Table of Contents

[1 - Introduction 2](#_Toc128857838)

[1.1 - Background 2](#_Toc128857839)

[1.2 - Aim 3](#_Toc128857840)

[1.3 - Objectives 3](#_Toc128857841)

[1.4 - Deliverables 3](#_Toc128857842)

[2 - Project Plan and Methodology 4](#_Toc128857843)

[2.1 - Methodology 4](#_Toc128857844)

[2.2 - Scope and Requirements 5](#_Toc128857845)

[2.3 - Gantt Chart 5](#_Toc128857846)

[3 - Initial Research 5](#_Toc128857847)

[3.1 - Data 5](#_Toc128857848)

[3.1.1 - Property Listing Data 6](#_Toc128857849)

[3.1.2 - User Data 6](#_Toc128857850)

[3.2 - Technology Stack 7](#_Toc128857851)

[3.2.1 - Frontend (SwiftUI) 7](#_Toc128857852)

[3.2.2 – Backend (Express JS) 8](#_Toc128857853)

[3.2.3 - Database (MongoDB) 9](#_Toc128857854)

[3.2.4 - Machine Learning (Python SciKit Learn) 9](#_Toc128857855)

[3.2.5 - Deployment (DigitalOcean) 9](#_Toc128857856)

[4 - Methods 10](#_Toc128857857)

[4.1 - Recommendation System 10](#_Toc128857858)

# 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, from 34.3%, in 2011 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 that there could be a gap in the market for a system which could help to simplify this.

For this project, we will build 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 my 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, we must consider how we will decide what factors will influence these recommendations, and the methods in which users will be able to explore the content. We must also consider how we will gather the information about the property listings.

## 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
* 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

# 2 - Project Plan and Methodology

## 2.1 - Methodology

Diagram

Description automatically generatedThis project has been developed using a similar methodology to the standard agile workflow, with some adaptations to allow it to better suit this individual project (Jeong et al., 2008).

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

We then moved into a planning phase which had similarities to a classic agile inception phase. During this period, we further broke down each of the requirements, coming up with a rough idea of how each of these would be implemented and the technologies that would be required to do so. 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 we would need to change certain parts later down the line to ensure that everything would integrate with each other 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 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 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 us to fix any problems that we encountered 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 system requirements had been satisfied, we then

**ADD REFINEMENT STAGE WRITE UP ONCE COMLETED**

## 2.2 - Scope and Requirements

## 2.3 - Gantt Chart

**TO BE UPDATED TO REFELECT ACTUAL TIMELINE AND PROCESS**

Graphical user interface, application, table, Excel

Description automatically generated

# 3 - Initial Research

## 3.1 - Data

As the main function of the application is to presents users with information about properties, we require access to a large set of property listing data. 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 we want to implement user-user recommendations.

When collecting data, we must ensure that this complies with any ethical requirements, and that we have permission 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

The first consideration will be where we will get the property listing data from. 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, and the Kaggle datasets also do not contain all the additional information that we would want about a property available to rent, such as images, letting agents, deposits etc.

After more consideration, I decided that the most appropriate way to get data would be to create my own dataset. This could be done by either 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, we would seek to partner with estate agents, and they would provide details of their listings as this would help them to promote their properties and find tenants. However, for the scope of this project it seems more appropriate to use existing up to date data which could be gathered from other property listing sites such as Rightmove or Zoopla. Not only will we need access to the data within the app, but we will also need direct access to our own database of listing data to create an effective content-based recommendation system. Rightmove is the leading platform for property listings featuring over X properties, although it does not have a public API, so we would need to manually scrape the data from the webpages if we were to use this, which goes against their terms of service. Alternatively, Zoopla is Rightmove’s main competitor and provides a public API that allows us to query their database and retrieve details of property listings, which gives us two options:

1 – Scrape data from the Zoopla API directly into our own database, and have our application only interact with the scraped data.

2 – Scrape data from the Zoopla API which can be used for machine learning recommendation purposes and have our app interface with our backend for recommendations, and the Zoopla API for listing data.

For simplicity, option 1 would be easiest as it would mean that all listing data can be accessed directly from our own database. However, scraped property data would need to be kept up to date as listings may be removed or modified so we would need to implement regular checks to ensure that the data we are providing to our users matches what Zoopla has listed.

### 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 follows:

Table

Description automatically generated

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 we were to add analytics functionality for estate agents. The implicit in-app data to be captured and sent to the backend is as follows:

Table

Description automatically generated

## 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. In this writeup, we will give a brief overview of both frameworks and explain why SwiftUI has been chosen over UIKit as the frontend framework for our iOS app.

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, that are not available in UIKit.

After careful consideration of the available options, we decided to use SwiftUI as the frontend framework for our 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, P. and Skublewska-Paszkowska, M., 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 our 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, and using SwiftUI will enable us to create a highly functional and user-friendly application that meets the needs of our 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 our system (Statista, 2022).

Express – When using JavaScript, we would use Node.js 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.

Python – When using Python, we could use Django or Flask as a 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.

