Post-Study System Usability Questionnaire (PSSUQ)Post-Study System Usability Questionnaire (PSSUQ)

2. Overall, I am satisfied with how easy it is to use this system.

Leave blank for N/A.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

Figure D.2: *Final user evaluation questionnaire - page 2 of 8.*

3. It was simple to use this system.
Leave blank for N/A.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

4. I was able to complete the tasks and scenarios quickly using this system.
Leave blank for N/A.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

5. I felt comfortable using this system.
Leave blank for N/A.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

6. It was easy to learn to use this system.
Leave blank for N/A.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

Figure D.3: *Final user evaluation questionnaire - page 3 of 8.*

7. I believe I could become productive quickly using this system.
   Leave blank for N/A.

   Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

8. The system gave error messages that clearly told me how to fix problems.
   Leave blank for N/A.

   Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

9. Whenever I made a mistake using the system, I could recover easily and quickly.
   Leave blank for N/A.

   Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

10. The information (such as online help, on-screen messages, and other documentation) provided with this system was clear.
    Leave blank for N/A.

    Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

Figure D.4: *Final user evaluation questionnaire - page 4 of 8.*

11. It was easy to find the information I needed.
Leave blank for N/A.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

12. The information was effective in helping me complete the tasks and scenarios.
Leave blank for N/A.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

13. The organisation of information on the system screens was clear.
Leave blank for N/A.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

14. The interface of this system was pleasant.
Leave blank for N/A.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

Figure D.5: *Final user evaluation questionnaire - page 5 of 8.*

15. I liked using the interface of this system.

Leave blank for N/A.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

16. This system has all the functions and capabilities I expect it to have.

Leave blank for N/A.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

17. Overall, I am satisfied with this system.

Leave blank for N/A.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	◯	◯	◯	◯	◯	◯	◯	Strongly Disagree

Part 3: Qualitative Feedback

Qualitative data will be collected to get a sense of how Seminar Roulette could be improved in future iterations of the software.

Figure D.6: *Final user evaluation questionnaire - page 6 of 8.*

18. Whilst using Seminar Roulette, did you encounter any errors? If so, please describe the error and how it arose. *

19. Do you think that Seminar Roulette could be a useful platform to provide seminar recommendations to university academics? If no, please specify why not. *

20. Do you have any suggestions on how the Seminar Roulette platform could be improved? *

21. Are there any additional features which you would like to see in Seminar Roulette? *

Figure D.7: *Final user evaluation questionnaire - page 7 of 8.*

22. In five words, how would you describe the overall design and functionality of Seminar Roulette? *

Debrief

The main aim of the experiment was to investigate the suitability and usability of the Seminar Roulette web application. Interactions on the platform will be associated with your University of Glasgow GUID. If you have any further questions or concerns about this evaluation then please contact the system developer, Ollie Gardner (2310049g@student.gla.ac.uk), or the project supervisor, Dr Jeremy Singer (jeremy.singer@glasgow.ac.uk).

Thank you for taking the time to participate in this evaluation.

Figure D.8: *Final user evaluation questionnaire - page 8 of 8.*

Bibliography

Alexa. Competitive analysis, marketing mix and traffic. https://www.alexa.com/siteinfo/gla.ac.uk, 2021. Last accessed: 2021-02-28.

A. S. Arefin, C. Riveros, R. Berretta, and P. Moscato. Gpu-fs-knn: a software tool for fast and scalable knn computation using gpus. *PloS one*, 7:e44000, 08 2012. doi: 10.1371/journal.pone.0044000.

Balsamiq. Balsamiq. https://balsamiq.com/, 2021. Last accessed: 2021-03-01.

J. Bennett, S. Lanning, and N. Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

Bookitbee. Bookitbee. https://event.bookitbee.com/university-of-glasgow-40, 2021. Last accessed: 2021-03-25.

business.com Member. How page load speed affects customer behavior. https://www.business.com/articles/website-page-speed-affects-behavior/, 2020. Last accessed: 2021-02-28.

Carleton College. Dimensionality reduction and the singular value decomposition. https://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/svd.html, 2007. Last accessed: 2021-03-02.

