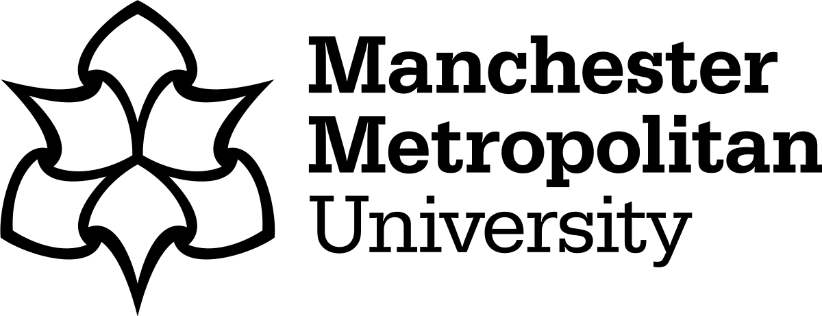
Manchester Metropolitan University



Final Year Dissertation

‘Empowering Users Through Transparent and Controllable On-Device Image Recommendation Using Visual Features and Lightweight Indexing’

|  |  |
| --- | --- |
| **Author:**  Husnain Ahmed | **Supervisor:**  Ashley Williams |
| 21308666 |  |
| **Degree:**  BSc (Hons) Software Engineering | **Department:**  Computing and Mathematics  **Faculty:**  Science and Engineering |

*May 2025*

Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work. This work has been carried out in accordance with the Manchester Metropolitan University research ethics procedures and has received ethical approval number 75120**.**

Signed

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Acknowledgements

This dissertation is dedicated to my family, who pushed me to work my hardest and provided me with support to finish my dissertation.

Abstract

This project presents an on-device image recommendation web application that leverages MobileNetV2 embeddings and the Annoy library to deliver real-time suggestions based on visual and semantic similarity.

The system is designed with a research direction focused on empowering users through transparent and controllable on-device image recommendation using visual features and lightweight indexing.

Prioritising transparency, user control, and privacy, the system diverges from traditional systems by performing all computations in the user's browser and not storing any user interaction data without their explicit consent, as well as by showing a visual representation of how user interactions refine future image recommendations so that they can understand the rationale behind their image recommendations.

An intuitive, user friendly interface allows for local image uploads and incorporates a rating-based feedback mechanism that allows users to shape future recommendations and explore visually or semantically related content in a more transparent and controllable way.

The system's effectiveness is evaluated through a combination of application testing, user feedback and an evaluation of user interaction data, demonstrating its potential to provide accurate, efficient, and user-centric image recommendations.

This research project investigates how this design influences user interaction and attention toward visual content, and how users perceive the benefits, drawbacks, and intrusiveness of such a system in various different contexts, including creative work, education, and browsing.

Furthermore, the project explores how transparent keyword-based recommendations, user-guided ratings, and the absence of cloud processing influence user trust, perceived autonomy, and system acceptance.

Table of Contents

[Declaration ii](#_Toc197023909)

[Acknowledgements iii](#_Toc197023910)

[Abstract iv](#_Toc197023911)

[List of Figures vii](#_Toc197023912)

[List of Tables viii](#_Toc197023913)

[Abbreviations ix](#_Toc197023914)

[Dataset x](#_Toc197023915)

[1. Introduction 1](#_Toc197023916)

[1.1. Project Background 3](#_Toc197023917)

[1.2. Aim and Objectives 3](#_Toc197023918)

[1.2.1. Aims 3](#_Toc197023919)

[1.2.2. Objectives 4](#_Toc197023920)

[1.3. Dissertation Structure 5](#_Toc197023921)

[2. Literature Review 6](#_Toc197023922)

[2.1. On Existing Image Recommendation Systems 6](#_Toc197023923)

[2.2. Content-Based Filtering (CBF) 7](#_Toc197023924)

[2.3. Deep Learning for Feature Extraction 8](#_Toc197023925)

[2.4. Semantic Analysis 9](#_Toc197023926)

[2.5. User Feedback and Trust in Image Retrieval 10](#_Toc197023927)

[2.5.1. Relevance Feedback 11](#_Toc197023928)

[2.5.2. Implicit Feedback 12](#_Toc197023929)

[2.5.3. Explainable AI (XAI) 14](#_Toc197023930)

[3. Design 16](#_Toc197023931)

[3.1. Requirements 16](#_Toc197023932)

[3.2. Design Choices 16](#_Toc197023933)

[3.3. Hardware / Software Requirements 17](#_Toc197023934)

[3.4. System Architecture 18](#_Toc197023935)

[3.5. UML Class Diagram 19](#_Toc197023936)

[3.6. Component Design 19](#_Toc197023937)

[3.6.1. Client-Side Design 19](#_Toc197023938)

[3.6.2. Server-Side Design 22](#_Toc197023939)

[3.7. Data Flow 23](#_Toc197023940)

[3.8. Database Design 24](#_Toc197023941)

[3.9. User Interface Design Principles 24](#_Toc197023942)

[4. Implementation 25](#_Toc197023943)

[4.1. Project Setup 25](#_Toc197023944)

[4.2. Image Feature Extraction and Similarity Search 25](#_Toc197023945)

[4.3. Database Implementation 26](#_Toc197023946)

[4.4. Client-Side Implementation 28](#_Toc197023947)

[4.5. Server-Side Implementation 28](#_Toc197023948)

[4.5.1. Node.js Server 29](#_Toc197023949)

[4.5.2. Python Server 33](#_Toc197023950)

[4.6. Deployment Of The System 34](#_Toc197023951)

[4.7. Challenges Faced 35](#_Toc197023952)

[4.8. Testing 36](#_Toc197023953)

[5. Evaluation 40](#_Toc197023954)

[5.1. Achievement of Objectives 40](#_Toc197023955)

[5.2. Comparison with Requirements 40](#_Toc197023956)

[5.3. User Feedback 41](#_Toc197023957)

[5.4. Performance Evaluation 43](#_Toc197023958)

[5.5. Ethical, Societal, and Legal Considerations 48](#_Toc197023959)

[5.6. Limitations and Future Work 48](#_Toc197023960)

[6. Conclusion 50](#_Toc197023961)

[6.1. Completed Work 50](#_Toc197023962)

[6.2. Limitations 51](#_Toc197023963)

[6.3. Future Work 52](#_Toc197023964)

[6.4. Hindsight 54](#_Toc197023965)

[6.5. Closing Remarks 54](#_Toc197023966)

[References 55](#_Toc197023967)

[Bibliography 59](#_Toc197023968)

[Appendices 60](#_Toc197023969)

List of Figures

[Figure 1: Relevance Feedback Implementation 12](#_Toc197023970)

[Figure 2: Bar Chart For User Preference Vector 15](#_Toc197023971)

[Figure 3: Force Directed Graph for Similarities Between Uploaded Image And Dataset 15](#_Toc197023972)

[Figure 4: High Level System Architecture Diagram 18](#_Toc197023973)

[Figure 5: Class Diagram 19](#_Toc197023974)

[Figure 6: Initial UI Design diagram 20](#_Toc197023975)

[Figure 7: Final completed UI design (After Development) 21](#_Toc197023976)

[Figure 8: Server-Side Component Diagram 22](#_Toc197023977)

[Figure 9: Data Flow Diagram 23](#_Toc197023978)

[Figure 10: ‘Annoy\_neighbors.json’ file screenshot 27](#_Toc197023979)