Although any of these three choices would have been suitable for our project, Express was chosen as the framework to be used for our backend. Some of the primary reasons for this was 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 we would be able to deliver the project 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).

Our recommendation system will run as a Python script so we must also consider how this will fit in with our backend solution. This makes a custom backend preferable over services such as Firebase or MongoDB Realm as it means that all requests can be processed centrally.

Both above languages would be suitable to use for the backend and both 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 us to integrate Python scripts to our code if we use a JavaScript based approach so this will not be an issue for us. 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 us due to the fact we will be using Swift for the frontend of our 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. I will choose to use the JavaScript based Node.js stack for the majority of the functionality due to having the most prior coding experience with it.

### 3.2.3 - Database (MongoDB)

### 3.2.4 - Machine Learning (Python SciKit Learn)

### 3.2.5 - Deployment (DigitalOcean)

# 4 - Methods

## 4.1 - Recommendation System

### 4.1.1 - Background

Over recent years, we have seen a huge change in the way that we consume online information, with algorithmically driven content curation becoming increasingly common across online platforms (DeVito, M.A., Gergle, D. and Birnholtz, J., 2017). This is being applied across a variety of forms of online media, ranging from product recommendations within online shopping sites (Hwangbo, H., Kim, Y.S. and Cha, K.J., 2018), to social media applications. TikTok is a prime example of this which completely take content navigation out of the users control and present them with tailored content that the system thinks they will be interested in and due to the popularity of its algorithm it has become one of the world’s fastest growing and most popular social media sites.

### 4.1.2 - Justification

When deciding on a property, renters take a variety of its attributes into account such as the price, size, location etc (Van Ham, M., 2012), and tenants are seen to then 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, Y., Schedl, M., Cremonesi, P. and Pasi, G., 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. By implementing a personalised recommendation system which will direct users directly to the content that they want without requiring any additional effort, we 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, P.A. and Kramer, H., 1994) outlines that if users are overloaded with 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 which will aim to remedy this by sifting out the unnecessary information and helping to direct the user towards properties that have a higher likelihood of being suitable to them. If we can improve the accuracy of recommendations and the proportion of relevant content that the user receives, then this will have a significant effect on overall user satisfaction when using the application (Liang, T.P., Lai, H.J. and Ku, Y.C., 2006).

### 4.1.3 - 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 our 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 showed 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, M.A., Gergle, D. and Birnholtz, J., 2017). By taking these main reasons into account when designing our system, we can aim to minimise the resistance that our algorithm will receive from our users. A lack of choice will not be an issue within our 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, we will also make it clear to them what the algorithm bases their results off In our case this will be simply 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, we must aim to make it as accurate and responsive to the user’s preferences as possible to ensure that our users can quickly see that it works as a useful tool to help them in their search for the perfect property.

We must also consider the types on recommendation that can be used, and their limitations. Initially, we 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 our 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 easy for a user to understand why they are being shown this content. They also 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 and 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 needing any information about what is being recommended. Instead, they work by analysing multiple user’s 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 benefit that can provide users with recommendations that may have been outside of their initial preferences that they end up liking. The main drawback of collaborative filtering is the cold-start problem, which is how it can recommend new items before they have had any interactions (Mohamed et al., 2019). For this reason, we will initially not use a collaborative filtering method as the system will provide information about so many properties that gathering user interaction data will not initially be feasible. However, after implementing a content-based recommendation system, if we gather interaction data within the app then this could later be used to implement a hybrid system which combines both content-based and collaborative filtering techniques in order to provide better recommendations and to reduce the impact of each of their drawbacks.

### 4.1.4 - Design

The content-based filtering system was built as a Python script which could be ran by the API, taking various parameters and returning the ID’s of properties which could then be retrieved from the database and recommended to the user.

Inputs:

## 4.2 - Backend

### 4.2.1 - Authentication

## 4.3 - Frontend

### 4.3.1 - Initial Design

The frontend was developed with a design first approach, using the tool Figma to assist with our UI design. We first created wireframes for each of the screens within the app which covered all the original functionality that had been outlined, and then moved on to create high-fidelity UI mock-ups which showed exactly how the screens should look within the app.

**Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generatedHome Page**

Functionality:

* Allow users to search for properties and be taken to the Browse page
* Show preview of recently added properties from Browse page
* Show preview of recommended properties from For You page

**Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, text, application

Description automatically generated****Property Listing Page**

Functionality:

* Show details of property
* Allow user to like property
* Show location of property on map
* Allow users to browse through multiple images
* Allow user to contact agent through provided contact details

**Graphical user interface, text, chat or text message

Description automatically generatedLetter

Description automatically generatedDiscover Page**

Functionality:

* Present user with recommended properties in card stack
* Allow user to swipe left or right to like or pass on a property
* Tap image to cycle through image gallery
* Add explicit filters to recommendations
* Info button to explain how the recommendation system provides them with suggestions
* Animate card stack to make it clear to user that they are swiping through suggestions

**Browse Page**

Functionality:

* Fetch and present properties from database
* Allow users to filter and change sort order for properties
* Pagination to fetch more properties when user reaches end of page

1. **Results**

With

Algorithm

User participation and involvement

Outline functionality

**Background Research**

The problem – user research

Similar products – Tinder/Rightmove

Gathering data – user/listing

Technologies – frontend/backend/machine learning/database

**Methods**

App flow diagram/full functionality overview

MVVM Overview

Frontend UI mock-up

Diagram to show links between frontend backend database etc

Scraper data collection explanation

Recommender explanation

**Results and Discussion**

Overview of Swift code – Per page explanation

Structure overview – MVVM

Testing?