Coverage. Coverage. https://pypi.org/project/coverage/, 2021. Last accessed: 2021-03-16.

M. Dabbs. The fundamentals of web application architecture. https://reinvently.com/blog/fundamentals-web-application-architecture/, 2019. Last accessed: 2021-04-02.

Datanyze. Event management market share table. https://www.datanyze.com/market-share/event-management--57, 2021. Last accessed: 2021-02-16.

Django. Django. https://www.djangoproject.com/, 2021. Last accessed: 2021-03-10.

Django REST Framework. Django rest framework. https://www.django-rest-framework.org/, 2021. Last accessed: 2021-03-10.

O. Djuraskovic. ecommerce statistics, facts, and trends 2021. https://firstsiteguide.com/ecommerce-stats/, 2021. Last accessed: 2021-02-19.

Docker. Docker. https://www.docker.com/, 2021. Last accessed: 2021-03-10.

W. W. Eckerson. Three tier client/server architecture: Achieving scalability, performance, and efficiency in client server applications. *Open Information Systems*, 10(1), 1995.

Eventbrite. Eventbrite. https://www.eventbrite.co.uk/, 2021. Last accessed: 2021-01-26.

Flask. Flask. https://flask.palletsprojects.com/en/1.1.x/, 2021. Last accessed: 2021-03-10.

B. George and L. Williams. A structured experiment of test-driven development. *Information and Software Technology*, 46(5):337–342, 2004. ISSN 0950-5849. doi: https://doi.org/10.1016/j.infsof.2003.09.011. URL https://www.sciencedirect.com/science/article/pii/S0950584903002040. Special Issue on Software Engineering, Applications, Practices and Tools from the ACM Symposium on Applied Computing 2003.

GitHub. Github. https://github.com/, 2021a. Last accessed: 2021-03-09.

GitHub. Github actions. https://github.com/features/actions, 2021b. Last accessed: 2021-03-09.

C. A. Gomez-Uribe and N. Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), Dec. 2016. ISSN 2158-656X. doi: 10.1145/2843948. URL https://doi.org/10.1145/2843948.

Google. Lighthouse. https://developers.google.com/web/tools/lighthouse, 2021a. Last accessed: 2021-03-16.

Google. Yapf. https://github.com/features/actions, 2021b. Last accessed: 2021-03-09.

G. F. A. Harding, P. M. Jeavons, and A. S. Edson. Video material and epilepsy. *Epilepsia*, 35 (6):1208–1216, 1994. doi: https://doi.org/10.1111/j.1528-1157.1994.tb01791.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1528-1157.1994.tb01791.x.

F. Isinkaye, Y. Folajimi, and B. Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015. ISSN 1110-8665. doi: https://doi.org/10.1016/j.eij.2015.06.005. URL https://www.sciencedirect.com/science/article/pii/S1110866515000341.

Kef. What cycle does the react component have? https://programmer.group/what-cycle-does-the-react-component-have.html, 2019. Last accessed: 2021-03-10.

J. Khan, I. U. Rehman, Y. Khan, I. J. Khan, and S. Rashid. Comparison of requirement prioritization techniques to find best prioritization technique. *International Journal of Modern Education and Computer Science*, 7:53–59, 2015.

Z. Kurtanović and W. Maalej. Automatically classifying functional and non-functional requirements using supervised machine learning. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 490–495, 2017. doi: 10.1109/RE.2017.82.

S. Liu. Global market share held by the leading web browser versions as of january 2021. https://www.statista.com/statistics/268299/most-popular-internet-browsers/, 2021. Last accessed: 2021-02-28.

Material Design. Material design. https://material.io/design, 2021. Last accessed: 2021-03-09.

Material-UI. Material-ui. https://material-ui.com/, 2021. Last accessed: 2021-03-09.

Meetup. Meetup. https://www.meetup.com/, 2021. Last accessed: 2021-03-25.

NLTK. Natural language toolkit. https://www.nltk.org/, 2020. Last accessed: 2021-03-11.

Pandas. Pandas. https://pandas.pydata.org/, 2021. Last accessed: 2021-03-15.

S. Perez. Eventbrite confirms the coronavirus outbreak will materially impact its business. https://techcrunch.com/2020/03/16/eventbrite-confirms-the-coronavirus-outbreak-will-materially-impact-its-business/, 2020. Last accessed: 2021-02-16.

G. Pipis. Item-based collaborative filtering in python. https://predictivehacks.com/item-based-collaborative-filtering-in-python/, 2020. Last accessed: 2021-02-28.

PostgreSQL. Postgresql. https://www.postgresql.org/, 2021. Last accessed: 2021-03-10.

Prettier. Prettier. https://prettier.io/, 2021. Last accessed: 2021-03-09.

React. React. https://reactjs.org/, 2021. Last accessed: 2021-03-09.

Samoa Events. Samoa events. https://samoa.dcs.gla.ac.uk/events/, 2021. Last accessed: 2021-01-13.

J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, page 158–166, New York, NY, USA, 1999. Association for Computing Machinery. ISBN 1581131763. doi: 10.1145/336992.337035. URL https://doi.org/10.1145/336992.337035.

SciPy. Scipy. https://www.scipy.org/, 2021. Last accessed: 2021-03-15.

Shibboleth. Shibboleth. https://www.shibboleth.net/, 2021. Last accessed: 2021-01-26.

B. Smith and G. Linden. Two decades of recommender systems at amazon.com. *IEEE Internet Computing*, 21(3):12–18, 2017. doi: 10.1109/MIC.2017.72.

StackShare. Who uses flask? https://stackshare.io/flask, 2021. Last accessed: 2021-03-15.

J. Stoll. Number of netflix paid subscribers worldwide from 1st quarter 2013 to 4th quarter 2020. https://www.statista.com/statistics/250934/quarterly-number-of-netflix-streaming-subscribers-worldwide/#:~:text=How%20many%20paid%20subscribers%20does,Netflix's%20total%20global%20subscriber%20base., 2020. Last accessed: 2021-02-18.

Trello. Trello. https://trello.com/, 2021. Last accessed: 2021-03-09.

I. Tuchkov. Color blindness: how to design an accessible user interface. https://uxdesign.cc/color-blindness-in-user-interfaces-66c27331b858, 2018. Last accessed: 2021-03-02.

University of Cambridge. University of cambridge talks. https://talks.cam.ac.uk/document/Who%20we%20are, 2006. Last accessed: 2021-02-25.

University of Glasgow. Staff numbers. https://www.gla.ac.uk/explore/facts/staffnumbers/, 2020a. Last accessed: 2021-02-28.

University of Glasgow. Student numbers. https://www.gla.ac.uk/explore/facts/studentnumbers/, 2020b. Last accessed: 2021-02-28.

University of Glasgow. University of glasgow website. https://www.gla.ac.uk/, 2021a. Last accessed: 2021-01-26.

University of Glasgow. Brand toolkit. https://www.gla.ac.uk/myglasgow/staff/brandtoolkit/brandelements/colours/, 2021b. Last accessed: 2021-03-02.

University of Glasgow. Gdpr compliance checklist for databases storing personal data. https://www.gla.ac.uk/myglasgow/dpfoioffice/a-ztopics/databasechecklist/, 2021c. Last accessed: 2021-03-08.

University of Strathclyde. sbs.strath. https://www.sbs.strath.ac.uk/feeds/listseminars.aspx, 2021. Last accessed: 2021-03-25.

Will T. Pssuq (post-study system usability questionnaire). https://uiuxtrend.com/pssuq-post-study-system-usability-questionnaire/, 2021. Last accessed: 2021-03-16.

S. Won and S. Westland. Colour meaning and context. *Color Research & Application*, 42(4):450–459, 2017. doi: https://doi.org/10.1002/col.22095. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/col.22095.

Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, ACL '94, page 133–138, USA, 1994. Association for Computational Linguistics. doi: 10.3115/981732.981751. URL https://doi.org/10.3115/981732.981751.

Yammer. Yammer. https://www.yammer.com/glasgow.ac.uk/#/home, 2021. Last accessed: 2021-03-25.