[Figure 11: ‘Metadata.json’ file screenshot 27](#_Toc197023980)

[Figure 12: Call to the search images endpoint 30](#_Toc197023981)

[Figure 13: Node.js search-images endpoint code 30](#_Toc197023982)

[Figure 14: Call to the consent endpoint 31](#_Toc197023983)

[Figure 15: Node.js consent endpoint code 31](#_Toc197023984)

[Figure 16: 'User\_ratings.json' file screenshot 32](#_Toc197023985)

[Figure 17: Node server's call to the python server to get the semantic embedding 33](#_Toc197023986)

[Figure 18: Python server for semantic search 34](#_Toc197023987)

[Figure 19: GitHub Actions WorkFlow 34](#_Toc197023988)

[Figure 20: Jest tests all passed 37](#_Toc197023989)

[Figure 21: Postman manually testing endpoints 38](#_Toc197023990)

[Figure 22: PyTest tests all passed 39](#_Toc197023991)

[Figure 23: Feedback form 41](#_Toc197023992)

[Figure 24: User feedback - Did you get any accurate recommendations? 42](#_Toc197023993)

[Figure 25: User feedback - Did you appreciate the transparency in how your ratings were used through the dynamic bar chart? 42](#_Toc197023994)

[Figure 26: Violin plot of user ratings by relevance 44](#_Toc197023995)

[Figure 27: Bar chart of user rating relevance 44](#_Toc197023996)

[Figure 28: Line Graph of user rating and relevance over time 45](#_Toc197023997)

[Figure 29: Mean user ratings 46](#_Toc197023998)

[Figure 30: ANOVA Testing results 46](#_Toc197023999)

[Figure 31: Welch's two sample T-test results 47](#_Toc197024000)

[Figure 32: User suggestions for improvements from the feedback form 53](#_Toc197024001)

List of Tables

[Table 1: Hardware 17](#_Toc196943077)

[Table 2: Software 17](#_Toc196943078)

[Table 3: Participant Responses 43](#_Toc196943079)

[Table 4: ANOVA Results 47](#_Toc196943080)

[Table 5: T-test Results 47](#_Toc196943081)

Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| ANOVA | Analysis Of Variance |
| CBF | Content Based Filtering |
| CNN | Convolutional Neural Networks |
| CORS | Cross-Origin Resource Sharing |
| Df | Degrees of Freedom |
| GDPR | General Data Protection Regulation |
| XAI | Explainable AI |
| WNID | WordNet ID |

Dataset

The Tiny ImageNet 200 dataset was used.

It can be found here for download: <https://cs231n.stanford.edu/tiny-imagenet-200.zip>

This image set contains 200 classes of non-specific, general images ranging from animals to food to nature. Each class contains 500 images.

This specific dataset was chosen with efficiency in mind due to the system running locally on the user’s system, a larger dataset may have caused longer runtimes.

# Introduction

As the internet has developed and become widespread, the concept of image retrieval and how it works has significantly evolved. Early systems had a heavy reliance on manual set up textual tags, sometimes called labels, and keyword-based search which was highly inefficient though suitable for the times. Users could only find images by searching for those specific tags. However, the development of digital cameras and online image-sharing platforms has resulted in an explosion of visual data, rapidly surpassing the capability of manual annotation through the growing volume of digital images (Hwang et al., 2012).

More complex methods of image retrieval had to be developed as a result of the explosion in image data and the intrinsic shortcomings of keyword-based approaches in capturing the rich visual information contained within digital images. Keyword-based searches tend to suffer from the ‘semantic gap’, where the labels associated with an image do not accurately represent its visual content and the user's intended meaning. As the results could potentially seem irrelevant, this can lead to a lack of user trust in the system which is not ideal (Milvus.io, 2025).

This report documents the development of an image recommendation system that is designed to address these challenges by providing users with visually similar images based on an uploaded image that is chosen by the user. The system aims to move beyond simple keyword matching by analysing the visual content of images directly and searching for images that are visually similar.

The research goal of this paper is to empower users through transparent and controllable on-device image recommendations making use of visual features and lightweight indexing. This method differs from conventional "black box" systems that provide results without providing an explanation by placing a higher priority on openness, user control, and privacy.

The system leverages recent developments in several areas such as deep learning and semantic analysis, to bridge the semantic gap and improve the accuracy and relevance of image retrieval.

Deep learning models, particularly Convolutional Neural Networks (CNNs), have shown that they are well adapted to the purpose of extracting visual features from images.

Semantic analysis techniques, on the other hand, are used to understand the high-level concepts and relationships within images, allowing for the retrieval of images based on meaning rather than just low-level visual features.

By combining on-device processing with a client-server architecture, the system aims to provide an efficient approach to image retrieval that boosts the user’s trust in the system.

Furthermore, the integration of user feedback in the form of relevance ratings from 1 to 10 is a core aspect of this system. This mechanism is designed to enhance user trust by allowing users to actively participate in the image curation process. Through this the users can guide the system's ‘learning’ process to improve the quality of their future recommendations and see this visually represented to them through dynamically updated graphs.

This user-guided approach not only increases the accuracy of the system but also helps foster a sense of transparency and autonomy that empowers users to understand and control the image recommendation process.

More specifically, this report will explore how this user-guided approach addresses the following research questions:

* How does the image recommendation interface reshape users' interactions with and attention toward visual content?
* What are the perceived benefits, drawbacks, and the varying levels of intrusiveness of such a system, compared to the other commercially available options?
* How do transparent keyword-based recommendations, user-guided ratings, and the absence of cloud processing or other third party interactions have an impact on user trust, perceived autonomy, and system acceptance?

The report will explore how this user-guided approach affects trust, autonomy, and interaction within the image retrieval context.

## Project Background

This project will tackle the shortcomings of the conventional keyword-based picture retrieval, motivated by the need for more user-friendly and efficient image search techniques. Systems that can reliably and effectively retrieve photos based on their visual content are in high demand due to the growing availability of digital images.

The richness and complexity of visual information cannot be adequately captured by traditional methods that rely on manual annotation and textual explanations. In order to produce some relevant, user-focused picture search results this project intends to create an image recommendation system that makes use of recent developments in the fields of both deep learning and semantic analysis. With the goal of increasing user control and trust via openness integrating their own actions into the recommendation algorithm.

## Aim and Objectives

Here I will outline the purpose of this paper.

### Aims

This project aims to:

1. Develop an image recommendation system that provides users with visually similar images based on an uploaded query image, using deep learning for feature extraction and semantic analysis to enhance retrieval accuracy.
2. Prioritise transparency and user control by incorporating a user feedback mechanism that allows users to rate the relevance of recommended images.
3. Display, through the sue of various graphs and charts, how the users input affects the algorithm and how the images are being recommended.

### Objectives

The objectives of this project are as follows:

* Conduct a thorough review of existing image retrieval techniques, focusing on content-based filtering (CBF), deep learning for feature extraction, semantic analysis, various feedback metrics and Explainable AI (XAI).
* Design and implement a system architecture that combines on-device feature extraction with a client-server approach for efficient image retrieval.
* Use ‘MobileNetV2’ for on-device image feature extraction and Annoy for efficient similarity search.
* Implement a semantic embedding function using Sentence Transformers to incorporate semantic information into the image retrieval process.
* Develop a user interface that allows users to upload query images, view recommended images, and rate the received recommendations.
* Design and implement a data storage mechanism to store user ratings for potential future use in refining the recommendation algorithm, provided the user consents to the storage of their data.
* Evaluate the system's performance, usability, and the impact of user feedback on the accuracy of recommendations and user trust.
* Document the development process, system architecture, and evaluation results in a comprehensive report.

## Dissertation Structure

The remainder of this dissertation is structured as follows:

Chapter 2 contains a Literature review discussing related research in the field of image-based recommendation systems.

Chapter 3 provides an overview of the design of the system.

Chapter 4 explains the implementation of the system as well as the testing process.

Chapter 5 is an evaluation of the system.

Chapter 6 is the conclusion of this dissertation.

# Literature Review

The development of image recommendation systems draws upon several key areas of research, including content-based image retrieval, also referred to as content-based filtering (CBF) to retrieve similar images. Deep learning, also known as deep search, for the purpose of image feature extraction, semantic analysis as well as the use of data gained from user feedback and user interaction with the system and Explainable AI (XAI) to ensure transparency and allow the user to understand how the system is working, especially with their interactions.

This section will delve into these areas and evaluate pre-existing systems and studies that made use of these features.

## On Existing Image Recommendation Systems

Image recommendation systems are designed to suggest images to users based on their preferences or search queries. These systems have evolved significantly, driven by advancements in computer vision, machine learning, and information retrieval. Early systems often relied on simple techniques such as keyword-based search or collaborative filtering. However, modern systems leverage more sophisticated approaches, including content-based filtering (CBF) and deep learning, to provide more relevant and personalised recommendations.

CBF is a core component of many image recommendation systems, used to filter the recommended images by their content, one of the features that can be used for this are the labels, or tags which may be assigned to images within datasets or on image sharing websites like Pinterest or Flickr, these may not always be accurate though as they are created by users. Due to this, the need for alternative methods of filtering arises, such as size, visual features, image quality or name. (Zhang et al., 2017).

Generally, how image recommendation systems work is by receiving an image via some sort of input, manual or not, and then extracting some of the image’s features into a vector, a number representing those features. An algorithm will then attempt to search for images with similar image features based on that extracted vector and will display the retrieved image to the user.

Some image recommendation systems work by leveraging user behaviour data from web search engines to infer user preferences for images. This involves a two-stage process: first, inferring a user's general interests, and then estimating if the user will be interested in a specific image (Li et al., 2014).

More recent systems have integrated deep learning models, particularly Convolutional Neural Networks (CNNs), to extract high-level features from images. These features capture complex semantic information, enabling the system to retrieve images that are not only visually similar but also semantically related. However, as AI systems become more and more complex it is crucial to ensure their transparency and explainability so as to not diminish user trust (Paz-Ruza et al., 2024).

Furthermore, semantic analysis techniques, such as those using transformer models, are employed to bridge the semantic gap between low-level visual features and high-level human understanding, as discussed in papers like “Five sources of bias in natural language processing" (Hovy and Prabhumoye, 2021).

User interaction and feedback also play a crucial role in refining image recommendation results. Systems may incorporate explicit feedback, such as user ratings, or implicit feedback, such as click-through data, to learn user preferences and improve the accuracy of recommendations. The data from this feedback will slightly alter the ‘weights’ of the algorithm to fine tune it towards being more accurate with its recommendations in the future.

## Content-Based Filtering (CBF)

Content-based filtering aims to retrieve images from a database based on their visual content, such as colour, texture, and shape. Modern CBF systems have significantly benefited from the progress made in deep learning. While traditional CBF relied on handcrafted features such as colour histograms, texture, and shape, the focus has shifted to automated feature extraction using CNNs (Hameed et al., 2021).

For such systems, there are two major components, image representation for image indexing and similarity measure for database search. Image representation is expected to be discriminative so as to distinguish images from one another whilst similarity measure is used to reflect the difference in semantics. (Li et al., 2021).

Recent research has explored novel approaches to improve the efficiency and effectiveness of CBF systems. For instance, studies have investigated the use of compact descriptors and indexing techniques to accelerate the search process in large-scale image databases (Hong et al., 2017).

Furthermore, researchers have explored the integration of contextual information and user interaction to enhance the relevance of retrieved images. Specifically, the introduction of the Earth Mover's Distance as a more effective measure for comparing image color distributions, demonstrating improved retrieval accuracy (Rubner et al., 1998).

My system uses extracted image features as a basis for recommending images from the dataset, before incorporating user feedback through ratings. The initial recommendations make good use of image features and keywords through the MobileNetV2 model which can analyse an inputted image and generate a word that it believes matches the images content. Though this could be improved through a better use of compact descriptors as some of the class names or ‘labels’ in the image set are semantically similar to each other, for example ‘dog’ and ‘chihuahua’. Manually naming these to be more distinct from each other may have resulted in more accurate recommendations.

## Deep Learning for Feature Extraction

Feature extraction is the first process in content based image retrieval, with the aim of successfully converting the human perception of an image into a numerical description that can be manipulated by machines (Hameed et al., 2021).

These numerical representations, or vectors, are then used as a base to find similar images.

So far, the most common method for retrieving multimedia content from an archive consists of using meta-data associated to the images such as the timestamp, the geolocation, keywords, tags, labels or short descriptions, and perform the retrieval task through a text-based search (Piras and Giacinto, 2017).

While this works, it is not very efficient and can be improved upon.

Deep learning, particularly convolutional neural networks (CNNs), has revolutionised image feature extraction. CNNs can automatically learn hierarchical representations of images, capturing both low-level and high-level features that are relevant for image similarity. CNNs are composed of multiple convolutional and subsampling layers with non-linear activations, designed to process input images through a series of learned filters. These filters extract increasingly complex features, making CNNs a highly effective architecture for visual information analysis (Tzelepi and Tefas, 2018).

They are best suited for 2D data, consisting of a convolutional filter for transforming 2D to 3D which is quite strong in performance and is a rapid learning model. For classification process, it needs a lot of labelled data. However, CNN’s face issues, such as local minima, a slow rate of convergence, and the potential for interference by humans during the training process (Suganyadevi et al., 2021).

Recent advancements in CNN architectures, such as the development of more efficient and lightweight models, have made it possible to perform feature extraction on devices with limited computational resources. This is particularly relevant to this project, which aims to incorporate on-device processing.

Furthermore, research has focused on developing more robust and discriminative feature representations by exploring novel training techniques, loss functions, and attention mechanisms. These advancements have led to significant improvements in the accuracy and performance of image retrieval systems.

I have done this through the use of the python package ‘Annoy’ which allows for the creation of an inverted index that maps extracted image features to labels found in the dataset that can then be sued for an approximate nearest neighbours search.

## Semantic Analysis

Semantic analysis is a sub field of Natural Language Processing (NLP) that has the aim of bridging the "semantic gap" between low-level visual features and high-level human understanding of images.

What this means in practice is that the keyword related to the image that the system is searching for may not exist, to solve this problem the system will ‘understand’ the underlying meaning of the keyword, for example ‘apple’ may be altered to ‘fruit’, and searches for the closest possible match in the dataset. If the dataset is limited, like in the case of this project, the closest possible match may not be very accurate (Stankevičius and Lukoševičius, 2024).

Techniques like semantic embedding, which map images and text into a shared embedding space, have shown promise in improving the relevance of image retrieval results. The semantic gap can also be bridged via the use of ontologies (Fernández et al., 2011).

Recent studies have explored the use of transformer-based models, such as Sentence Transformers, to generate more effective semantic embeddings for both images and text. These pre trained models can capture complex semantic relationships and contextual information, leading to improved retrieval accuracy.

In this system, a Python-based server utilises the All-MiniLM-L6-v2 Sentence Transformer model to generate semantic embeddings of keywords. This process allows the system to understand the search query at a deeper level than simple keyword matching.

Studies have also explored how semantic analysis can be used in specific domains, such as using domain-specific ontologies for the analysis of traditional Chinese medicine formulas (Yin and Zhang, 2024). This paper also utilised the All-MiniLM-L6-v2 Sentence Transformer model, as I did and as such is more relevant to this project.

Additionally, researchers have investigated the integration of knowledge graphs and ontologies to provide a more structured representation of semantic information, enabling more accurate and interpretable image retrieval.

Furthermore, it must be taken into account that NLP is vulnerable to bias, this can arise from the type of data used for training, the annotation process, input representations such as word embeddings, the chosen model that will have been trained on a specific set of data and have a bias towards that data as well as research design as more NLP research focuses on English and other Indo-European text sources (Hovy and Prabhumoye, 2021).

## User Feedback and Trust in Image Retrieval

User feedback plays a crucial role in improving the performance and user experience of image retrieval systems. Relevance feedback, where users provide explicit ratings on retrieved images, has been shown to be effective in refining search results and learning user preferences.

Recent research has focused on developing more sophisticated methods for incorporating user feedback, such as implicit feedback techniques that infer user preferences from their interactions with the system. Additionally, there is a growing emphasis on building user trust in image retrieval systems by promoting transparency, explainability, and control.

Studies have explored the impact of different interface designs and feedback mechanisms on user trust and satisfaction. It has been found that explaining the outputs of recommendation systems helps boost user trust by a large margin, granting users insight into any automated decisions made by data-treating systems is also mandated by the European Union’s General Data Protection Regulation (GDPR) so it is a very good decision to do so (Paz-Ruza et al., 2024).

Furthermore, researchers have investigated the use of techniques like Explainable AI (XAI) to provide users with insights into how the system works and why certain images are recommended.

### Relevance Feedback

Relevance feedback is a technique where users explicitly judge the relevance of retrieved images, typically by marking them as "relevant" or "not relevant." This information is then used to refine the subsequent search results, often by re-weighting the importance of different image features.

One such paper has made use of relevance feedback for the purpose of improving image recommendations, it was found that users often struggle to describe the qualities of the image they want in words, one may focus on colour, another on texture. Even when considering the same feature people may have differing interpretations of similarity. Using user feedback, whether a thumbs up or down to determine if an image is relevant or a numerical rating system. This user feedback can be used to dynamically update the weights assigned to different image features and can allow the algorithm to refine its search results (Yong Rui et al., 1998).

Recent studies have explored various ways to improve the effectiveness of relevance feedback. For example, some research has focused on how to best select which images to present to the user for feedback, while others have investigated how to combine relevance feedback with machine learning techniques to learn more accurate retrieval models.

A key challenge in relevance feedback is the cold start problem, this is where the system has little to no information about a new user's preferences. Researchers have explored techniques to mitigate this, such as using collaborative filtering or transferring knowledge from other users. Another solution is to use context-aware recommender systems that incorporate data like location, time, social information, and user group information to address cold start scenarios (Panda and Ray, 2022).

I have made use of relevance feedback through the addition of a numerical rating mechanism that appears below the recommended images, by rating the images from 1 through 10 the users ‘preferences’ will be added to a vector and taken into account for future recommendations. This vector is dynamically updated and will slowly gain more weight and importance as it is updated, so that the recommendations are catered to the users interests.

A screenshot of a search results page

AI-generated content may be incorrect.

Figure 1: Relevance Feedback Implementation

### Implicit Feedback

Implicit feedback differs from relevance feedback as it infers user preferences from their interactions with the retrieval system, without requiring them to provide explicit judgments. For example, a system might infer that a user finds an image relevant if they click on it, spend a long time viewing it, or save it to a collection.

However, registering all clicks would be counterintuitive as not all clicks will result in a purchase, these ‘noises’ in the implicit feedback are not ideal and so when training a model using implicit feedback you will need to denoise the feedback to improve the efficacy of recommender training (Wang et al., 2021).

Recent research has explored the use of various types of implicit feedback, such as clickthrough data, dwell time, and mouse movements. Furthermore, researchers have developed models that can effectively combine implicit feedback with other sources of information, such as visual features and textual queries, to improve retrieval performance. One area of active research is in developing more robust methods for inferring user intent from implicit signals, as these signals can often be noisy or ambiguous.

Another study has addressed the problem of popularity bias in recommender systems, where popular items are over-represented in recommendations, often at the expense of less popular, long-tail items. They introduce the User Popularity Deviation metric, which the difference between the popularity of the items a user has historically interacted with, and the popularity of the items recommended to that user. This is used to determine whether the recommendations shown to a suer are matched with their past behaviour.

To mitigate this bias, they propose a user-specific calibration method that adjusts the proportion of popular and unpopular items in the recommendations based on the user's past interaction data. For users who have shown a preference for popular items, the system may include more of them in the recommendations. Conversely, for users who prefer less popular items, the system will reduce the weight of their importance in the algorithm. This approach aims to align the popularity of recommendations with individual user preferences, effectively reducing popularity bias from a user-centred perspective (Abdollahpouri et al., 2021).

I have not made use of this approach as a core feature of my system is transparency and ensuring that no data is stored without the user’s consent. No user interaction data or history is retained beyond a single session which makes implementing a feature using implicit feedback unfeasible.

### Explainable AI (XAI)

Explainable AI (XAI) is a field that aims to develop techniques for making AI systems, including image retrieval systems, more transparent and understandable to users. As humans are reticent to adopt any techniques who are not directly interpretable, tractable and trustworthy the demand for ethical and less opaque AI systems rises (Barredo Arrieta et al., 2020).

In the context of image retrieval, XAI can help users understand why certain images are recommended, how the system works, and how to control its behaviour. As users expect personalised recommendations that align with their preferences but also want to understand why those recommendations are made, providing these explanations can enhance user trust, thereby allowing users to make informed decisions, and will result in a more interactive and engaging user experience (Paz-Ruza et al., 2024).

Recent research in XAI for image retrieval has focused on developing methods for visualizing the image features that are most important for retrieval, explaining the relationships between query images and retrieved images, and providing users with tools to interact with the system in a more intuitive way. For example, some systems might highlight the specific parts of an image that are most similar to the query image, or allow users to filter results based on different visual attributes. Another approach involves using natural language explanations to describe the reasons behind an image recommendation, making the system's decision-making process more transparent (Dwivedi et al., 2022).

The development of XAI techniques is crucial for building user trust in image retrieval systems, as it empowers users to understand and control the retrieval process, rather than treating the system as a "black box" as typical deep learning models are with the millions or even billions of parameters of training data. For the purpose of this project the Whitebox model will be selected as it is the best for research-based situations where the algorithm needs to be showcased (Angelov et al., 2021).

Alternatives to the Blackbox level of transparency are the ‘Whitebox’ model which is completely transparent and can directly reveal the features and mechanisms at use and ‘Greybox’ model which is partially transparent, only displaying how changes in features can alter recommendations (Zhou et al., 2024).

Furthermore, XAI can help users to identify potential biases in the system and to provide more meaningful feedback. This can be done by simply displaying a similarity score of a keyword and displayed images, or through visual representations such as bar charts and force graphs. This is particularly useful as humans can process visual information faster and easier compared with textual information (Mouadh Guesmi et al., 2023).

I have done this through the use of both a bar chart to dynamically display the user’s preferences and a force directed graph to display the similarities of the uploaded image to the ones found in the dataset.

A screenshot of a bar chart

AI-generated content may be incorrect.

Figure 2: Bar Chart For User Preference Vector

A screenshot of a graph

AI-generated content may be incorrect.

Figure 3: Force Directed Graph for Similarities Between Uploaded Image And Dataset

# Design

This project aims to create an Imarge Recommendation System, which will adapt to the users’ preferences. In this section of the report I will explain my systems design, the requirements I set for the system to be capable of, as well as the reasoning behind the different design choices I made.

## Requirements

The main requirement of the image recommendation system is to recommend relevant images, this can be broken down into the following points:

* **Image Upload:** The system must allow users to upload an image as a query.
* **Image Retrieval:** The system must retrieve a set of images from a database that are visually and semantically similar to the uploaded image.
* **Efficient Search:** The retrieval process must be efficient to provide timely results.
* **Clear Presentation:** The system must present the retrieved images in a clear and user-friendly interface.
* **User Feedback:** The system should allow users to provide feedback on the relevance of the retrieved images.
* **Algorithm Graphs:** The system should display its workings through graphs to explain how it works to the user so they can more easily trust it.

## Design Choices

To address these requirements, the following design choices were made:

* **Client-Server Architecture:** The system employs a client-server architecture, where the client handles image upload / display, and the server handles image retrieval from the dataset using feature vectors and semantic search.
* **MobileNetV2 for Feature Extraction:** The MobileNetV2 model was chosen for its efficiency in extracting image features without the need to make any API calls, this allows for on-device feature extraction which aligns with the privacy centric goal of this system.
* **Sentence Transformers for Semantic Analysis:** A Sentence Transformer model (all-MiniLM-L6-v2) is used to generate semantic embeddings for keywords, allowing for semantic similarity search via the dataset.
* **Annoy for Approximate Nearest Neighbour Search:** The Annoy library was selected for its efficiency in performing approximate nearest neighbour search in high-dimensional feature spaces.
* **User Interface:** A web-based user interface is used for its accessibility and ease of use, it was designed with a minimalistic style with clear separation between each component, this allows it to be easily understood and allows for future improvements.

## Hardware / Software Requirements

Here is a table outlining the hardware and software I used in the development of this system:

|  |
| --- |
| **Hardware (PC)** |
| CPU: Intel Core I7-8750H 2.20GHz |
| GPU: Nvidia GTX 1060 6GB |
| RAM: 16GB DDR4 2667 MHz |
| HDD: 500GB Storage |

Table 1: Hardware

|  |
| --- |
| **Software (Programs)** |
| IDE: VS Code |
| Programming Languages: Python,  JavaScript, TypeScript, Html, CSS |
| Sprint Planning: Trello |
| Web browser: Google Chrome |
| Version Control: GitHub |
| CI/CD: GitHub Actions |
| Feedback: Google Forms |
| Testing: Postman |
| Feature Extraction Model: MobileNet V2 |
| Semantic Search Model: All-MiniLM-L6-v2 |
| Database: Annoy Index |

Table 2: Software

## System Architecture

The high-level architecture of the system is shown in the following diagram:

A diagram of a computer

AI-generated content may be incorrect.

Figure 4: High Level System Architecture Diagram

The system comprises the following components:

* **Client:** The client-side component, implemented as a web application, provides the user interface for uploading images, viewing results, and providing feedback. It uses JavaScript and a library like React.
* **Server:** The server-side component, implemented using Node.js and Python, handles the image retrieval process. It receives the uploaded image information from the client, performs the search, and sends the results back to the client.
* **Feature Vector Database:** The feature vectors representing the images in the database are stored in an Annoy index for efficient retrieval.

## UML Class Diagram

A diagram of a software system

AI-generated content may be incorrect.

Figure 5: Class Diagram

## Component Design

### Client-Side Design

The client-side interface consists of the following elements:

* **Image Upload Component:** Allows the user to select an image from their local file system.
* **Image Display Area:** Displays the retrieved images in a grid or list format.
* **Feedback Mechanism:** Provides a way for users to rate the relevance of the displayed images (e.g., using star ratings).
* **Graphs:** Provides a layer of transparency to the user so they can understand how the system works.

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 6: Initial UI Design diagram

Below is a screenshot of the working website, It follows the rough initial design diagram.

A screenshot of a website

AI-generated content may be incorrect.

A screenshot of a screenshot of a bar chart

AI-generated content may be incorrect.

Figure 7: Final completed UI design (After Development)

### Server-Side Design

The server-side component is responsible for the following:

* **Receiving Image Data:** Receiving the uploaded image and its associated keyword from the client.
* **Performing Search:** Querying the Annoy index to retrieve similar images.
* **Sending Results:** Sending the file paths of the retrieved images back to the client.

A diagram of a diagram

AI-generated content may be incorrect.

Figure 8: Server-Side Component Diagram

## Data Flow

The following diagram illustrates the data flow within the system:

A diagram of a software process

AI-generated content may be incorrect.

Figure 9: Data Flow Diagram

The data flow can be summarised as follows:

1. The user uploads an image through the client.
2. The client extracts the features from the image using MobileNetV2 and sends the features as well as the top predicted keyword to the server.
3. The server uses the keyword to generate a semantic embedding using the All-MiniLM-L6-v2 Sentence Transformer model.
4. The server queries the Annoy index using the semantic embedding to find similar images within the loaded images in the dataset.
5. The server sends the file paths of the similar images to the client.
6. The client displays the top retrieved images to the user.
7. The user provides their feedback on the relevance of the images, which is sent back to the server.

## Database Design

The system uses an Annoy index to store and efficiently retrieve image feature vectors. The Annoy index contains pre-computed feature vectors for all images in the Tiny ImageNet dataset in the form of a read only text file, this is set up once a script is run and remains unaltered afterwards. An example of this script is available in Appendix C.

## User Interface Design Principles

The user interface is designed with the following principles in mind:

* **Simplicity:** The interface is kept simple and intuitive to minimise the learning curve for users.
* **Clarity:** The retrieved images are presented in a clear and organised manner, making it easy for users to browse the results.
* **Responsiveness:** The system is designed to be responsive, providing timely feedback to user actions.
* **Consistency:** The interface follows a consistent design language to provide a cohesive user experience.

# Implementation

In this section of the report, I will explain the work undertaken during the development of my Image Recommendation Engine as well as the results obtained.

## Project Setup

The project was set up using the following technologies:

* Image Feature Extraction: MobileNetV2 Model, TensorFlow
* Database: Annoy Index (stored as files)
* Similarity Search: Annoy Library (used for approximate nearest neighbours search)
* Client-side: HTML, CSS (for front-end visuals), TypeScript (for functionality)
* Server-side: Node.js, Python
* Semantic Analysis: all-MiniLM-L6-v2 Sentence Transformer Model

The implementation involved setting up the development environments for each of these components, installing the necessary libraries and dependencies, and configuring the communication between the client and server. A virtual environment was used to manage Python dependencies and avoid conflicts between different project requirements.

## Image Feature Extraction and Similarity Search

* **Image Feature Extraction:**
  + The MobileNetV2 model, pre-trained on ImageNet, is used to extract feature vectors from the images. This process is performed on the client-side to reduce server load and improve performance. The client-side code (from index.ts) handles this using TensorFlow.js.
* **Similarity Search:**
  + The Annoy library is used to perform an approximate nearest neighbour search on the pre-computed feature vectors. The Annoy index is built using the build\_annoy\_index.py script (see Appendix C) and loaded into memory on the server-side for fast retrieval. For the purpose of efficiency only 50 images out of 500 per class were loaded into memory for each class totalling 10,000 images.
  + The search function, executed by the Node.js server, takes the semantic embedding of the query keyword and returns the file paths of the most similar images in the dataset. The closest matching label should be understandable as a similar type of image at the very least, for example an image of a wolf may return a few images of various dogs.

## Database Implementation

The image feature vectors are stored in an Annoy index. The index is built offline as a pre-processing step using the build\_annoy\_index.py script. The script performs the following steps:

1. Loads the image data and metadata.
2. Extracts image feature vectors from the images using the pre-trained MobileNetV2 model.
3. Builds an Annoy index with these feature vectors.
4. Saves the Annoy index to a read only file (image\_annoy\_index.ann).
5. Saves the metadata to a JSON file (metadata.json).

During runtime, the Node.js server loads the pre-built Annoy index and metadata into memory for efficient similarity search.

Below is a snippet of the ‘annoy\_neighbors.json’ file. It contains the names of all the images that I have loaded in my index, as stated earlier I have chosen to only load a small portion of the images from each class to make the system more efficient for testing purposes as it is all running locally:

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 10: ‘Annoy\_neighbors.json’ file screenshot

Below is a snippet of the ‘metadata.json’ file. It contains the filename, file path, label (image class name) and the WordNet ID (WNID):

A black background with white lines

AI-generated content may be incorrect.

Figure 11: ‘Metadata.json’ file screenshot

## Client-Side Implementation

The client-side application was implemented as a web application using HTML, CSS, and TypeScript. The key components implemented on the client-side include:

* **Image Upload Component:**
  + This was implemented using the HTML <input type="file"> element. It allows the user to select an image from their local file system to upload for the purpose of receiving similar images from the web application.
* **Image Display Area:**
  + This was implemented as a grid of image elements using HTML and styled with CSS, it dynamically displays the top 6 of the images received from the server.
  + Below this section is a bar chart of the user’s preferences and a force directed graph displaying the links between the uploaded image and those found in the dataset, this is a part of XAI and attempts to make the system more transparent so that the user can understand the impact their interactions with the system has on its workings.
* **Feedback Mechanism:**
  + This was implemented using a rating system as well as a google form. It allows the user to rate the relevance of each displayed image out of 10, this rating will then be taken into account to create a user preference vector that will impact the future recommendations based on favoured image features.

The client-side application handles the user interface, image upload, display of results, and user feedback as well as the bar chart and fore directed graph that explain to the user how the system is working behind the scenes. HTML, CSS, and Typescript are used to create a dynamic and interactive user experience.

## Server-Side Implementation

The server-side implementation consists of two main parts: a Node.js server and a Python server.

### Node.js Server

The Node.js server is responsible for handling communication between the client and the Python server, and for orchestrating the image search process. Its key functionalities include:

* **Receiving The Image Data:**
  + This makes use of the Express.js framework to handle HTTP requests.
  + The uploaded image data and the associated top predicted keyword from the client is received.
* **Sending The Keyword to Python Server:**
  + This sends the keyword to the Python server for semantic embedding generation.
* **Receiving The Search Results:**
  + This receives the file paths of the similar images from the Python server.
* **Sending The Results to Client:**
  + This sends the file paths of the retrieved images back to the client.
* **Performing The Approximate Nearest Neighbours Search:**
  + Here the pre-built Annoy index is loaded for use.
  + The Annoy index is queried with the semantic embedding received from the Python server to find similar images. It attempts to match the keyword with its closest match in the dataset’s labels.

Here's a snippet of the Node.js code for handling the image search request:

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 12: Call to the search images endpoint

This is the call to the search images endpoint; it is used to retrieve a relevant image from the dataset and display it to the user for them to rate.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 13: Node.js search-images endpoint code

This code snippet illustrates how the Node.js server receives the keyword, gets the semantic embedding from the Python server, uses this embedding to query the Annoy index, and then retrieves the corresponding image file paths.

Here's a snippet of the Node.js code for storing the user’s ratings when they consent to their data being stored:

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 14: Call to the consent endpoint

This is the call to the consent endpoint; it is used to send user rating data to the node server to be stored in a Json file for research purposes.

A computer screen shot of text

AI-generated content may be incorrect.

Figure 15: Node.js consent endpoint code

This code snippet illustrates how the Node.js server receives the user ratings data in the form of an array, parses it to Json format and writes it to a file for later use

Below is a screenshot of the ‘user\_ratings.json’ file:

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 16: 'User\_ratings.json' file screenshot

### Python Server

The Python server is responsible for generating semantic embeddings for the keywords using the sentence transformers model. It makes use of the Flask framework to provide a simple API endpoint called ‘/embed’.

* **Uploading of the Keyword:**
  + This stage uses Flask to define an API endpoint that receives the keyword from the Node.js server, this keyword should be generated by the MobileNet V2 model based on the analysis of the users’ uploaded image.
* **Generating The Semantic Embedding:**
  + This stage uses the Sentence Transformers library to generate the semantic embedding for the keyword.
* **Sending Embedding to Node.js Server:**
  + This stage sends the generated embedding back to the Node.js server.

Here is how the Node.js code calls the python function that transforms the highest predicted keywords based on the uploaded image into a semantic embedding:

A computer screen with text

AI-generated content may be incorrect.

Figure 17: Node server's call to the python server to get the semantic embedding

Here's a snippet of the Python code:

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 18: Python server for semantic search

This code snippet shows how the Python server uses Flask to create an API endpoint that receives a text, encodes it into a vector using the Sentence Transformer model, and returns the vector.

A GitHub Actions script was also implemented so that the tests are run on any push to the GitHub repository.

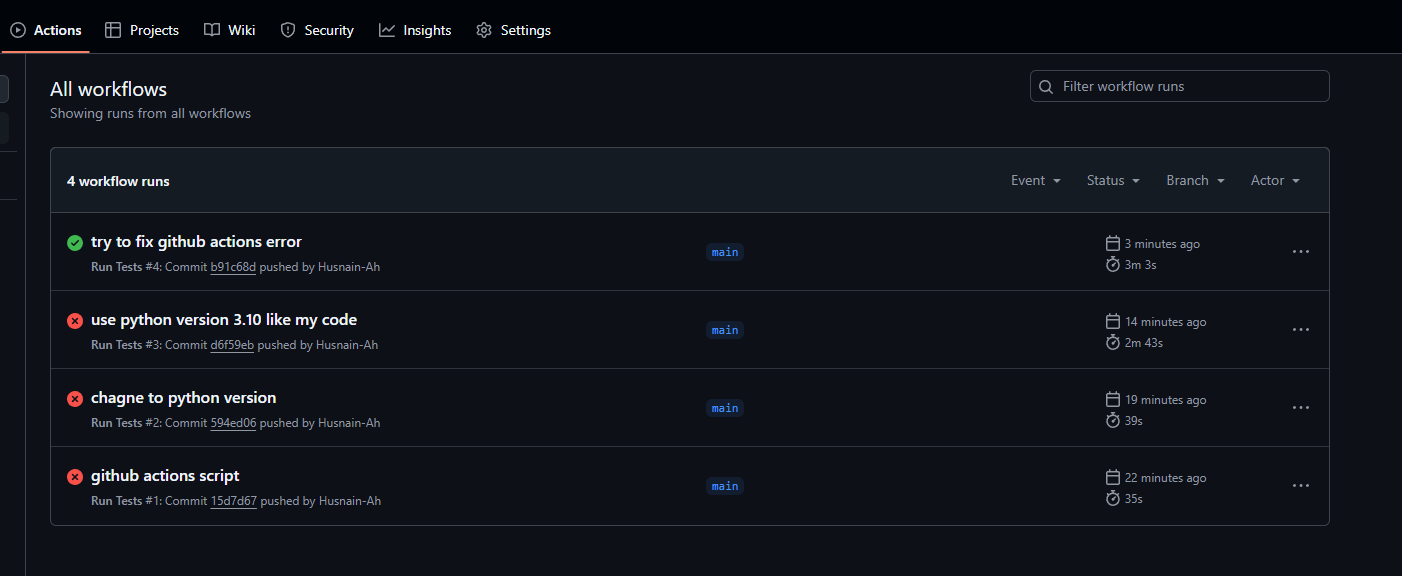


Figure 19: GitHub Actions WorkFlow

## Deployment Of The System

The system was deployed as follows:

* Server-side: The Node.js and Python servers were deployed on separate servers locally.
* Client-side: The web application can be deployed locally.

Each individual server and the front end web applications was run in its own terminal, this is explained in the ReadMe.md file found in my projects GitHub repository (See Appendix A).

In the future this web app could be deployed and hosted for ease of use and testing but for the time being and for the purpose of user security everything will be ran locally to prevent any third parties from affecting the users experience or gaining access to their data.

## Challenges Faced

During implementation, the following challenges were encountered:

* **Cross-Origin Resource Sharing (CORS) Issues:** Initially, there were issues with the web application making requests to the server due to CORS restrictions. This was resolved by configuring the server to allow cross-origin requests in the Server.js file.
* **Python Version Issues:** Due to the Annoy library not being compatible with Python version 3.13.3 that I had pre-installed I had to downgrade to an older version of python, specifically version 3.10. This solved issues with my inverted index not working and allowed me to query my dataset as intended.
* **Asynchronous Operations:** Handling asynchronous operations, such as fetching data from the Python server and updating the UI, required careful use of Promises and async/await in JavaScript.
* **Bad Label Matching:** Initially I was making use of a fuzzy search component (fuse.js) in my algorithm to find a good match for the highest likely predicted keyword that the image recognition model created based on my uploaded image. This however, led to an issue where the keyword ‘Cheeseburger’ was mapped to the ‘Sombrero’ class in my dataset. This was determined as a match because of the word lengths and the position of the individual characters that made up the words. This was not ideal, to remedy this error I switched over to using a semantic search component using the ‘all-MiniLM-L6-v2’ sentence transformer model. This understood the meaning behind the word cheeseburger as being a food item and found the ‘Pretzel’ class in my dataset which more closely matched it.

## Testing

The system was tested using a combination of unit and integration testing, with a focus on both server-side and client-side components. The following testing strategies were employed to ensure the robustness and reliability of the system.

The ImageProcessor class, responsible for image processing in the browser (Client side), was tested using the Jest framework for JavaScript and Typescript.

The key aspects tested include:

* The processImage method throws an error if the MobileNet model is not initialised.
* The processImage method throws an error if the input tensor has invalid dimensions.
* The processImage method returns both predictions and embeddings for a valid image tensor.
* The getKeywordScore method returns 0 if either input label is missing, 1 if they share common terms, and 0 for unrelated terms.
* The calculateImageScore method calculates a score that is a number greater than or equal to 0.
* The calculateImageScore method correctly applies contextual boosting when labels are matched.
* The getSimilarityThreshold method returns the base threshold (0.1) when no ratings are given, increases the threshold with the number of ratings, and caps the threshold at 0.6.

These tests ensure that the uploaded image is processed as intended and that the users ratings are updated dynamically and have an increasing weight on the algorithm as time goes on. Initially they will have a weight of 0.1 (10%) and slowly work up to 0.6 (60%) of the filtering being done by the user preferences and the remainder being filled with content based filtering based on the extracted features from the uploaded image.

**Node.js Server Testing:**

Integration tests were implemented using Jest to validate the behavior of the Node.js server.

The key aspects tested include:

* The /search-images endpoint returns a 400 error when no keyword is provided.
* The /search-images endpoint returns a 500 error when the connection to the Python embedding server fails.
* The cosineSimilarity function returns the correct similarity value for two given vectors.
* The /consent endpoint returns a 200 status and successfully stores user ratings.
* The /consent endpoint returns a 400 error if the ratings data is invalid JSON.

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 20: Jest tests all passed

Integration tests were performed to verify the interaction between different components, such as the client-server communication and the data flow between the Node.js and Python servers. Postman was used to test the API endpoints.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 21: Postman manually testing endpoints

Manual testing was also conducted to ensure the user interface was functional, responsive, and user-friendly. This included testing image upload, image display, and the feedback mechanism.

These tests cover the main functionalities of the system and ensure its robustness and reliability.

**Python Server Testing:**

Unit tests were written using PyTest to ensure the ‘/embed’ endpoint of the Flask application functions correctly.

The key aspects tested include:

* Correct embedding is returned for a given text input.
* Error 400 is returned when no text is provided in the request.
* Error 400 is returned when an empty payload is sent.
* The same embedding is consistently returned for the same text input.
* Different texts result in different embeddings.

These tests ensure that the semantic embedding generation works as expected.

A black screen with text on it

AI-generated content may be incorrect.

Figure 22: PyTest tests all passed

# Evaluation

The primary goal of this project was to develop an image recommendation system that allows users to upload an image and receive visually similar images from a local dataset. This section evaluates the completed work against the initial objectives, considers user feedback, discusses the system's overall achievements, limitations, as well as the legal, societal, and ethical considerations.

## Achievement of Objectives

* + The system successfully implements the core functionality of image-based search. Users can upload an image, and the system retrieves and displays similar images from the Tiny ImageNet 200 dataset.
  + The system effectively integrates several technologies, including MobileNetV2 for feature extraction, the all-MiniLM-L6-v2 Sentence Transformer for semantic analysis, and Annoy for efficient similarity search.
  + The client-server architecture, with Node.js and Python components, functions as intended, facilitating communication and data flow.
  + The system includes a user feedback mechanism, allowing users to rate the relevance of the returned images. This data can be used to improve the system over time.

## Comparison with Requirements

* + The implemented features align well with the project requirements. The system provides an intuitive user interface for image upload and result display.
  + The image search results are generally relevant, demonstrating the effectiveness of the chosen feature extraction and similarity search methods.
  + The system is responsive and performs the search operation in a reasonable time.
  + The use of MobileNetV2 for client-side feature extraction reduces the load on the server.

## User Feedback

* User feedback was collected through ratings of the recommended images out of 10, allowing for quantitative assessment of image relevance. Further qualitative feedback could be gathered to provide more in-depth insights into the system's usability and effectiveness.
  + A deeper analysis of user ratings will be crucial for future improvements, such as refining the similarity search parameters with a larger dataset.
  + The user interface was designed to be intuitive, and initial feedback suggests that users find it easy to upload images and view the results.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 23: Feedback form

Feedback was also received through the above google form, the link for which can be found in Appendix B.