API Documentation/Explanation

Evidence of recommended similar properties

Further development plans

**Evaluation**

Effectiveness of recommender - quantitative

App interface – qualitative

**Appendix**

|  |  |  |
| --- | --- | --- |
| Reference | Information | Where this will be used |
| *Office for National Statistics (ONS), released 5 January 2023, ONS website, statistical bulletin, Housing, England and Wales: Census 2021* | An increase in the proportion of households that rented their accommodation, to 37.3%, in 2021 (from 34.3%, in 2011). | Introduction |
| *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* | Rightmove has stated 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%. | Introduction |
| Van Ham, M., 2012. Housing behaviour. *The SAGE handbook of housing studies*, pp.47-65. | When deciding on a property, renters take a variety of its attributes into account such as the price, size, location etc. | Introduction |
| 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. | Tenants are seen to then choose a property that best fits the attributes that they are looking for. | Introduction |
| Deldjoo, Y., Schedl, M., Cremonesi, P. and Pasi, G., 2020. Recommender systems leveraging multimedia content. *ACM Computing Surveys (CSUR)*, *53*(5), pp.1-38. | Recommendation algorithms can identify trends in the human decision-making process and then be used to provide content which is deemed to be of interest to the user. | Introduction |
| 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. | Agile methodology | Methodologies |
| 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. | SwiftUI vs UIKit | Technology Choices - Frontend |
| Gilchrist, E., 2021. SwiftUI vs UIKit: A Case Study on How a Declarative Framework Can Improve Learnability of UI Programming. | Learnability of declarative framework | Technology Choices - Frontend |
| Statista (2022). Most used web frameworks among developers worldwide, as of 2022 [Online] Date Accessed: 28th February 2023 | Most common Backend Frameworks | Technology Choices - Backend |
| Muittari, J., 2020. Modern web back-end. | Details of Backend Frameworks | Technology Choices - Backend |
| Zhang, M. and Liu, Y., 2021. A commentary of TikTok recommendation algorithms in MIT Technology Review 2021. *Fundamental Research*, *1*(6), pp.846-847. | TikTok is one of the world's most attractive and fastest-growing social media platforms. It has more than 2.6 billion downloads globally and over 100 million users in the US. The “secret weapon” is its unique methodology of discovering and delivering content. | Recommendation system example |
| 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. | Ecommerce uses of recommendation systems | Recommendation system example |
| 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). | Resistance to algorithmic change | Recommendation Systems – Background |
| Bozdag, E., 2013. Bias in algorithmic filtering and personalization. *Ethics and information technology*, *15*, pp.209-227. | Algorithmically driven content curation has become increasingly common across online platforms such as Facebook | Recommendation Systems – Background |
| 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. | Resistance to algorithms – Facebook newsfeed | Recommendation systems - considerations |
| Amy Kraft, 2016. Backlash Continues over Instagram’s New Algorithm. [Online] Date Accessed: 27th February 2023 | Resistance to algorithms – Facebook newsfeed | Recommendation systems - considerations |
| 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. | Content Based vs Collaborative vs Hybrid drawbacks/advantages |  |
| Beck, S.J., 1951. Review of Human behavior and the principle of least effort: An introduction to human ecology. | Individuals adopt the course of action that involves the least work, even if it means sacrificing quality or quantity of information. |  |
| Herbig, P.A. and Kramer, H., 1994. The effect of information overload on the innovation choice process: Innovation overload. *Journal of Consumer Marketing*. | If users are overloaded with information, then they are unable to locate what they need most and may overlook what they consider critical. | Algorithm |
| Shahabi, C. and Chen, Y.S., 2003. Web information personalization: Challenges and approaches. *Databases in Networked Information Systems: Third InternationalWorkshop, DNIS 2003, Aizu, Japan, September 22-24, 2003. Proceedings 13 3*, pp.5-15. | Content based recommendation systems can help to alleviate the problem of information overload. | Content based recommendation system - justification |
| 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. | The number of items recommended to the user and the recommendation accuracy both have significant effects on overall user satisfaction. | Content based recommendation system - justification |
| 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. | Recommendation systems are not without their challenges. Cold start, novelty, convergence | Content based recommendation system - justification |
| 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 | NodeJS better for response time | Introduction  Content based recommendation system - justification |
|  |  | Recommendation System - Considerations |
| 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/ | Mobile app abandonment |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |