



Final Report

**Sift: A real-estate mobile application with a custom
recommendation system**

Edward Day

Computer Science (Digital & Technology Solutions)

2022/2023

COMP3932 Synoptic Project

The candidate confirms that the following have been submitted:

Items	Format	Recipient(s) and Date
<i>Final Report</i>	<i>PDF file</i>	<i>Uploaded to Minerva (23/04/23)</i>
<i>Participant Consent Forms and Results</i>	<i>File archive</i>	<i>Uploaded to Minerva (22/04/23)</i>
<i>GitHub Code Repository</i>	<i>URL</i>	<i>Sent to supervisor and assessor (24/03/23)</i>
<i>API Documentation</i>	<i>URL</i>	<i>Sent to supervisor and assessor (24/03/23)</i>
<i>User Story Cards</i>	<i>Excel File</i>	<i>Sent to supervisor and assessor (24/03/23)</i>
<i>Demonstration Video</i>	<i>URL</i>	<i>Sent to supervisor and assessor (24/03/23)</i>

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student): Edward Day

Summary

This project is a software product that is centred around a mobile application which aims to ease the process of finding a property to rent.

The first major component of the project is an iOS mobile application which allows the system's users to create an account and browse through property listings using various methods of navigation, with a heavy focus on providing tailored recommendations to the user. These recommendations are generated by the system using a variety of information about the user and their preferences.

The second main component of the system is the backend which has been built using Node.js and interfaces with a MongoDB database. The backend provides a REST API which it uses to communicate with the mobile application and uses the information from the database in combination with a custom recommendation system to provide the app with all of the data that it requires.

This report, in combination with the deliverables provided, gives details of the full process that was involved with developing this system, outlining the reasoning behind each of the key design decisions and the full details of how each piece of functionality has been implemented. The report is concluded with a full evaluation which reviews what has been developed and the effectiveness of the system, as well as detailing future plans for how the system could be further developed.

Acknowledgements

I would like to use this section to give thanks to Professor Andy Bulpitt who has supervised this project. His guidance has helped hugely, and the support that he has provided throughout my entire time at university couldn't be more appreciated. His expertise and insight have been invaluable to me.

I would also like to thank my family, girlfriend and friends for the constant support, inspiration and encouragement that they have given me this year. The success of this project has been hugely influenced by the support received, and I couldn't be more grateful to everyone who has been there for me throughout the final year of my studies.

Table of Contents

Summary	3
Acknowledgements.....	4
Chapter 1 - Introduction	7
1.1 - Background	7
1.2 - Aim	7
1.3 - Objectives	8
1.4 - Deliverables	8
Chapter 2 - Project Plan and Methodology	9
2.1 - Methodology	9
2.2 – Risk Mitigation.....	10
2.3 - Gantt Chart.....	10
Chapter 3 - Initial Research.....	11
3.1 - Data	11
3.1.1 - Property Listing Data	11
3.1.2 - User Data	12
3.2 - Technology Stack.....	13
3.2.1 - Frontend (SwiftUI)	13
3.2.2 – Backend (Express JS)	13
3.2.3 - Database (MongoDB Atlas)	15
3.2.4 – Recommendation System (Python SciKit Learn)	16
3.2.5 - Deployment (DigitalOcean)	16
3.2.6 – Other Libraries and Frameworks	17
3.3 – Recommendation System	18
3.3.1 – Background	18
3.3.2 – Justification	18
Chapter 4 – Methods.....	19
4.1 - Recommendation System.....	19
4.1.1 - Considerations	19
4.1.2 – Content Based Design.....	21
4.1.3 – Collaborative Filtering Design.....	22
4.2 – Backend	25
4.2.1 - Overview	25
4.2.2 – Database	25
4.2.3 – Authentication	26
4.3 – Application / Frontend.....	27
4.3.1 – Development Process	27
4.3.2 – UI/UX Design.....	27
4.4 – Property Listing Scraper.....	29
Chapter 5 – Results and Evaluation	30
5.1 - Main Application Functionality	30

5.2 - Content-Based Filtering Recommender System	31
5.3 - Collaborative Filtering Recommender System	31
Chapter 6 – Conclusion	34
6.1 – Legal/Ethics	34
6.1.1 – Rightmove Property Data	34
6.1.2 – User Data (Data Protection Act)	34
6.2 – Finished System Overview	35
6.3 – Further Developments	36
6.2.1 – Property Listing Data	36
6.2.2 – Application Functionality	37
6.2.3 – Recommendation System	37
6.4 – Self Appraisal	38
Chapter 7 – Appendix	39
7.1 – References	39
7.2 - Evaluation Surveys/Results	41
7.2.1 - Main Application Functionality	41
7.2.2 – Content Based Filtering Evaluation	42
7.2.3 – Collaborative Filtering Evaluation	43
7.3 - Example User Story Card	45
7.4 – User Story Backlog.....	45
7.5 - Terms and Conditions.....	46
7.6 – Participant Consent Form	47
7.7 – Other Figures and Tables.....	48
7.8 – External Links.....	55

Chapter 1 - Introduction

1.1 - Background

Within the past decade, the UK has seen significant growth within its rental market, with the latest census data showing an increase in the proportion of households that rented their accommodation over buying to 37.3% in 2021 (Census, 2021). Alongside this, Rightmove has stated in their Q3 2022 rental activity report that competition amongst tenants to secure a property is at a record high, with tenant demand up 20% compared to 2021, and the number of available properties is down 9%, making it an increasing struggle for tenants to find a property which meets their needs (Rightmove, 2022). This current climate which is seeing an insufficient supply for the demand and an increase in prices makes the already stressful process of finding a property to rent an even more daunting task, and as a result tenants are more likely to have to make compromises in the features of the property which they are looking for. These combined statistics emphasise the fact that finding a property is not easy, and whilst the underlying problem of the number of available properties may require a longer-term solution, there may be a gap in the market for a system which could help to simplify the process and guide tenants towards options that they may not have otherwise considered.

This project will detail the development of a real-estate platform called Sift which will feature a custom recommendation system with the view of increasing user satisfaction with their chosen property. According to DeVito et al. (2017), changes to many of the leading social media platforms indicate that a suggestion driven method of navigation may be the direction that content exploration may be heading in. This will be explored in further detail later within this project. This will be developed in the form of a mobile application which will interface with a hosted API to communicate with the backend, providing the app with all the data that it will require.

1.2 - Aim

The aim of this project is to create a real estate mobile application where users can browse through property listings. Based off the information that they have provided and how they have interacted with other listings, they will be able to use a recommendation system which will direct them towards listings which have been identified as likely to be of interest to them. As part of the project, the factors which will influence these recommendations must be considered, as well as the methods in which users will be able to explore the content. It must also be considered how the system will gather the information about the property listings that are provided to users.

1.3 - Objectives

- Gather data about current property listings
- Develop a way to identify and recommend content that users are likely to be interested in
- Develop an iOS mobile app which includes user authentication that allows users to browse and interact with listings
- Consider and implement a way for users to be presented with and navigate content which is deemed to be relevant to them
- Design and implement a modern and intuitive frontend user interface that makes content navigation simple
- Deploy the backend to the system onto a hosted platform so that the mobile application can function from anywhere

1.4 - Deliverables

- Source code and results of recommendation system
- Collection of User Story Cards which provide full implementation details for each piece of functionality
- Deployed API and documentation detailing all endpoints alongside their parameters and responses
- Source code and demonstration video of full-stack application
- Access to and demonstration of completed final application
- Final report

Chapter 2 - Project Plan and Methodology

2.1 - Methodology

This project has been developed using a similar methodology to the standard agile workflow (Jeong et al., 2008), with some adaptations to allow it to better suit this individual project, as outlined in Figure 1.

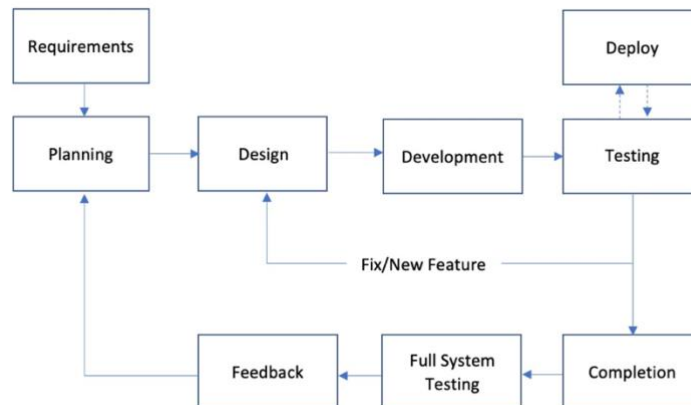


Figure 1

To begin, the high-level requirements of the product and the functionality that the product would need were captured, which were shaped around both our initial background research, and primary research that was carried out by interviewing renters from a variety of backgrounds.

The project then moved into a planning phase which had similarities to a classic agile inception phase. During this period each of the requirements was further broken down to come up with a rough idea of how they would be implemented as well as the technologies that would be required to do so, forming the initial backlog of user stories. As this is such a large project with so many different elements, it was important not to go into too much detail at this stage, as there was a high likelihood that certain parts would require changes later down the line to ensure that each part would integrate with everything else correctly.

The planning stage was followed by the development stage, which used a design-first iterative approach to gradually build out the functionality. This involved choosing a feature and analysing the requirements in more detail using user story cards so that it was clear exactly how it would be implemented before moving onto the development and writing the actual code. After implementation, it would then be tested against the acceptance criteria to ensure that it worked as expected, and depending on the feature, deployed to a web server to ensure that the system still worked when being ran externally. This approach allowed any problems encountered to be fixed as early as possible whilst the code was fresh to mitigate the risk of issues impacting other parts of the system later down the line. After each feature

was successfully completed and accepted to be used in the system, this process would then be repeated.

Once all of the initial development was completed, the system was then tested as a whole against the acceptance criteria of the initial system requirements to ensure that these had been met. The system was first evaluated personally and then evaluated by other users by asking for feedback on both the current functionality and what other features they thought would be most beneficial. Based on this evaluation and feedback, another planning phase was started so that the requirements could be set out for how the system would be refined and further developed in the remaining time. The main piece of functionality that was implemented in this refinement phase was the collaborative filtering recommender system.

After all development was completed, the system was thoroughly tested and evaluated for a final time. The results of the system developed make up section 5 and the results of the evaluation make up section 7 within this report.

2.2 – Risk Mitigation

Figure 2 outlines the risks considered at the start of this project and the steps taken for each to minimize them.

Risk	Likelihood	Impact	Actions Taken to Minimize Risk
Would not be able to complete project due to scope being too large	High	High	Focus on MVP and prioritise most important functionality, plan full timeline
Recommendation system would not provide suitable results	Moderate	Moderate	Start with a simple system and add complexity later
Unable to get data to use for system	Moderate	High	Put this as first priority and plan multiple approaches for if initially unsuccessful
Unable to get separate parts of system to interface with each other	Low	High	Extensive research before starting development and be able to prove functionality of each part individually

Figure 2

2.3 - Gantt Chart

Figure 3 shows a Gantt chart which outlines the initial timeline set out for how the project would progress.

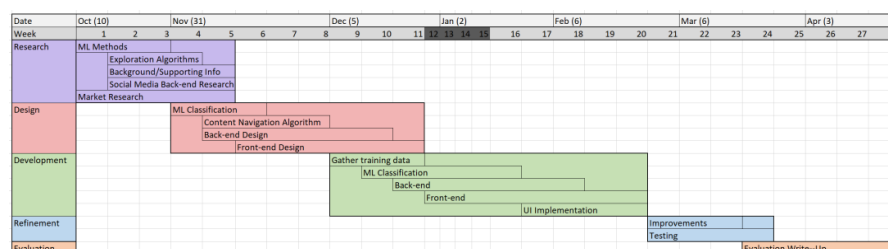


Figure 3

Chapter 3 - Initial Research

3.1 - Data

As the main function of the application is to present users with information about properties, access to a large set of property listing data is required. This is needed to build the content-based filtering recommendation system.

It is also important to consider the user data that is needed to provide them with suitable recommendations, and to bear in mind that collecting additional information about users and their in-app interactions may be useful if user-user recommendations are to be implemented. When collecting data, it is essential to ensure that this complies with any ethical requirements, and that permission is given to gather and share the data that is being collected. This will have different implications for both the property data and user data, both of which will be covered in more detail later in the ethics section.

3.1.1 - Property Listing Data

An initial consideration was where the system would get the data to be used for the property listings. The initial idea was to use the site Kaggle which has premade open datasets designed to be used by data scientists for machine learning tasks and features various datasets with information about properties. However, as the app is going to display data for current property listings which are available, historical data will not be appropriate. Also, the Kaggle datasets do not contain all the additional information that would be needed about a property available to rent, such as images, letting agents or deposits. This led to the conclusion that the most appropriate way to get data would be to create the system's own database of property listings. This could be achieved by reaching out to estate agents manually, using a public API, or by creating a web scraper to retrieve the necessary data from another site.

If this app was being used for production, the ideal option would be to seek to partner with estate agents, and they would provide details of their listings in return for the exposure that the app would give to their properties, in turn making it easier for them to find prospective tenants. However, due to the scope of this project, a more appropriate solution was to use existing up-to-date data which could be gathered from other property listing sites such as Rightmove or Zoopla. As well as requiring access to data for individual listings within the app, the system will also need direct access to a complete database of listing data in order to process and create an effective content-based recommendation system.

The final decision was to gather the data from Rightmove which is the leading platform in the UK for residential properties, with over 1 million properties being listed each month. As Rightmove does not have a public API, the data was gathered directly from the site using a

custom web-scraper as outlined in section 4.4. This has legal/ethical implications which are discussed in section 6.1.

3.1.2 - User Data

When the user first downloads the app, they will be prompted to sign up and create a profile so that they can use the application. A simplified sign-up process is essential as studies have shown that the average user has a very limited attention span and will quickly lose patience with a sign-up process that is overly complicated or takes too long. According to a study by Statista (2022), 25% of mobile app users abandon an app after only one use, with sign-up processes cited as one of the top reasons for abandonment. For this reason, the required data will be kept to a minimum, and make the second page where they can provide us with more details about themselves optional. By providing additional information, users will help the system to provide more tailored recommendations by providing input for a potential collaborative filtering system. The explicit data to be captured during sign up is as shown in Figure 4.

Field Location	Field Name	Mandatory	Field Type	Condition
Initial Sign Up	First Name	TRUE	String	
	Last Name	TRUE	String	
	Email	TRUE	Email	
	Mobile	FALSE	Mobile	
	Profile Image	FALSE	Image	
	About	FALSE	String	
	Date of Birth	FALSE	Date	
Additional Details	Type of Renter	FALSE	Typelist	
	Salary	FALSE	Integer	
	Profession	FALSE	Typelist	
	Add Current Property Details	TRUE	Boolean	
Current Property	Address	TRUE		Add current property = Yes
	Bedrooms	TRUE		Add current property = Yes
	Bathrooms	TRUE		Add current property = Yes
	Are renting?	TRUE		Add current property = Yes
	Current Price	FALSE		Renting = Yes
	Type of property	TRUE		Add current property = Yes

Figure 4

In addition to this explicit data, in-app interactions will also be tracked which can provide value when looking to further build out our system – in particular, if analytics functionality was going to be added for estate agents. The implicit in-app data to be captured and sent to the backend is shown within Figure 5.

Data	Fields
Listing Views	User ID, Property ID, Date
Listing Enquiries	User ID, Property ID, Agent ID, Details, Date

Figure 5

As there is no immediate urgency for the backend to receive this data, the interactions can be tracked within the app with the data being held in an Interactions model, and periodically sent to the backend to reduce the number of requests that are made.

3.2 - Technology Stack

3.2.1 - Frontend (SwiftUI)

UIKit and SwiftUI are the two native frontend frameworks for iOS app development, both provided by Apple. UIKit is the original frontend framework for iOS app development, released in 2008. It provides a wide range of tools for building user interfaces, including storyboards and interface builder. With UIKit, developers can create complex and customizable user interfaces using Swift or Objective-C. However, UIKit can be more difficult to learn than some other frameworks, and it requires more code to achieve the desired results. SwiftUI, on the other hand, is a newer frontend framework for iOS app development that was introduced by Apple in 2019. It is designed specifically for use with Swift and provides a more efficient and intuitive way of building user interfaces. SwiftUI uses a declarative syntax, which means that developers can define the UI in a more concise and expressive way, reducing the amount of code required. SwiftUI also provides a range of powerful features, such as animations and transitions, which are not available in UIKit.

After careful consideration of the available options, the decision was made to use SwiftUI as the frontend framework for the iOS app. Firstly, SwiftUI is a more modern and efficient framework compared to UIKit. It is faster and more intuitive, allowing developers to create advanced UI designs with less code (Wiertel et al., 2021). Another advantage of SwiftUI is its declarative syntax, which reduces the amount of code required to define the user interface and makes it easier to learn for developers, which will be advantageous in a project which involves a wide range of new and unfamiliar technologies (Gilchrist, E., 2021). This will in turn make development faster and more efficient, particularly when creating complex UI designs or animations that aim to improve the overall user experience within the app. Finally, as SwiftUI is Apple's latest framework, they will continue to support it for years to come, meaning that our application will be unlikely to have any reason to move to a different framework in the near future.

In conclusion, while UIKit is a powerful and capable frontend framework for iOS app development, SwiftUI is a more modern, efficient, and future-proof option that provides a more intuitive way of building user interfaces. Therefore using SwiftUI will enable creating a highly functional and user-friendly application that meets the needs of its users.

3.2.2 – Backend (Express JS)

The iOS app will then connect to a server, which will handle any requests, gathering necessary data from a database and returning it to the app using a REST API. Three of the most popular backend frameworks, Express, Django and Flask were considered as options to use for the system (Statista, 2022).

When using JavaScript, Node.js would be used as the server-side runtime environment and Express.js as the server-side application framework. This combination commonly uses MongoDB as the database for the application as it is a NoSQL option and instead the queries are written in JavaScript which fits nicely with the rest of the code.

If the decision was made to use Python, Django or Flask would be likely choices for the backend framework. Both of these have advantages and disadvantages; however, Flask is generally considered a much more lightweight framework. Although Flask is great for small applications or simple API's, Django offers much more potential and is better suited to more complex projects. It also provides many out of the box features which can be used to reduce development effort needed.

Whilst any of these three choices would have been suitable for our project, Express was chosen as the framework to be used for the backend. Some of the primary reasons for this were due to its simplicity to learn, the wide range of documentation and tutorials available, and the productivity benefits that it offers with its wide range of HTTP methods and modular program structure. As mentioned previously, due to the scope of this project, it was crucial to make development choices that would help ensure that the project would be able to be delivered within the given timeframe. Performance of the frameworks was also initially considered, although it was decided that this was not an important factor as even slower languages such as Python are good enough for medium-sized web applications when ran on moderate hardware. This is backed up by the fact that frameworks such as Django and Flask are used by some of the largest web applications such as Netflix, Reddit, and Instagram (Muittari, J., 2020).

The recommendation system will run as a Python script which makes a custom backend preferable over services such as Firebase or MongoDB Realm as it means that all requests can be processed from one system, reducing the overall complexity.

Both above languages would be suitable to use for the backend and each have their own benefits. The main benefit of using Python would initially appear to be the fact that this will be the same language that the machine learning content identification algorithm will be written in. However, there are Node.js packages which allow Python scripts to be integrated within code if a JavaScript based approach is used so this will not be an issue. One of the main reasons that developers often choose JavaScript is because it allows them to use the same language for both the front and back end, although again, this will not be a benefit to due to the fact that Swift will be used for the frontend of the mobile app.

Overall, it comes down mostly to personal preference as both are well established solutions, which are both used by the main large social media platforms. The JavaScript based Node.js stack was chosen to be used for the majority of the functionality due to having the most prior coding experience with it.

3.2.3 - Database (MongoDB Atlas)

All of the data for the system is managed using MongoDB – a non-relational database which is based on JSON documents. The database holds a set of collections which have no predefined schemas and store the data as BSON documents (binary encoded JSON like objects) (Khan et al., 2013). A document is a set of fields and can be thought of as a row in a collection. It can contain complex structures such as lists, or even an entire document. Each document has an ID field, which is used as a primary key and each collection can contain any kind of document.

The decision to use MongoDB for the database instead of a traditional SQL-based option was made for multiple reasons. Firstly, as the database uses JSON-like documents, this makes it simple to store both structured and unstructured data which then can be mapped directly to native objects within our other chosen programming languages, eliminating the need for data normalisation. Moreover, as documents are not required to conform to a specific schema, it adds a huge amount of flexibility when developing and scaling the system. This is a large benefit when working on such a quickly evolving project which has a high potential for changes in requirements as it allows fields to either be modified or added during development with ease without requiring a large amount of work.

Following on from this, another benefit of MongoDB over SQL alternatives is their modern and intuitive query language. This comes down to personal preference and despite one of the hallmarks of SQL being its rich query language, MongoDB has its own unique rich querying capabilities which uses a “cascade” of operator/operand structures, typically in name:value pairs (Moschetti, B., 2015). The benefit of this is that the same techniques used to manipulate data going into and coming out of MongoDB can be used to construct, manipulate, and “parse” query expressions. Furthermore, as the query expression is a structured cascade and not a single string, it makes it easy to incrementally add subexpressions to the query without first having to break it apart into component pieces, adding even more flexibility to the system as it evolves.

Another key reason for choosing MongoDB was due to their managed database service – MongoDB Atlas. By using a DBaaS (managed database service) such as MongoDB Atlas, a cluster can be set up within minutes, taking care of all of the administrative processes which

would normally be required behind the scenes such as provisioning resources, setting up clusters and scaling the system. This ensured that a sufficient time allocation could be dedicated to developing the functionality of the system which was a key priority due to the restricted timeframe and large scope of this project.

3.2.4 – Recommendation System (Python SciKit Learn)

In order to produce accurate recommendations for properties, the system uses machine learning frameworks to help with performing the necessary tasks such as classification, clustering, and pre-processing. The most popular machine learning frameworks such as SciKit Learn, TensorFlow, Keras and PyTorch are all written in Python, and the system utilises different libraries for each type of recommendations being made.

For content-based filtering, the recommendation system will simply use SciKit-Learn as this type of system does not require a complex trained model. Instead, it works by calculating a similarity metric such as the cosine similarity or Euclidean distance, which SKL provides as built in functions.

For implementing collaborative filtering, the choice of library to be used depends on the technique being used for providing recommendations. As the decision was made to use a SVD model-based design for the CF recommender, the SurPRISE (Simple Python Recommendation System Engine) library was chosen as it provides multiple ready to use prediction algorithms. This was the most obvious choice for an initial CF implementation, as it is specifically designed for building recommendation engines and provides a simple API which makes it easy to load, pre-process and train recommendation models.

Tensorflow was also considered as an option for building a machine learning model, as there is a vast amount of documentation and multiple examples of similar systems that have been built using it. Although this could have been referenced to help with building the recommendation system, using Tensorflow would have still added a lot more complexity because it is a more general-purpose library. Due to the lack of real data for the CF recommendations, which is discussed further throughout this report, this part of the system features as more of an initial proof of concept, so using a simpler approach to demonstrate the functionality seemed more appropriate.

3.2.5 - Deployment (DigitalOcean)

DigitalOcean was chosen as the cloud hosting provider for the backend to be deployed to. DigitalOcean is a cloud computing platform that provides scalable and affordable virtual private servers (VPS) for developers, businesses, and other organizations. It was founded in 2011 with a mission to simplify web infrastructure by offering cloud servers, object storage, managed databases, and other related services.

The backend was deployed to a DigitalOcean droplet which is their name for their scalable virtual machines that can be created and setup within seconds. The configuration chosen for this project was the most basic tier which provides a shared CPU, 1GB of memory and 25GB of storage, costing approximately \$6.00 per month. For the purpose of this project, this is a sufficient amount of computing power as the system will not have a large userbase so will not need to deal with a significant number of concurrent requests. However, if the system did need to be scaled in the future, this could be done by simply selecting a different plan within their web interface. They also provide additional scaling solutions such as load balancing, horizontal scaling, and Kubernetes cluster management which could be considered if the application grew and formed a larger userbase.

There are multiple other cloud hosting providers which could have been used as an alternative, with some of the most popular options being AWS, Microsoft Azure, and Google Cloud (Richter, F., 2022). It could be argued that one of these would be a more suitable choice for an application that is to be used on a large scale, as the resources that they have available due to their size and popularity allows them to provide unmatched performance, infrastructure, and reliability. However, the performance offered by DigitalOcean is more than adequate for any small to mid-sized application, and the decision to use it was mostly due to its affordability and simplicity.

3.2.6 – Other Libraries and Frameworks

Figure 6 details the other main libraries and frameworks that are used by the system.

Library	Reasoning
Pandas – Python	Provides easy to use data structures for manipulating database data within Python code
Numpy - Python	Provides various mathematical functions to help with the recommendation system
Pickle - Python	Persist data that has been calculated for the recommendation system
BeautifulSoup - Python	Parse HTML documents to extract data for web scraper
JWT - NodeJS	Generate and validate JSON Web Tokens for user authentication
Mongoose - NodeJS	Interface with MongoDB database and translate database objects to code
Bcrypt - NodeJS	Provides password hashing function
Google OAuth2Client - NodeJS	Use Google access token to retrieve protected user data from Google and to verify users' identities
Lottie - iOS	Provides framework for displaying .lottie animation files
Kingfisher -iOS	Optimised library for downloading and displaying images from the web
GoogleSignIn - iOS	Provides the functionality for adding sign-in with Google

Figure 6

3.3 – Recommendation System

3.3.1 – Background

Over recent years, there has been a huge change in the way that users consume online information, with algorithmically driven content curation becoming increasingly common across online platforms (DeVito et al., 2017). This is being applied across a variety of forms of online media, ranging from product recommendations within online shopping sites (Hwangbo et al., 2018), to social media applications. TikTok is a prime example of this which completely takes content navigation out of the user's control and presents them with tailored content that the system thinks they will be interested in. Due to the popularity of its algorithm, the app has become one of the world's fastest growing and most popular social media platforms.

3.3.2 – Justification

When deciding on a property, renters take a variety of its attributes into account such as the price, size and location (Van Ham, M., 2012). Tenants are then seen to choose a property that best fits the attributes that they are looking for (Earnheart, D., 2002). Recommendation systems which feature custom algorithms can be used to identify the trends that users make in their decision-making process (Deldjoo et al., 2020), and can then be used to provide further suggestions which fit these patterns, with the view of making the content more relevant, and increasing the likelihood of it being of interest to them. The principle of least effort (Beck, S.J., 1951) states that individuals adopt the course of action that involves the least work, even if it means sacrificing quality or quantity of information. Implementing a personalised recommendation system will navigate users directly to the content that they want without requiring any additional effort. By doing this, the system will aim to go against this with the view of reducing the need for them to sacrifice on the quality of the information that they consume. Additionally, the concept of information overload (Herbig et al., 1994) outlines that if users are presented with too much information, then they struggle to locate what they need most and may overlook what they consider critical. This applies directly to the problem of finding a property where renters are faced with a huge number of options to search through, so a content recommendation algorithm will aim to remedy this by sifting out the unnecessary information. As a result, this will help to direct the user towards properties that have a higher likelihood of being suitable for them. If the accuracy of recommendations and the proportion of relevant content that the user receives can be improved, then this will have a significant effect on overall user satisfaction when using the application (Liang et al., 2006).

Chapter 4 – Methods

The whole system is made up of multiple components and is designed with the following structure shown in Figure 7.

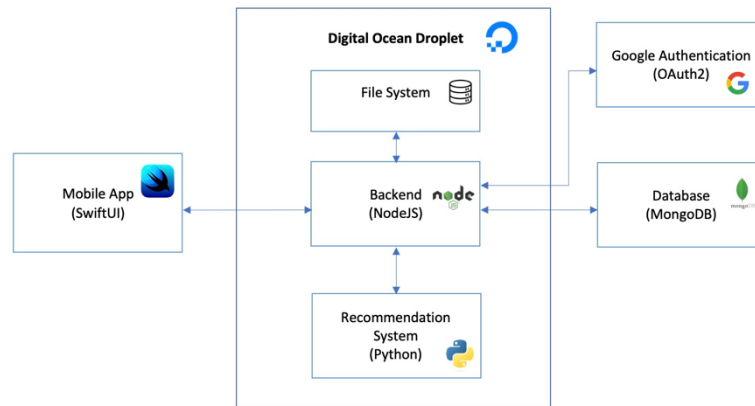


Figure 7

This section breaks down the functionality and design of each of these components.

4.1 - Recommendation System

4.1.1 - Considerations

Although recommendation systems can have a significantly positive effect on the user experience when using a system as outlined above, they are not without their drawbacks which must be considered during development to ensure that the application meets its user's needs.

Firstly, many users have shown resistance to algorithmic change. For example, when Instagram updated their feed to use a more complex algorithm instead of simply displaying posts in chronological order, they received significant backlash from many of their users (Amy Kraft, 2016). Users have shown resistance to new algorithms for a wide variety of reasons, but some of the main ones include a lack of trust in the recommendations provided being suitable, a lack of understanding as to how they work, and a lack of choice in whether they want to be provided with content in this way (DeVito et al., 2017). By taking the above into account when designing the system, it can aim to minimise the resistance that its algorithm will receive from its users. A lack of choice will not be an issue within the application as the recommendation system based approach to browsing through properties will only be one method of navigation within our application, and users will still have the ability to manually search for properties without having their results being influenced by an algorithm. To ensure that users are not put off by a lack of understanding, it will also be

made clear to them what the algorithm bases their results off. In this case, it will be providing them with properties that have similar attributes to those that they have already liked (content-based filtering), and properties that users who have similar preferences to them have liked (collaborative filtering). Lastly, due to the rapid increase in the use of algorithms in online systems, users are becoming more likely to trust that they will work. However, making the system as accurate and responsive to the user's preferences as possible will help to ensure that its users can quickly see that it works as a useful tool to help them in their search for the perfect property.

The types of recommendation that can be used and their limitations must also be considered. Initially, the system will use a content-based filtering approach to provide recommendations which works by recommending similar items to those that the user has interacted with positively in the past (Throat et al., 2015). In this case, this would be recommending properties which have similar attributes such as price, bedrooms, size, and location to those that the user has previously liked. The benefits of this approach are that recommendations are user independent, giving the user more control over the content that they will be shown and making it simple for a user to understand why they are being shown this content. Additionally, they work well for new systems which do not have any existing data about interactions as they can recommend items that have not yet been rated. The main drawback, however, is that recommendations can converge towards specific attributes. As a result, users may lack the ability to be shown different types of content that they may have been interested in due to the problem of over-specialization (Mohamed et al., 2019).

Collaborative filtering is an alternative approach which can be used to provide recommendations without requiring any information about what is being recommended. Instead, they work by analysing multiple users' behaviours making recommendations based off similarities that are identified between user's interactions. Alongside the benefit of not requiring details of the items to recommend, this also has the advantage that it can provide users with recommendations that may have been outside of their initial preferences that they end up liking. The main issue which collaborative filtering systems face is the cold-start problem, which is how they can recommend new items before they have had any interactions (Mohamed et al., 2019). For this reason, the system will start by only using a simple content-based filtering approach. However, after implementing a content-based recommendation system, if interaction data is gathered within the app, then this could later be used to implement a hybrid system. This would combine both content-based and collaborative filtering techniques in order to provide better recommendations and to reduce the impact of each of their drawbacks.

4.1.2 – Content Based Design

The content-based filtering system (CBF) is designed as an initial way for the system to recommend properties to users based on those that they have previously explicitly liked within the application.

Initially, all of the listings within the database are retrieved and loaded into a Pandas dataframe, as CBF is a memory-based approach, so all of the available data is required to be able to make recommendations. The data is then required to pre-processed before calculating any similarity values. The following features are considered: **Price, Bathrooms, Bedrooms, Size, Latitude, Longitude, Property Type**

As “Property Type” is a categorical feature, a one-hot encoding technique is used so that this can be represented as a numerical value alongside the other features, which is carried out by the Pandas *get_dummies()* function. After this has been completed and the dataframe only contains numerical values, the values are then normalised so that they all fit within the range of 0-1.

From here, a similarity dictionary is then created by iterating through each listing and using the SKL *cosine_similarity()* function which returns a value between 0 and 1 to indicate how similar the features of one property are to another. The dictionary produced contains the similarity for every property listing in the database and is referenced by the recommender algorithm when making predictions. To improve the speed of the recommender and save resources, this dictionary is then serialised and dumped to .pkl file using the Pickle library. For consequent requests to the recommender, the saved dictionary is loaded instead of recalculating all of the similarity values. As mentioned in section 4.4, more properties will not be added to the database once it has been initially populated, so for the purpose of this project the similarity dictionary will not need to be recalculated. However, if the system needed to be able to include newly added properties, then it could easily be modified to recalculate the similarity matrix in the background on a daily basis.

To provide users with actual recommendations, the CBF recommendation system takes the user's 5 most recently liked properties, the user's previously viewed properties, and the user's search filters that they have set within the app as inputs. It then filters the properties within the database according to the user's explicit preferences to ensure that only relevant properties are recommended, before removing any that the user has already viewed. The recommender then iterates through the remaining property listings and calculates the mean of the cosine similarity between each property and the 5 properties that the user has recently liked, as outlined in Figure 8. The 5 properties with the highest mean similarity score are returned as recommendations for the user.

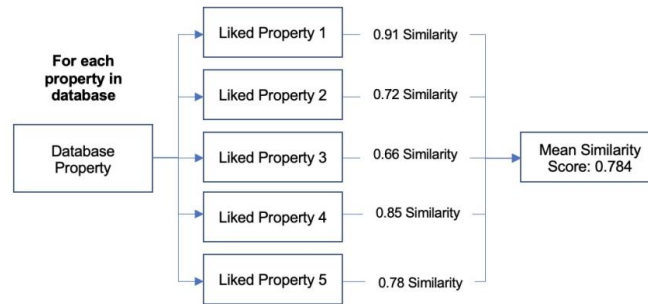


Figure 8

As the recommendations are presented using a card stack, and browsed through by using a swipe gesture (David et al., 2016), the CBF recommender also includes some extra properties that fit the user's general search criteria without considering their similarity to those that the user has recently liked. This means that if the users are not interested in the properties that are similar to their current ones that they have recently liked, they have the opportunity to also view other ones. If this was not the case, the recommendations would eventually converge to a very specific feature set and the user would be unable to direct the recommender away from this when browsing within this screen.

4.1.3 – Collaborative Filtering Design

Depending on the method of implementation used, recommender systems can typically be classified as either memory-based or model-based. Memory-based methods make recommendations by accessing the database directly, while model-based method uses the transaction data to create a model that can generate recommendation (Bobadilla et al., 2013). By directly accessing the database, a memory-based method is adaptive to data changes but requires large computational time according to the data size. Alternatively, for the model-based method, it has a constant computing time regardless the size of the data but not adaptive to data changes, making cold start a greater problem.

As the system has a memory-based content filtering approach, this can be utilised to help reduce the cold problem that a model-based approach would typically face. For this reason, collaborative-filtering has been implemented using a SVD based (Singular Vector Decomposition) model technique, which was heavily used during the Netflix Prize competition by the winning BellKor team (Koren et al., 2009). When providing recommendations for new users that the model does not have any data for, the system can either use the content-based approach or provide them with similar recommendations to those that it would provide to a user whose profile has similar characteristics to theirs. As mentioned in section 3.1.2, the system will capture implicit data from user interactions within the application, which can be used as the rating data for the CF recommendation system. The ratings will be calculated as per the criteria in Figure 9.

Rating	Condition
1	User has viewed a property listing and exited within 10 seconds
2	User has viewed a property listing for longer than 10 seconds
3	User has viewed a property listing for longer than 20 seconds, and scrolled to bottom/viewed additional details, or viewed the property twice
4	User has viewed a property for longer than 30 seconds, looked through more than 2 pictures and scrolled to bottom/viewed additional details, or explicitly liked the property from the discover (swipe) view
5	User has explicitly liked a property from the listing view

Figure 9

Within the iOS application, after viewing a property listing, a rating will be calculated and then an interaction will be recorded. To reduce the number of requests made to the backend, these are stored locally and then periodically sent to the backend to be stored within the “Events” collection in the database.

To develop and test the CF model, the system will require a set of existing interactions data. In a real-world situation, this could be gathered by first launching the system with just the content-based recommender and collecting real user interaction data. However, for the purpose of this project, this is infeasible so instead an artificial set of ratings data has been created.

To do this, 5 user personas have been generated as shown in Figure 10 and a set of ratings have been created to fit these. The system can later be tested to see if it provides recommendations that fit these.

Persona	Preferences
1 - Young professional moving to London for new job in city, looking for a property to share with 2 others. £37,000 per year income each.	Property Type: Flat Location: Clapham Bedrooms: 3 Price: ~ £3,000 pcm
2 - 35-year-old working in design industry, sharing with one partner. £130,000 per year combined income.	Property Type: Apartment/Flat Balcony if possible Bedrooms: 1 or 2 Location: Peckham Price: ~ £4,300 pcm
3 - Second-year Imperial College university student, sharing in house of 5 students at other universities. High income parents to support on top of student loan.	Property Type: House Bedrooms: 5 Price: ~ £5,500 pcm Furnished: Yes

4 - Low-income parent, living with partner and two children. Currently living in outer London but struggling with cost-of-living crisis.	Property Type: House Price: As low as possible Bedrooms: 4+ Price: ~ £2,500 pcm
5 - 25-year-old founder of real estate application, living with partner. Works remotely so location does not matter.	Property Type: House Price: £10,000+ pcm Swimming Pool if possible

Figure 10

The SVD++ algorithm has been chosen for the recommendation system model, which builds on the basic SVD approach to matrix factorisation by considering implicit ratings (Koren et al., 2009). In a traditional collaborative filtering system using SVD, the user-item rating matrix is first decomposed into three matrices: U , S , and V . U represents the users, S represents the strengths of the user-item interactions, and V represents the items. Once the matrix is decomposed, the system can use the matrices to predict missing ratings. Specifically, to predict the rating for a user u and item i , the system takes the dot product of the corresponding rows in U and V to get a predicted rating. This predicted rating can then be compared to the actual rating to calculate the error, which can be used to refine the matrices and improve the accuracy of the predictions.

As the SurPRISE library provides a ready to use implementation of the SVD++ algorithm so the implementation within the code was relatively straightforward. The ratings data is first retrieved from the system's database and loaded into a Pandas dataframe, and then loaded into SurPRISE with the `load_from_df()` function. The SVD++ model is then defined and trained using the `fit()` method that takes the training dataset as input and trains the model using stochastic gradient descent (SGD) optimization. During training, the model updates the user and item latent factors, as well as the user and item biases, based on the observed ratings and implicit feedback.

After training the model, the `predict()` method is then used to predict how the user will rate each item that they have not yet seen. The items with the highest predicted rating are then fetched from the database and displayed to the user.

Similarly to the content-based filtering recommendation system, this has been implemented as a Python script that is executed by the backend, with the results being sent to the application through a HTTP API request.

4.2 – Backend

4.2.1 - Overview

In order to provide the iOS application with data and to handle any requests that are made, the application communicates with a custom backend which is built using the Express.JS framework for the reasons outlined in Section 3.2.2. The main functionality is shown in Figure 11.

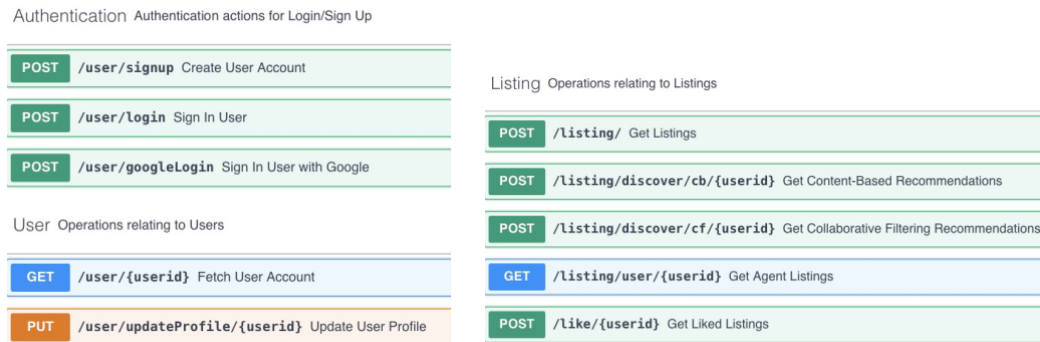


Figure 11

A complete overview of the backend functionality, including full details of all endpoints, can be found within the API Documentation which has been written following the industry standard OpenAPI 3.0 Specification so that it can easily be referenced when further developing the system. The requirements and design for all endpoints are also explained within the relevant user story cards that were documented for each feature.

4.2.2 – Database

The backend uses the Mongoose library to interface with the MongoDB database, allowing it to asynchronously access, create and modify documents using advanced queries.

The database schema is as follows:

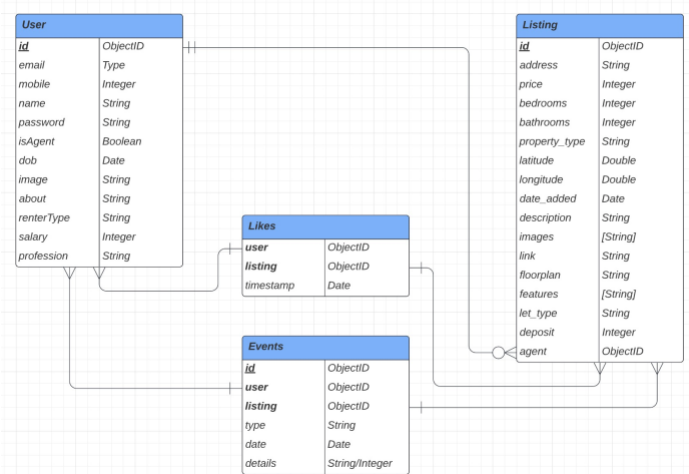


Figure 12

4.2.3 – Authentication

In order for users to use the mobile app, they must first create an account or sign in. Users are given the choice to either create a completely new account, or sign in with their Google account, using the OAuth2 standard.

If a user opts to create an account, they will be taken through the standard sign up flow where they must provide a name, email, and password, as well as being given the option to provide other optional data about themselves as outlined in the user schema. Upon completion, the data will be sent to the backend to be verified. The backend will first check that no account already exists for the given email address in the database, and then verify that all of the required fields for creating an account have been sent by the user. If this is the case, then a new user object will be created from the given data and stored in the database. Leaving plaintext passwords in a database puts both users and the system at risk so in order to ensure that user data is kept secure, the passwords are hashed before being stored in the database. To do this, the backend uses Bcrypt, which is the current industry standard function for password hashing, designed by Niels Provos and David Mazières and based on the Blowfish cipher (Sriramya et al., 2015). Bcrypt works by first creating a random salt, and adding this to the end of the password before hashing to ensure that each hash gives a different output, preventing our system from being vulnerable to a rainbow table attack (Boonkrong et al., 2015). When signing in users, the system uses the `bcrypt.compare` function to verify whether the hash stored in the database has originated from the unencrypted version of the password provided by the user.

Alternatively, if a user chooses to sign in through Google, the process is slightly different. The application first verifies the user's identity by calling `GIDSignIn's signIn(presentingViewController:completion:)` method which takes the user to a standard login screen. If the login is successful, Google then returns a user token through the function's completion handler which contains information about the user and allows the system to verify the authenticity of the request. This token is sent to the backend using a secure HTTPS POST request and verified again using the Google Auth Library for Node.js. From here, the backend can check whether an account for the user already exists in the system's database, and either retrieve the user's details or create a new user record using the information contained in the token's payload. Once a user's identity has been verified, the backend generates a JSON Web Token from the user's ID and signs it with a secret key so that it can later verify that this was generated by the system and not a third party. This is then returned to the user and stored on their device so that it can be passed to any future API calls, allowing the backend to authenticate that they have come from a signed-in user.

4.3 – Application / Frontend

4.3.1 – Development Process

As outlined in section 2.1, before starting development, the requirements for the system were broken down into a backlog of individual user stories. This gave a clear outline of the functionality and screens that would be needed. As each user story was elaborated, the screens were first designed as wireframes within Figma to cover the outlined functionality, and then refined into high fidelity UI mock-ups which showed exactly how the screen would look.

The functionality was then developed using the MVVM architecture, which organises the design of the application around 3 roles – model, view, and view-model. A model is a representation of the data objects such as a User or Property Listing. Views are responsible for displaying the data within the app and make up the UI. View models are used to facilitate communication between the views and models. This meant that the initial interface for each screen was built into a view in line with the design that had been created, and the relevant view models were created to hold any required data and to accommodate the required functionality.

The user story cards which clearly document each piece of functionality that has been documented are provided as a deliverable and should be referenced for a complete breakdown of what has been developed within the application. An example user story card as well as the full backlog of user stories is provided in the appendix.

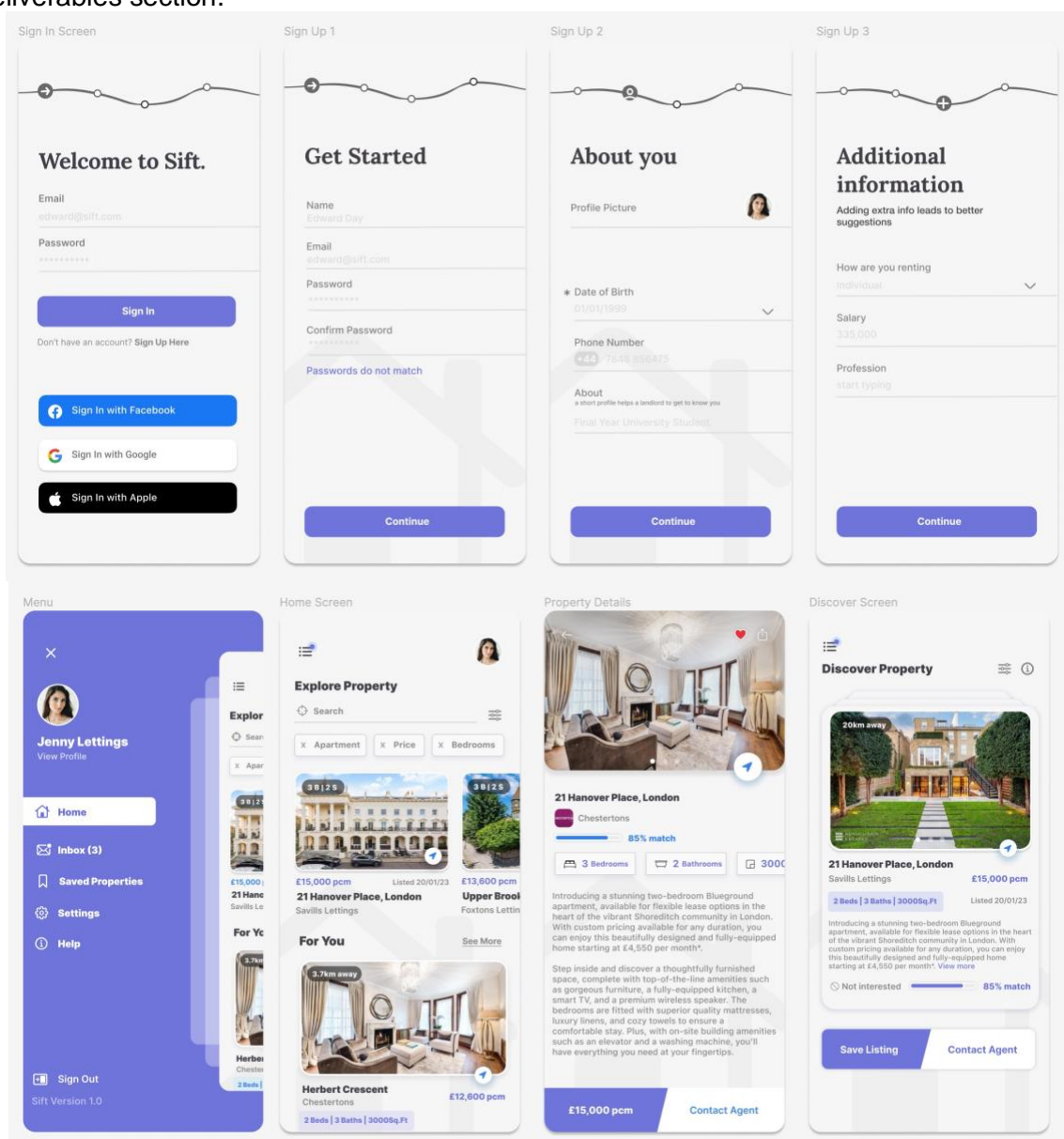
4.3.2 – UI/UX Design

As mentioned earlier, the interface was designed using Figma to create an initial wireframe followed by a high-fidelity prototype of how each screen and any animations that it features would look before being developed. This allowed development time to be reduced as it is far quicker to build an interface through a design tool than through code, and this meant that the code for each screen did not need to be repeatedly reworked.

As the recommendation system is the piece of functionality that makes this system unique from its competitors, a large amount of time was invested into designing how the user would view and navigate through the properties that are suggested to them. The final design was built around a Tinder-like card stack (David et al., 2016) which is used to present the recommended properties to the user in a visually appealing and engaging way. This involves displaying a series of cards stacked on top of one another, with the topmost card being the most visible. Each card provides details of a property that the system has provided as a recommendation to the user, and they can then swipe left or right on each card to indicate

whether they are interested in that property or not. The app then uses this information to help with making future suggestions to the user. An advantage of this style of interface is that it allows users to consume a large amount of content quickly and easily. The visual and interactive nature of the interface can also make the content more engaging and memorable for users and as such has been repeatedly proven to increase user satisfaction with applications. Additionally, the swipe-based interactions are intuitive and familiar to many users, making the interface easy to use and navigate.

The design for each screen is shown in Figure 13 and a full overview of the interface of the system with all animations can be found in the demonstration video provided in the deliverables section.



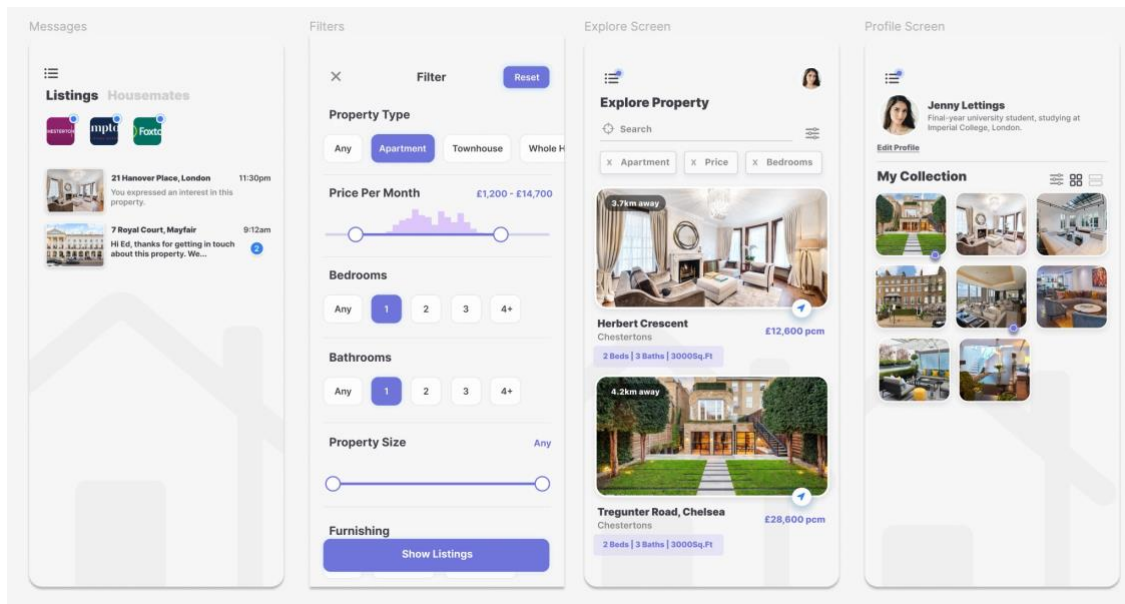


Figure 13

4.4 – Property Listing Scraper

As outlined in section 3.1.1, the system requires a collection of property listings which are to be stored within the database and presented to its users. The data for these is scraped from the Rightmove website with a Python-based scraper which uses the library BeautifulSoup to parse the HTML documents.

The scraper works by first going to the Rightmove search page with the Sort mode set to Most Recent and retrieving the links to the pages for each listing shown. The scraper will continue to iterate through each page of results until the desired number of links have been retrieved, or until a link is found which already exists within the current database of results as this means that any listings past this point have already been scraped. After the links have been retrieved, the scraper then retrieves the HTML document for each individual listing and uses corresponding element for each piece of data such as the address, description, bedrooms etc to populate a dictionary with all of the details of the property. As each property is linked to an estate agent, it searches for the account within the database (or creates one if necessary) and stores the agent's ObjectID against the property. After each of the fields for the property have been populated, it is then verified to ensure that there are no errors and that all of the required information could be found, before being saved to the database to be used within the app.

For the purpose of this project, it was decided to populate the database with an initial set of around 2000 properties located within London, instead of running the scraper continuously. The reason for this was so that the results could be tested more clearly, as having a constantly changing set of data that was too large would make it harder to demonstrate and replicate the recommendations that are produced by the system. The potential for how this could be further developed or used going forward is outlined in section 7.1.

Chapter 5 – Results and Evaluation

5.1 - Main Application Functionality

The final application was tested against the acceptance criteria (see appendix) set out for each requirement to ensure that it is fully functional. After the system had passed all of these, the main functionality of the application outside of the recommendation system was compared to the Rightmove iOS application using A/B testing to ensure that the developed system was optimised. This testing involved various participants who had never used either application before. They were then given a set of tasks to complete within each application which cover some of the main functionality and the time taken for both systems was measured. An overview of the results is detailed in Figure 14.

Task	Average Time (s) – Sift	Average Time (s) - Rightmove	Sift/Rightmove Time Factor
Create Account	48	44	1.09
Sign Out	5	5	1
Sign In	15	10	1.5
Find Property – Criteria A	20	47	0.43
Like Property – Criteria B	31	28	1.11
Find Property – Criteria C	18	18	1
Retrieve Liked Property	12	8	1.5
Overall	149	159	0.94

Figure 14

After completing the tasks, participants then took part in a survey where they were asked to give feedback about different aspects of the system. The full survey and results for each participant are covered in the appendix (section 7.2.1) and an overview of some of the responses that participants gave after using the system is detailed in Figure 15:

Question	Average Score (0-10)
How easy was the application to use?	8.7
How would you rate your experience using the application?	9
Do you think a recommendation system would be beneficial to potential renters?	9.3
Did you like the way the application looks?	9.3
How likely would you be to use this application over Rightmove?	7.3

Figure 15

By looking at the overview detailed above, as well as the full responses of the survey found in the appendix, it can be seen that participants were generally happy with their experience

of using the application and gave positive feedback about how easy it was to use and the way that it looked. The time taken for users to use the main functionality of the application was very comparable to when they used Rightmove, with most tasks taking around the same amount of time, and the overall time taken using Sift actually being 6% less. This was a very satisfying result, as Rightmove is the leading platform within the UK property market, and the main goal of this project was not to outperform their system. What makes Sift unique is the recommendation system section of the application, and as long as users were satisfied enough with the rest of the app then the recommendation system would be the reason for users to choose it over an alternative. A key observation that influenced the time taken was the fact that Rightmove offered slightly more options within some screens which resulted in users taking longer to find what they wanted. However, it was also noted that Rightmove's app was slightly more polished and had features such as autofill which made some screens slightly smoother and easier to use. The evaluation of the recommendation system is detailed in the next sections.

5.2 - Content-Based Filtering Recommender System

To evaluate the content-based filtering recommendations, a survey was carried out. To start, participants were presented with 5 sample users and the details of the properties that each had liked. The CBF recommendation system was then used to generate recommendations for each of these sample users, and the participants were asked to evaluate how suitable they thought each recommendation was for the user, and how similar they thought it was to the recently liked properties. The full survey and results are detailed within the appendix (section 7.2.2), and a summary of the results is provided in Figure 16.

	Average Similarity Rating	Average Estimated Satisfaction
Overall	87.2%	86.4%

Figure 16

These figures clearly indicate that the content-based recommendation system is very effective at making recommendations. This means that in the situation where a user is looking to find other properties which are similar to ones that they have already liked, then they are highly likely to be satisfied with the results. However, it should be considered that this does not necessarily reflect the overall effectiveness of the recommendation system, as users may not know what they are looking for or may prefer to be given different suggestions to help them find something new that they would have been unable to find themselves.

5.3 - Collaborative Filtering Recommender System

After making recommendations for each user persona using the content-based approach, the collaborative filtering system was then used for one user from each persona to produce

a set of recommendations that were deemed to be of interest to them. The top 3 properties that were recommended for each user are outlined in Figure 17.

Persona	Address	Beds	Price	Distance to Preferred Location	Property Type	Image
1	Creek Lane, Wandsworth	3	£3,500 pcm	2.5 miles	Flat	
1	Tachbrook Street, Pimlico	3	£3,000 pcm	2.7 miles	Flat	
1	Royal Oak Yard, London	3	£3,500 pcm	4.8 miles	Flat	
2	64 New Kent Road, London	2	£3,270 pcm	3.5 miles	Flat	
2	St. Katherine's Way, London	2	£3,792 pcm	3.8 miles	Flat	
2	Churchyard Row, London	2	£3,400 pcm	4.1 miles	Flat	
3	Ashbourne Road, Ealing	5	£4,600 pcm	No preference	House	
3	Barnfield Place, London	5	£4,650 pcm	No preference	Town House	
3	Hartington Road, West Ealing	5	£4,498 pcm	No preference	Semi-detached House	
4	Avenue Road, Southgate	4	£2,250 pcm	No preference	Maisonette	


4	Glenwood Road, South Tottenham	4	£2,500 pcm	No preference	Terraced House	
4	Braxted Park, Streatham	4	£2,800 pcm	No preference	Semi-detached house	
5	Frognaal, Hampstead	7	£41,167 pcm	No preference	Detached House	
5	The Bishops Avenue, Hampstead Garden Suburb	5	£26,000 pcm	No preference	Detached House	
5	Herbert Crescent, Knightsbridge	7	£67,167 pcm	No preference	House	

Figure 17

To evaluate how appropriate these recommendations are for the corresponding user, a survey was carried out. Participants were first made familiar with each of the 5 user personas before being presented with the full details for each of the 15 properties listed above in a random order. They were then asked to state which user they thought the property would be most appropriate for, as well as how happy they thought the chosen user would be with the suggestion on a scale of 1-5. The full survey and results are detailed within the appendix (section 7.2.3).

In the survey, 97% of recommendations were matched to their corresponding persona, and participants estimated an 83% satisfaction rate for the users. This demonstrates that out of the properties available, the recommender is consistently selecting the ones for each user that are most appropriate for them. The 83% satisfaction rate is also a positive indication about the effectiveness of the recommender, although it should be remembered that this is an estimate given by participants based on how satisfied they think that someone who fits the persona would be, rather than by someone who actually fits the outlined criteria. Ideally, the recommender would have enough users and data to be able to make recommendations for any type of user, and this survey would have been carried out with participants who are actively searching for a new property.

Chapter 6 – Conclusion

6.1 – Legal/Ethics

6.1.1 – Rightmove Property Data

Web scraping, also known as data scraping, is a technique that involves the automated extraction of data from websites. It is becoming increasingly popular among businesses and researchers as a way to gather data for various purposes. However, web scraping can raise legal concerns, particularly when it comes to accessing data from websites without authorization.

According to the terms of service of Rightmove, web scraping is allowed for personal use. Rightmove's terms of service specify that "Rightmove grants you a limited, revocable, non-transferable, non-exclusive license to access and make personal use of all data provided by the Website." This means that individuals are permitted to use web scraping tools to extract data from Rightmove for their personal use, as long as they comply with the website's terms of service and applicable laws. However, it is important to note that Rightmove's terms of service may change over time, and the current terms of service should be regularly reviewed.

Additionally, even though Rightmove may allow web scraping for personal use, it is still important to be mindful of legal and ethical considerations, such as copyright infringement, data protection, and data privacy. Web scraping should only be conducted in compliance with applicable laws and ethical standards, and proper consent would be needed before sharing any scraped data with third parties or using it for commercial purposes. As a result, the method used for gathering the data within this project would not be suitable if this application was being used in the real world.

6.1.2 – User Data (Data Protection Act)

As the system collects and processes user data, if it was to be used in the real world then it would need to comply with GDPR. The General Data Protection Regulation came into effect in 2018 and controls how businesses or organizations use their users' personal data.

The following considerations have been made to ensure that the system complies with GDPR, although if it was to be used in the real world, then this is something that would require further thought and professional advice:

Obtain Consent – Users must give clear and informed consent before the system can collect or process any personal information about them. In order to comply with this, users

are provided with a Terms and Conditions statement during sign-up that they must state that they agree to before they can create an account. This can be found within the appendix, and outlines exactly how the system will use their data and who it will be shared with.

Limiting Data Collection – As the application is based around a recommendation system, both implicit and explicit data collection is essential for users to receive the most accurate recommendations. However, all of the data which is collected by the application was carefully considered, and there is justification for all data that is collected, with the main reason being around giving the user the best possible experience.

Security Measures – The system stores personal information about users so it is important to consider how data can be kept secure to prevent any unauthorised access. As the system is not being used in the real world, this was not a key priority for this project, so user accounts are simply protected by a hashed password. The database is managed by MongoDB Atlas and protected with 2-Factor Authentication, so for the purpose of this project can be considered secure. As mentioned in section 4.2.3, the backend uses JWT to verify the authenticity of requests which prevents unauthorised access to information through API calls. However, if this system was being used for production, the API requests could be made using HTTPS to secure the data being transferred, and user account information could be encrypted to further secure the database.

Data Subject Rights – As per GDPR regulations, it is important to ensure that users have the right to access, erase, rectify, and restrict the processing of their data. This has been considered, and for the purpose of this small-scale project, this can be done manually by a system admin if requested by a user. However, if this project had a larger userbase, this could be implemented as a process which can be carried out instantly and automatically.

6.2 – Finished System Overview

In summary, the following has been developed for the finished system :

- **User Authentication** – Users can sign up for an account which they can use to verify their identity to the system and store and access their personal data from anywhere. The system also has a Sign In with Google option which uses OAuth2.
- **Property Listings** – Users can browse through a variety of property listings within the application. The data for these has been retrieved using a web scraper which was created for the purpose of this project. The details of each property listed are presented to users using a visually appealing user interface.

- Filter Properties – To help find relevant properties, users can apply search filters within the application which ensures that they are only shown properties that are of relevance to them.
- Recommendation System – The application has both a content-based and collaborative filtering recommendation system which have been developed to provide users with properties which are deemed to be of interest to them
- User Interactions – Users can interact with properties explicitly to “like” them and save them to return to later. When viewing properties, the system also calculates an implicit rating as an estimate for how interested users are
- Card Stack – Alongside the standard method of viewing returned properties in a list, users can browse using an intuitive animated card stack interface where they use swipe gestures to explore the results and state their preference about the property
- Background Recommendations – The system works in the background to search for suitable properties for the user even when they are not, and if a suitable property is found then users are sent a push notification to notify them.
- Async/Await – The application makes requests to the backend in the background using Swift Async and Await methods to ensure that the application is as smooth as possible to use.
- Backend – All data sent to and from the mobile application is processed by a Node.JS server which is accessible through a REST API. This is deployed to a DigitalOcean droplet, so the application is functional from anywhere.
- User Interface – A large amount of time was put into the design of the UI for the application. It was first designed in Figma and went through multiple refinements to ensure that it was as aesthetically pleasing as possible. The final application consists of 20 screens.

6.3 – Further Developments

6.2.1 – Property Listing Data

One aspect of the application which could have been developed further was the property listing scraper. As discussed in section 4.4, in order to allow the system to be better evaluated, the database was initially populated with a set of property listings which were scraped from Rightmove. These were not updated and no more were added throughout the duration of the project. The scraper was designed to ignore property listings which already exist within the database, which means that it could have been configured to continuously retrieve the latest properties. However, as properties which already exist in the database are ignored by the scraper, this would mean that properties which are updated or removed on Rightmove are not updated within database for our application. The scraper could easily be modified to periodically update the current listings within the database, however, as the

number of listings grows this would take an increasingly large amount of time. Also, as mentioned in section 6.1.1, a property listing scraper would not comply with Rightmove's terms of service if it was being used for commercial use. As a result, if this project was taken further, then it would require estate agents to partner and share their property data directly in return for the advertising that Sift would help provide for them. If this approach was used, then it would ideally require a content management system.

This has been considered and would be developed as a platform called "Sift Agent". Sift Agent would function as a web-app, providing an easily accessible dashboard where estate agents can manage and update their listings, view detailed analytics, and easily contact prospective tenants. It could also feature automation and AI functionality to help improve the quality of listings and minimise the work required by the estate agent.

6.2.2 – Application Functionality

All of the functionality which was initially planned for this project has been developed. However, the following ideas have been set out for functionality which could be added if it was further developed:

- Allow users to browse as a group and have recommendations made based on their combined preferences
- Help users find potential housemates/people to share with using a recommendation system
- Add in-app messaging between groups of users or between users and estate agents
- Simplify the process of acquiring a property by allowing users to verify their rental information (guarantors, income, identification, etc.) within the app which could then be passed on to estate agents
- Add more preferences within the app for users to help refine their search

6.2.3 – Recommendation System

As concluded in the results section, the recommendations provided within the application generally appear to be relevant to users. However, there is still a large room for improvement.

For the collaborative filtering system, the results produced greatly rely on the amount of data which is available, so it would be expected that the recommendations would improve as the app gets more users. Once there is more data, more time could also be put into optimising the model which is used, and there could also be potential to use a more advanced machine learning algorithm to produce the recommendations. There could also be more consideration put into the factors which influence recommendations, and a hybrid recommendation system could be implemented with the view of increasing user satisfaction.

6.4 – Self Appraisal

Overall, I am extremely satisfied with the final result of my project and the application which has been developed. When I initially had the idea for the project, I was conscious of how large the scope was and the fact that the work required for developing a full-stack application with so many components should not be underestimated. As I was also using such a wide range of unfamiliar technologies, I was also aware of the amount of learning that would be required so began working on it as early as possible in order to give myself the best chance of achieving my goals. I am extremely passionate about the idea for this project which has helped massively with motivation. As such, I have gone above and beyond with dedicating as much time and effort as possible to every aspect of the project to ensure that everything has been completed to the highest standard possible, and I believe that this shows through in quality the finished product.

I feel that the project has been managed extremely effectively, with everything being delivered in-line with the original schedule that was set out, and all of the original requirements being met. The agile workflow has helped massively as when initially starting development it was such a daunting task, and I didn't know where to start with building a system with so many different components. However, the process of working through each user story card worked perfectly. By trusting the process, each of the components ended up tying together and integrating with each other without facing any issues which was one of the initial concerns when beginning the project.

In terms of the functionality of the application I am also very satisfied. As discussed in the results section, the base functionality of the application without considering the recommendation system is comparable to the leading property apps which are currently available, which will have had a full team working on them and an extensive amount of funding. As a result, I feel that it is an amazing achievement to have been able to produce this as an individual for a university project alongside the rest of my studies and shows the potential for how far this project could be taken if I had more time and resources.

The greatest challenge of the project was definitely building and testing the recommendation system without having real world user data, although from the results that have been discussed I feel that what has been developed is still a very functional system and clearly demonstrates that the system is able to make relevant and useful recommendations to the user.

I have greatly enjoyed the time that I have dedicated to working on this project and I have gained a wealth of experience which will be carried with me for all future endeavours. I am excited to continue working on this project beyond my academic pursuits and see its continued development.

Chapter 7 – Appendix

7.1 – References

- [1] Office for National Statistics (ONS), released 5 January 2023, ONS website, statistical bulletin, Housing, England and Wales: Census 2021
- [2] Rightmove (2022) The Rightmove rental trend tracker, Q3 2022 [Online] Date accessed: 27th February 2023 <https://hub.rightmove.co.uk/content/uploads/2022/10/Rightmove-Rental-Trends-Tracker-Q3-2022.pdf>
- [3] Van Ham, M., 2012. Housing behaviour. The SAGE handbook of housing studies, pp.47-65.
- [4] Earnhart, D., 2002. Combining revealed and stated data to examine housing decisions using discrete choice analysis. *Journal of Urban Economics*, 51(1), pp.143-169.
- [5] Deldjoo, Y., Schedl, M., Cremonesi, P. and Pasi, G., 2020. Recommender systems leveraging multimedia content. *ACM Computing Surveys (CSUR)*, 53(5), pp.1-38.
- [6] Jeong, Y.J., Lee, J.H. and Shin, G.S., 2008, February. Development process of mobile application SW based on agile methodology. In *2008 10th International Conference on Advanced Communication Technology* (Vol. 1, pp. 362-366). IEEE.
- [7] Wiertel, P. and Skublewska-Paszkowska, M., 2021. Comparative analysis of UIKit and SwiftUI frameworks in iOS system. *Journal of Computer Sciences Institute*, 20, pp.170-174.
- [8] Gilchrist, E., 2021. SwiftUI vs UIKit: A Case Study on How a Declarative Framework Can Improve Learnability of UI Programming.
- [9] Statista (2022). Most used web frameworks among developers worldwide, as of 2022 [Online] Date Accessed: 28th February 2023
- [10] Muittari, J., 2020. Modern web back-end.
- [11] Khan, S. and Mane, V., 2013. SQL support over MongoDB using metadata. *International Journal of Scientific and Research Publications*, 3(10), pp.1-5.
- [12] Moschetti, B (2015). MongoDB vs SQL: Day 14 – Queries [Online] Date Accessed: 17th March 2023
- [13] Elliot, J., Mooney, P (2021). '2021 Kaggle Machine Learning & Data Science Survey'. Kaggle. [Online] Date Accessed: 17th March 2023 <https://kaggle.com/competitions/kaggle-survey-2021>.
- [14] Richter, F. (2022). Amazon, Microsoft & Google Dominate Cloud Market [Online] Date Accessed: 21st March 2023 <https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/>
- [15] Zhang, M. and Liu, Y., 2021. A commentary of TikTok recommendation algorithms in MIT Technology Review 2021. *Fundamental Research*, 1(6), pp.846-847.
- [16] Hwangbo, H., Kim, Y.S. and Cha, K.J., 2018. Recommendation system development for fashion retail e-commerce. *Electronic Commerce Research and Applications*, 28, pp.94-101.
- [17] DeVito, M.A., Gergle, D. and Birnholtz, J., 2017, May. " Algorithms ruin everything" # RIPTwitter, Folk Theories, and Resistance to Algorithmic Change in Social Media. In *Proceedings of the 2017 CHI conference on human factors in computing systems* (pp. 3163-3174).
- [18] Bozdag, E., 2013. Bias in algorithmic filtering and personalization. *Ethics and information technology*, 15, pp.209-227.

- [19] Hoadley, C.M., Xu, H., Lee, J.J. and Rosson, M.B., 2010. Privacy as information access and illusory control: The case of the Facebook News Feed privacy outcry. *Electronic commerce research and applications*, 9(1), pp.50-60.
- [20] Amy Kraft, 2016. Backlash Continues over Instagram's New Algorithm. [Online] Date Accessed: 27th February 2023
- [21] Thorat, P.B., Goudar, R.M. and Barve, S., 2015. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4), pp.31-36.
- [22] Beck, S.J., 1951. Review of Human behavior and the principle of least effort: An introduction to human ecology.
- [23] Herbig, P.A. and Kramer, H., 1994. The effect of information overload on the innovation choice process: Innovation overload. *Journal of Consumer Marketing*.
- [24] Shahabi, C. and Chen, Y.S., 2003. Web information personalization: Challenges and approaches. *Databases in Networked Information Systems: Third International Workshop, DNIS 2003, Aizu, Japan, September 22-24, 2003. Proceedings 13* 3, pp.5-15.
- [25] Liang, T.P., Lai, H.J. and Ku, Y.C., 2006. Personalized content recommendation and user satisfaction: Theoretical synthesis and empirical findings. *Journal of Management Information Systems*, 23(3), pp.45-70.
- [26] Mohamed, M.H., Khafagy, M.H. and Ibrahim, M.H., 2019, February. Recommender systems challenges and solutions survey. In *2019 international conference on innovative trends in computer engineering (ITCE)* (pp. 149-155). IEEE.
- [27] Effendy, F. and Adhilaksono, B., 2021. Performance Comparison of Web Backend and Database: A Case Study of Node. JS, Golang and MySQL, Mongo DB. *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)*, 14(6), pp.1955-1961.
- Node
- [28] Bobadilla, J., Ortega, F., Hernando, A. and Gutiérrez, A., 2013. Recommender systems survey. *Knowledge-based systems*, 46, pp.109-132.
- [29] Rebentrost, P., Steffens, A., Marvian, I. and Lloyd, S., 2018. Quantum singular-value decomposition of nonsparse low-rank matrices. *Physical review A*, 97(1), p.012327.
- [30] Koren, Y., Bell, R. and Volinsky, C., 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8), pp.30-37.
- [31] Statista, 2022. iOS app retention rates worldwide in 3rd quarter 2022 [Online] Date accessed: 4th March 2023 <https://www.statista.com/statistics/1248207/ios-app-retention-rate/>
- [32] Provos, N. and Mazieres, D., 1999. Bcrypt algorithm. In *USENIX*.
- [33] Sriramya, P. and Karthika, R.A., 2015. Providing password security by salted password hashing using bcrypt algorithm. *ARPN journal of engineering and applied sciences*, 10(13), pp.5551-5556.
- [34] Boonkrong, S. and Somboonpattanakit, C., 2016. Dynamic salt generation and placement for secure password storing. *IAENG International Journal of Computer Science*, 43(1), pp.27-36.
- [35] David, G. and Cambre, C., 2016. Screened intimacies: Tinder and the swipe logic. *Social media+ society*, 2(2)

7.2 - Evaluation Surveys/Results

7.2.1 - Main Application Functionality

As explained in section 5.1, the main functionality of the application was tested against that of Rightmove by measuring the time taken for participants to carry out various tasks within both applications. The participants were then asked a series of questions about their experience. The evaluation form and results are provided below:

Evaluation Exercise 1

Thank you for agreeing to take part in this evaluation for Sift.

As part of this evaluation, you will be provided with an iPhone which has both Sift and Rightmove mobile applications installed. You will be instructed which order to use the applications in. If you get stuck with any task, you can request to skip it and will be moved on to the next.

For each application, please carry out the following tasks:

1. Open the application and sign up for an account by providing an email and password. Provide as much non-mandatory information as you feel you would if you were using the application independently.
2. After signing up for an account, log out of your account.
3. Log back in using the account you created in step 1.
4. Find a property between **£3,000** and **£3,500 pcm** which has **2 bedrooms** and **2 bathrooms**.
5. Look at the first 2 properties you find between **£5,000** and **£7,000 pcm** and "Like" the one that you prefer.
6. Find a house with **5 bedrooms**.
7. The property that you "liked" in step 5 will be accessible from your profile. Go to your profile and retrieve the property that you liked.

Results

Task	Time (Sift)	Time (Rightmove)
Create Account		
Sign Out		
Sign In		
Find Property - Criteria A		
Like Property - Criteria B		
Find Property - Criteria C		
Retrieve Liked Property		

Feedback

Please answer the questions 1-5 about Sift on a scale of 0-10, with 0 being negative and 10 being positive.

1. How easy was the application to use?

2. How would you rate your experience using the application?

3. Do you think a recommendation system would be beneficial to potential renters?

4. Did you like the way the application looks?

5. How likely would you be to use this application over Rightmove?

6. Briefly explain how you thought the application compared to Rightmove and outline any key observations.

7. Additional comments (optional)

Results:

7.2.2 – Content Based Filtering Evaluation

As explained in section 5.2, the content-based filtering recommendations were evaluated by showing participants the properties which users had liked and then asking them to provide feedback about the recommendations generated for each user by the system. The evaluation form and results are provided below:

Evaluation Exercise 3

Thank you for agreeing to take part in this evaluation for Sift.

For this evaluation exercise, you will be 5 users and the properties that they have recently liked. You will then be shown a property that has been recommended for the user, based on the fact that it has been classed as similar to those that they have already liked.

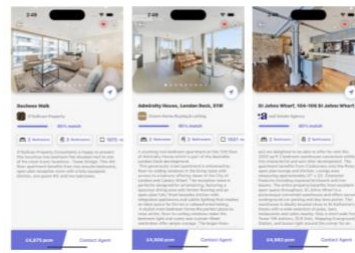
For each user, state how similar you think the recommendation is to their recently liked properties, and how satisfied you think the user would be with the recommendation, rating both between 1 and 5.

Results

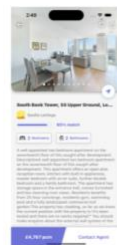
User	Recommendation Similarity (1-5)	Satisfaction with Recommendation (1-5)
1		
2		
3		
4		
5		

User 1

Recently Liked Properties:

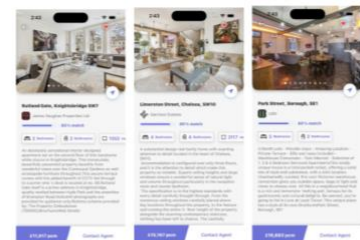


Recommended Property:

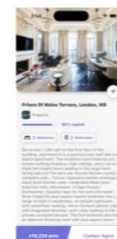


User 2

Recently Liked Properties:

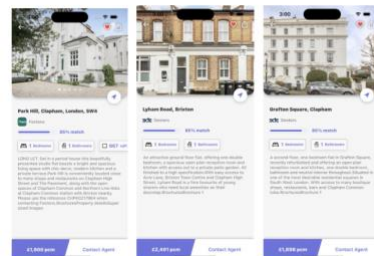


Recommended Property:



User 3

Recently Liked Properties:

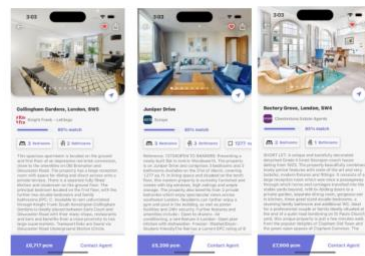


Recommended Property:



User 4

Recently Liked Properties:

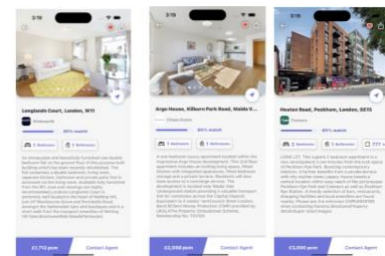


Recommended Property:



User 5

Recently Liked Properties:



Recommended Property:



Results:

Sample User	Average Similarity Rating	Average Estimated Satisfaction
1	100%	100%
2	80%	76%
3	68%	76%
4	96%	100%
5	92%	80%
Overall	87.2%	86.4%

7.2.3 – Collaborative Filtering Evaluation

As explained in section 5.3, the collaborative filtering recommendations were evaluated by getting participants to select which persona they thought each property recommendation was the best fit for. The evaluation form and results are provided below:

Evaluation Exercise 1

Thank you for agreeing to take part in this evaluation for Sift.

Before completing this evaluation, please make yourself familiar with the following 5 user personas.

Persona	Preferences
1 - Young professional moving to London for new job in city, looking for a property to share with 2 others. £37,000 per year income each.	Property Type: Flat Location: Clapham Bedrooms: 3 Price: ~ £3,000 pcm.
2 - 35-year-old working in design industry, sharing with one partner. £130,000 per year combined income.	Property Type: Apartment/Flat Balcony if possible Bedrooms: 1 or 2 Location: Peckham Price: ~ £4,300 pcm.
3 - Second-year Imperial College university student, sharing in house of 5 students at other universities. High income parents to support on top of student loan.	Property Type: House Bedrooms: 5 Price: ~ £5,500 pcm. Furnished: Yes
4 - Low-income parent, living with partner and two children. Currently living in outer London but struggling with cost-of-living crisis.	Property Type: House Price: As low as possible Bedrooms: 4+ Price: ~ £2,500 pcm.
5 - 25-year-old founder of real estate application, living with partner. Works remotely so location does not matter.	Property Type: House Price: £10,000+ pcm. Swimming Pool if possible

For this evaluation exercise, you will be presented with 15 properties. For each property, state which persona you feel that it is best suited to, and rate on a scale of 1-5 how satisfied you feel that the chosen user would be with the recommendation, based off the information provided about their preferences.

Results


Property	User	Satisfaction with Recommendation (1-5)
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Property 1

Price: £3,000 pcm

3 Bedrooms

Flat



Technick Street, London

85% match

3 bedrooms

2 bathrooms

£3,000 pcm


Contact Agent

Property 2

Price: £4,650 pcm

5 Bedrooms

Town House



Barnfield Place, London, E14

80% match

5 bedrooms

3 bathrooms

£4,650 pcm


Contact Agent

Property 3

Price: £69,117 pcm

5 Bedrooms

Terraced



Hurlford Crescent, Knightsbridge, SW1X

80% match

5 bedrooms

4 bathrooms

£69,117 pcm


Contact Agent

Property 7

Price: £2,800 pcm

4 Bedrooms

Semi-detached House



Brixton Park, Streatham

80% match

4 bedrooms

2 bathrooms

£2,800 pcm


Contact Agent

Property 8

Price: £4,498 pcm

5 Bedrooms

Semi-detached House



Hurlingham Road, West Ealing

80% match

5 bedrooms

2 bathrooms

£4,498 pcm


Contact Agent

Property 9

Price: £3,500 pcm

3 Bedrooms

Flat



Ladbroke Grove, London, W2

80% match

3 bedrooms

2 bathrooms

£3,500 pcm


Contact Agent

Property 4

Price: £2,550 pcm

4 Bedrooms

Maisonette



Avenue Road, Houghton, W14

80% match

4 bedrooms

2 bathrooms

£2,550 pcm


Contact Agent

Property 5

Price: £26,000 pcm

5 Bedrooms

Detached House



The Shilpesh Avenue, Houghton, W14

80% match

5 bedrooms

3 bathrooms

£26,000 pcm


Contact Agent

Property 6

Price: £3,792 pcm

2 Bedrooms

Flat



St. Katharine Way, London E1W

80% match

2 bedrooms

2 bathrooms

£3,792 pcm


Contact Agent

Property 10

Price: £3,270 pcm

2 Bedrooms

Flat



68 New Kent Road, London SE5

80% match

2 bedrooms

2 bathrooms

£3,270 pcm


Contact Agent

Property 11

Price: £41,167 pcm

7 Bedrooms

House



Princes Road, Houghton, W14

80% match

7 bedrooms

3 bathrooms

£41,167 pcm


Contact Agent

Property 12

Price: £3,600 pcm

2 Bedrooms

Flat



Churchyard Row, London, SE11

80% match

2 bedrooms

2 bathrooms

£3,600 pcm


Contact Agent

Property 13

Price: £3,500 pcm

3 Bedrooms

Flat



Creek Lane, Wandsworth

5 mins

80% match

3 bedrooms

2 bathrooms

A well presented, three double bedrooms, two bathrooms, big front garden offering views over Wandsworth Common. A large balcony with access to the garden. The property is in a prime location within the London Underground. The property is in a prime location within the London Underground. The property is in a prime location within the London Underground.

£3,500 pcm


Contact Agent

Property 14

Price: £4,600 pcm

5 Bedrooms

House



Ashbourne Road, Ealing

5 mins

80% match

5 bedrooms

2 bathrooms

Stunning property, 5 bedrooms, 2 bathrooms, large front garden, close to transport, close to transport, close to transport. The property is in a prime location within the London Underground. The property is in a prime location within the London Underground. The property is in a prime location within the London Underground.

£4,600 pcm


Contact Agent

Property 15

Price: £2,500 pcm

4 Bedrooms

Terraced House



Glasgow Road, South Tottenham, W15

5 mins

80% match

4 bedrooms

2 bathrooms

For rent is this recently refurbished three/ four bedroom, two bathroom Victorian terraced house located off St. John's Road in a prime location. The property is in a prime location within the London Underground. The property is in a prime location within the London Underground. The property is in a prime location within the London Underground.

£2,500 pcm

Contact Agent

Results:

Participant	Correct Answers	Average Satisfaction
1	15/15	91%
2	15/15	75%
3	15/15	88%
4	15/15	80%
5	15/15	80%
Average	97%	83%

7.3 - Example User Story Card

As explained, all features were fully elaborated using user story cards to document the full requirements and implementation details. An example USC is provided below for reference and the full collection can be found as a deliverable.

USC Number	USC Name	Feature
3	Sign Up - Email	1 - Authentication

User Story:

As a new user,
I want to be able to sign up for an account,
So that I can use the application AND my data can be remembered

Description:

Within the application, users must first create an account so that they can save their details and verify their identity. If they request to create an account, they will first be asked to input all of the required fields for creating an account, and then also given the option to add additional account information. This will then be sent to the backend, and if successful then an account will be created for them and they will automatically be signed in.

Frontend:

- User is first asked to input name, email, password and confirm password
- When user presses continue, fields will be verified to ensure the input is valid, and make HTTP request to /user/checkEmail route to ensure that no account exists already
- If everything is valid, they will be taken to next screen, or if not they will be given an appropriate warning message
- User is then given the chance to add Profile Picture, date of birth, phone number, about info.
- If user tries to sign up, fields are then verified again and application will make HTTP request to user/signup route, including all entered information in the request body.
- If response code is 200:
- Set LoginModel currentUser to the user returned in response
- Set LoginModel isAuthenticated to True
- Save UserId and JWT (see USC 5) to userDefaults
- If response code is not 200, then present error message to user

Backend:

/user/signup API route

- Decode request body to corresponding variables
- Check that user does not exist in database for provided email
- If user does exist, return 404 authentication error
- If user does not exist, hash provided password using Bcrypt and create a new User object with all of the provided details, setting the isAgent field to false
- Store user account in database, generate JWT (see USC 5) and return status 200, with user account and JWT in response body

/user/checkEmail API route

- Check if user account exists for user ID contained in request parameters
- Return true or false accordingly

Acceptance Criteria

GIVEN: There is an account registered with an email
WHEN: I enter the email in the sign up screen
THEN: I will not be able to proceed
AND: I will be notified that an account already exists

GIVEN: I have entered passwords that do not match within the sign up screen
WHEN: I try to proceed
THEN: I will be notified that the passwords do not match

Dependencies

5

Mockups:

API Endpoint:

POST /user/signup Create User Account

By sending the details of a new user, you can create a new user account in the database

Parameters: No parameters

Request body **required**

Example Value Schema:

```
{
  "email": "edward@eddy.com",
  "password": "password123456789"
}
```

7.4 – User Story Backlog

Features	Name
1	Authentication
2	Screens
3	Core Functionality

User Story Cards	Name	Feature
1	Sign In - Email	1
2	Sign In - Google	1
3	Sign Up - Email	1
4	Sign Up - Google	1
5	JWT Authentication	1
6	Sign Out	1
7	Home Page	2
8	Browse Properties	2
9	Discover Properties - Frontend	2
10	Property Listing Details	2
11	User Profile Page	2
12	Search Filters	3
13	Search Terms	3
14	Sidebar	2
15	Location Information	3
16	Pagination	3
17	Events Tracking	3
18	Like Properties	3
19	Share Property	3
20	Contact Estate Agent	3
21	Estate Agent Profile	2
22	Edit Profile	2

7.5 - Terms and Conditions

Welcome to Sift. The following terms and conditions govern your use of the application. By using Sift, you agree to these terms, which constitute a legally binding agreement between you and us. If you do not agree to these terms, please do not use Sift.

User Accounts

- a. You must create an account to use certain features of the app.*
- b. You are responsible for maintaining the confidentiality of your account login information and password, and for all activities that occur under your account.*
- c. You agree to provide accurate and complete information when creating your account and to update your information as necessary.*
- d. You may not transfer or share your account with anyone else.*
- e. We reserve the right to suspend or terminate your account at any time, for any reason, without notice.*

Data Collection

- a. We collect and use certain data from your use of Sift, including your property viewing history and feedback, to provide recommendations and influence other users' recommendations.*
- b. We may also collect certain personal information from you, such as your name and contact information, in connection with your account.*

Intellectual Property

- a. Sift and its content are protected by copyright, trademark, and other intellectual property laws.*
- b. You may not reproduce, modify, distribute, or create derivative works of Sift or its content without prior written consent.*

User Conduct

- a. You agree to use Sift only for lawful purposes and in accordance with these Terms.*
- b. You may not interfere with or disrupt the App or its servers or networks.*

Indemnification

- a. You agree to indemnify and hold us harmless from any claims, damages, or expenses arising from your use of the App or your violation of these Terms.*

Changes to Terms

- a. We reserve the right to change these Terms at any time, without notice.*
- b. Your continued use of the App after any changes to these Terms constitutes your acceptance of the revised Terms.*

Contact Us

- a. If you have any questions or concerns about these Terms, please contact us at edward57day@gmail.com*

7.6 – Participant Consent Form

Participant Information and Consent Form

Sift Evaluation Exercise

You are being invited to take part in an evaluation exercise to help assess the functionality and effectiveness of various aspects of the Sift mobile application. Before you decide whether to take part in the experiment, it is incredibly important to understand what the experiment will consist of, what is required of you and the purpose of the experiment. Please take your time to carefully read the following information.

Do I have to take part?

No, you have complete freedom over the choice. If you do not wish to take part in the experiment, please complete the Consent Form to show your rights in relation to the research. If you do decide to take part, you are still able to withdraw at any time without a reason. Please contact Edward Day via email at edward57day@gmail.com if you wish to withdraw from the evaluation at any point.

What will the evaluation consist of?

You will take part in one of three evaluation exercises. Each exercise should take no longer than 15 minutes.

1 – Evaluation of Mobile Application

This will involve carrying out various tasks within both the Sift and Rightmove mobile applications. The time taken to complete each task will be measured, and you will then be asked a series of questions afterwards about your experience. Please note that you will be required to provide personal information to both platforms in order to sign up for an account. This can be deleted afterwards and will not be viewed or shared.

2 – Evaluation of Content-Based Filtering Recommendations

This will involve analysing recommendations provided by the system for sample users and providing feedback about the effectiveness of the system.

3 – Evaluation of Collaborative Filtering Recommendations

This will start with participants being made familiar with a set of 5 user personas. They will then be shown a set of properties that have been recommended and asked which user they think the recommendation was for.

What are the benefits of the experiment?

There is no financial reward for taking part in the experiment.

Background

This experiment is part of the Final Year Project - Development of a real-estate mobile application with a custom recommendation system to help users find a suitable property – “Sift”. For more information on the exercise please contact Edward Day via email at edward57day@gmail.com

Signature Form

- I understand that I do not have to take part in the experiment.
- I understand that if I do decide to take part, I can withdraw at any time without giving a reason.
- I have read and understood the background and the purpose of the experiment and had all my queries regarding the experiment answered.
- I understand how the experiment will be conducted.
- I understand that I can withdraw my permission to use my data within 7 days after completing the experiment.
- I am aware of what data will be collected and I understand how the data will be used.
- I give consent for my data requested to be used and published.
- I understand that the information stored cannot be used to directly identify me and the any personal data will be anonymised.
- I understand that while the data is stored, I am entitled to access and alter any information relating to myself.
- I understand that if I have any further queries or require any clarification, I can contact any of the experiment leaders and I have been provided with their contact details.

Signature

Date.....

7.7 – Other Figures and Tables

Figure 1

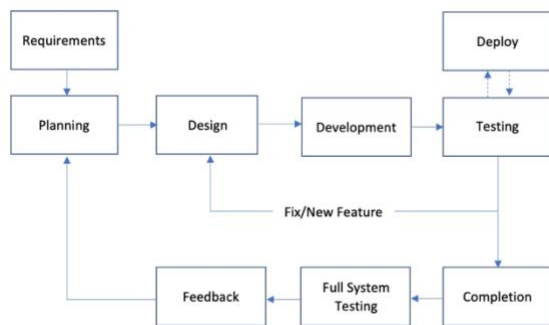


Figure 2

Risk	Likelihood	Impact	Actions Taken to Minimize Risk
Would not be able to complete project due to scope being too large	High	High	Focus on MVP and prioritise most important functionality, plan full timeline
Recommendation system would not provide suitable results	Moderate	Moderate	Start with a simple system and add complexity later
Unable to get data to use for system	Moderate	High	Put this as first priority and plan multiple approaches for if initially unsuccessful
Unable to get separate parts of system to interface with each other	Low	High	Extensive research before starting development and be able to prove functionality of each part individually

Figure 3

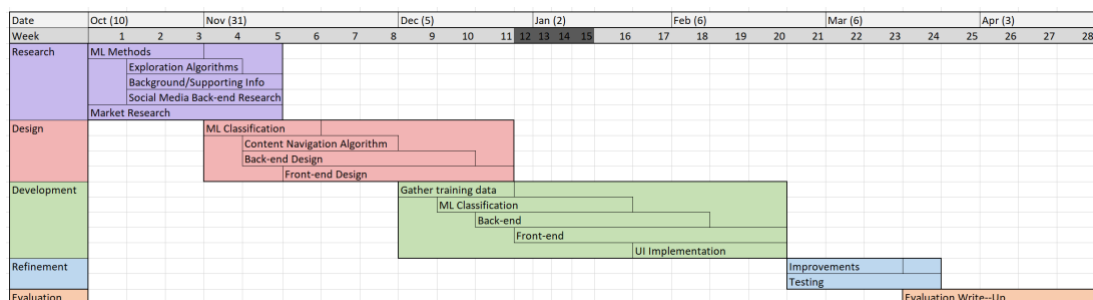


Figure 4

Field Location	Field Name	Mandatory	Field Type	Condition
Initial Sign Up	First Name	TRUE	String	
	Last Name	TRUE	String	
	Email	TRUE	Email	
	Mobile	FALSE	Mobile	
	Profile Image	FALSE	Image	
	About	FALSE	String	
	Date of Birth	FALSE	Date	
Additional Details	Type of Renter	FALSE	Typelist	
	Salary	FALSE	Integer	
	Profession	FALSE	Typelist	
	Add Current Property Details	TRUE	Boolean	
Current Property	Address	TRUE		Add current property = Yes
	Bedrooms	TRUE		Add current property = Yes
	Bathrooms	TRUE		Add current property = Yes
	Are renting?	TRUE		Add current property = Yes
	Current Price	FALSE		Renting = Yes
	Type of property	TRUE		Add current property = Yes

Figure 5

Data	Fields
Listing Views	User ID, Property ID, Date
Listing Enquiries	User ID, Property ID, Agent ID, Details, Date

Figure 6

Library	Reasoning
Pandas – Python	Provides easy to use data structures for manipulating database data within Python code
Numpy - Python	Provides various mathematical functions to help with the recommendation system
Pickle - Python	Persist data that has been calculated for the recommendation system
BeautifulSoup - Python	Parse HTML documents to extract data for web scraper
JWT - NodeJS	Generate and validate JSON Web Tokens for user authentication
Mongoose - NodeJS	Interface with MongoDB database and translate database objects to code
Bcrypt - NodeJS	Provides password hashing function
Google OAuth2Client - NodeJS	Use Google access token to retrieve protected user data from Google and to verify users' identities
Lottie - iOS	Provides framework for displaying .lottie animation files
Kingfisher -iOS	Optimised library for downloading and displaying images from the web
GoogleSignIn - iOS	Provides the functionality for adding sign-in with Google

Figure 7

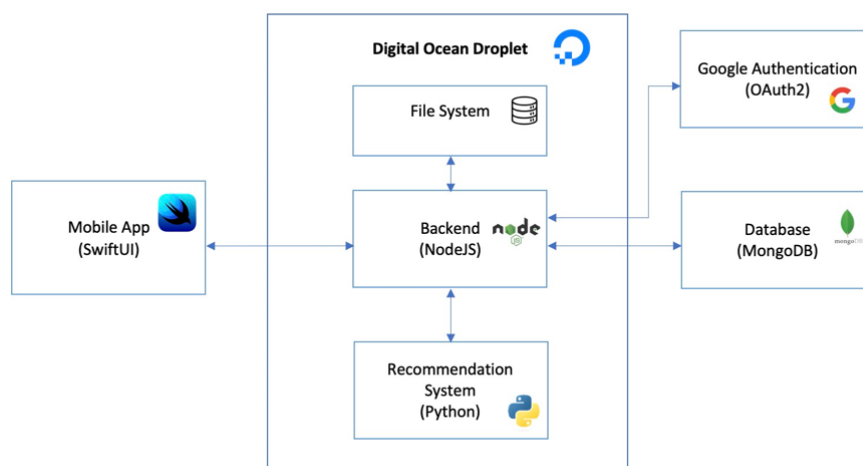


Figure 8

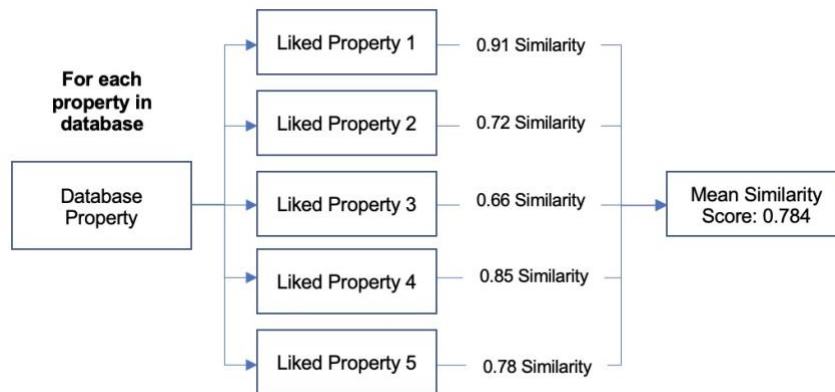


Figure 9

Rating	Condition
1	User has viewed a property listing and exited within 10 seconds
2	User has viewed a property listing for longer than 10 seconds
3	User has viewed a property listing for longer than 20 seconds, and scrolled to bottom/viewed additional details, or viewed the property twice
4	User has viewed a property for longer than 30 seconds, looked through more than 2 pictures and scrolled to bottom/viewed additional details, or explicitly liked the property from the discover (swipe) view
5	User has explicitly liked a property from the listing view

Figure 10

Persona	Preferences
1 - Young professional moving to London for new job in city, looking for a property to share with 2 others. £37,000 per year income each.	Property Type: Flat Location: Clapham Bedrooms: 3 Price: ~ £3,000 pcm
2 - 35-year-old working in design industry, sharing with one partner. £130,000 per year combined income.	Property Type: Apartment/Flat Balcony if possible Bedrooms: 1 or 2 Location: Peckham Price: ~ £4,300 pcm
3 - Second-year Imperial College university student, sharing in house of 5 students at other universities. High income parents to support on top of student loan.	Property Type: House Bedrooms: 5 Price: ~ £5,500 pcm Furnished: Yes

<p>4 - Low-income parent, living with partner and two children. Currently living in outer London but struggling with cost-of-living crisis.</p>	<p>Property Type: House</p> <p>Price: As low as possible</p> <p>Bedrooms: 4+</p> <p>Price: ~ £2,500 pcm</p>
<p>5 - 25-year-old founder of real estate application, living with partner. Works remotely so location does not matter.</p>	<p>Property Type: House</p> <p>Price: £10,000+ pcm</p> <p>Swimming Pool if possible</p>

Figure 11

Authentication Authentication actions for Login/Sign Up

POST /user/signup Create User Account	
POST /user/login Sign In User	
POST /user/googleLogin Sign In User with Google	
	POST /listing/ Get Listings
	POST /listing/discover/cb/{userid} Get Content-Based Recommendations
	POST /listing/discover/cf/{userid} Get Collaborative Filtering Recommendations
GET /user/{userid} Fetch User Account	GET /listing/user/{userid} Get Agent Listings
PUT /user/updateProfile/{userid} Update User Profile	POST /like/{userid} Get Liked Listings

Figure 12

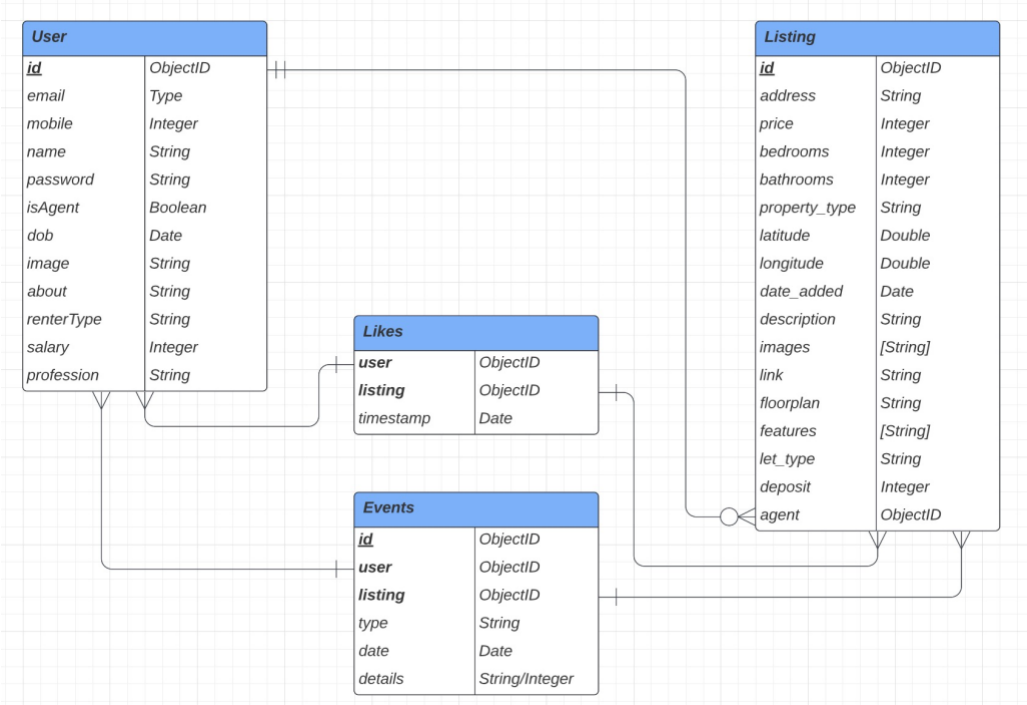


Figure 13

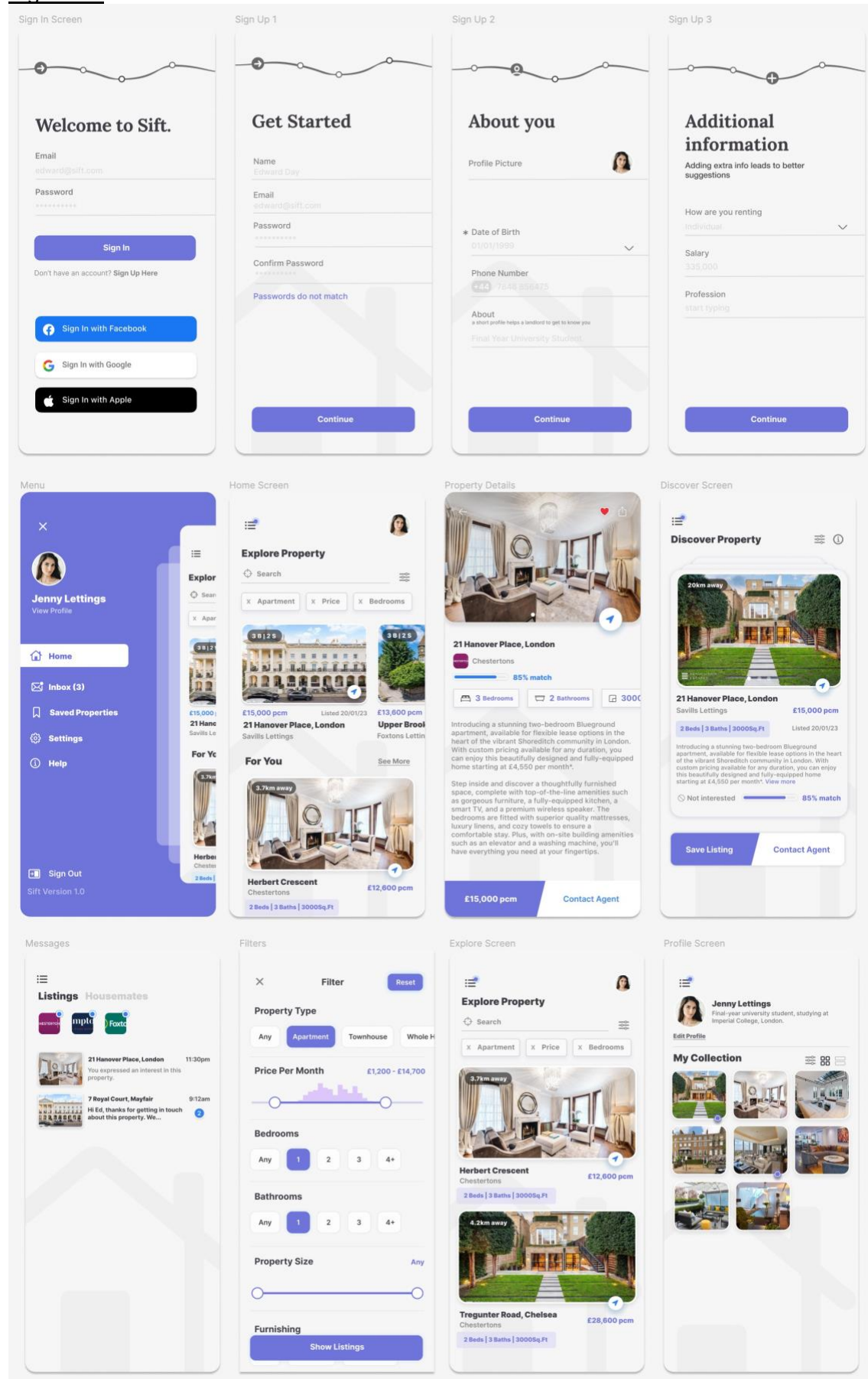


Figure 14

Task	Average Time (s) – Sift	Average Time (s) - Rightmove	Sift/Rightmove Time Factor
Create Account	48	44	1.09
Sign Out	5	5	1
Sign In	15	10	1.5
Find Property – Criteria A	20	47	0.43
Like Property – Criteria B	31	28	1.11
Find Property – Criteria C	18	18	1
Retrieve Liked Property	12	8	1.5
Overall	149	159	0.94

Figure 15

Question	Average Score (0-10)
How easy was the application to use?	8.7
How would you rate your experience using the application?	9
Do you think a recommendation system would be beneficial to potential renters?	9.3
Did you like the way the application looks?	9.3
How likely would you be to use this application over Rightmove?	7.3



Figure 16

	Average Similarity Rating	Average Estimated Satisfaction
Overall	87.2%	86.4%

Figure 17

Persona	Address	Beds	Price	Distance to Preferred Location	Property Type	Image
1	Creek Lane, Wandsworth	3	£3,500 pcm	2.5 miles	Flat	
1	Tachbrook Street, Pimlico	3	£3,000 pcm	2.7 miles	Flat	

1	Royal Oak Yard, London	3	£3,500 pcm	4.8 miles	Flat	
2	64 New Kent Road, London	2	£3,270 pcm	3.5 miles	Flat	
2	St. Katherine's Way, London	2	£3,792 pcm	3.8 miles	Flat	
2	Churchyard Row, London	2	£3,400 pcm	4.1 miles	Flat	
3	Ashbourne Road, Ealing	5	£4,600 pcm	No preference	House	
3	Barnfield Place, London	5	£4,650 pcm	No preference	Town House	
3	Hartington Road, West Ealing	5	£4,498 pcm	No preference	Semi-detached House	
4	Avenue Road, Southgate	4	£2,250 pcm	No preference	Maisonette	
4	Glenwood Road, South Tottenham	4	£2,500 pcm	No preference	Terraced House	
4	Braxted Park, Streatham	4	£2,800 pcm	No preference	Semi-detached house	
5	Frognaal, Hampstead	7	£41,167 pcm	No preference	Detached House	

5	The Bishops Avenue, Hampstead Garden Suburb	5	£26,000 pcm	No preference	Detached House	
5	Herbert Crescent, Knightsbridge	7	£67,167 pcm	No preference	House	

7.8 – External Links

API Documentation - <https://app.swaggerhub.com/apis-docs/EDWARD57DAY/Sift/1.0.0>

GitHub Repository - <https://github.com/edday57/Sift>

Demonstration Video - <https://www.youtube.com/watch?v=DJ0dBEAEvKI>