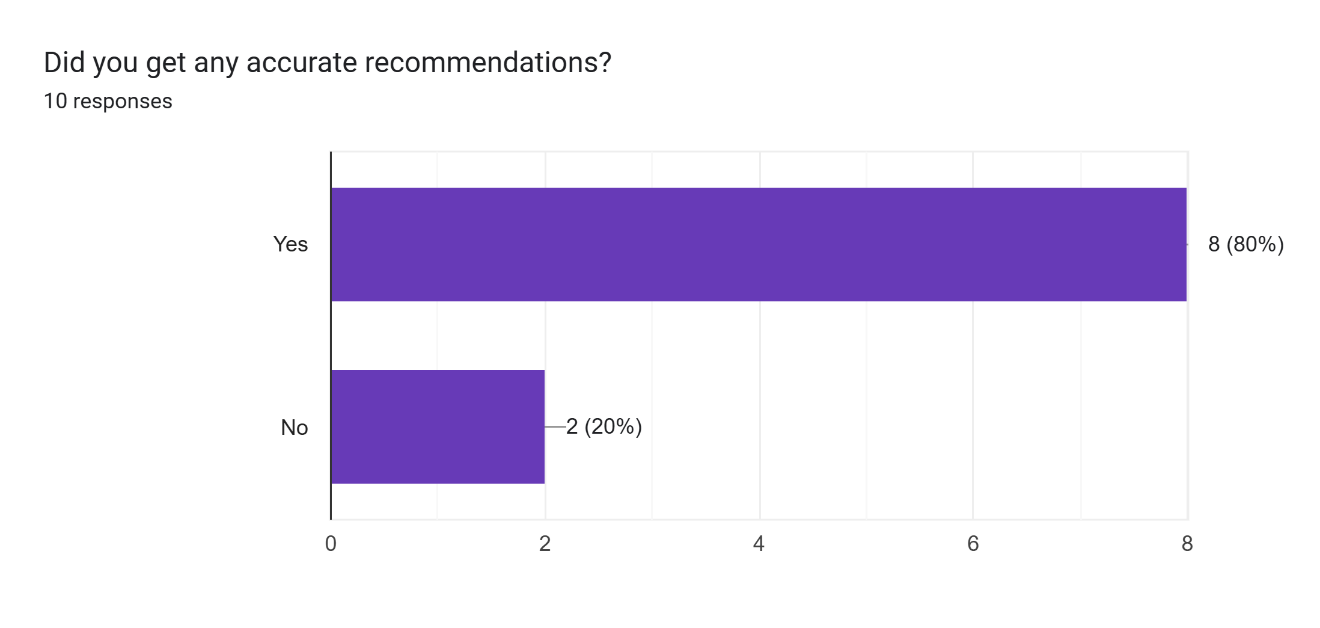


Figure 24: User feedback - Did you get any accurate recommendations?

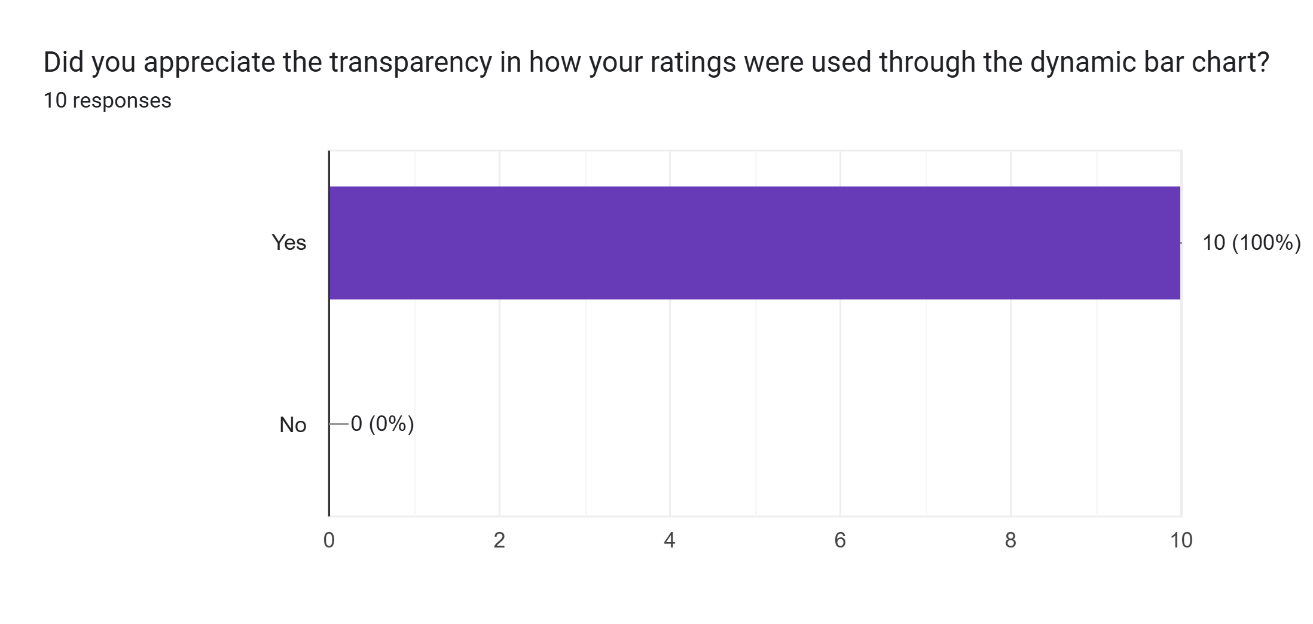


Figure 25: User feedback - Did you appreciate the transparency in how your ratings were used through the dynamic bar chart?

It can be seen from the responses from the feedback from that the users mostly got relevant recommendations and appreciated the transparency in how the system works.

Below is a table introducing the 10 anonymised testing participants and whether they had a positive opinion on the system or not. To keep their identities anonymous no personal information about them was stored.

|  |  |
| --- | --- |
| Participant ID | Positive opinion (Y/N) |
| A | Y |
| B | Y |
| C | Y |
| D | N |
| E | Y |
| F | Y |
| G | Y |
| H | N |
| I | Y |
| J | Y |
| K | Y |

Table 3: Participant Responses

As shown above, 8/10 participants had a positive opinion on the system. This proves that it was mostly working as intended.

## Performance Evaluation

* + The system's performance is primarily determined by the speed and accuracy of the image search.
  + The Annoy library provides efficient approximate nearest neighbour search, enabling fast retrieval of similar images from a large dataset.
  + The accuracy of the search depends on the quality of the image features extracted by MobileNetV2 and the effectiveness of the semantic embeddings generated by the all-MiniLM-L6-v2 Sentence Transformer.
  + Informal user testing shows that the system returns visually similar images the majority of the time, but a more in depth evaluation with a larger number of queries and testing personnel would be ideal.

Below is a violin plot and a bar chart of the user ratings by relevance, if the user rated an image above 5 out of 10 the relevance will be logged as 1, if the rating is below 5 the relevance will be set to 0. Data is only stored if the user gives their consent.

A diagram of a violin plot

AI-generated content may be incorrect.

Figure 26: Violin plot of user ratings by relevance

A graph with blue squares

AI-generated content may be incorrect.

Figure 27: Bar chart of user rating relevance

Through this violin plot and bar chart we can see that more ratings are relevant than not, this is a good sign that the algorithm is returning images that the user finds accurate the majority of the time.

I have also created a line graph of the average ratings over time, this shows the user experience more vividly.

A graph with red and blue lines

AI-generated content may be incorrect.

Figure 28: Line Graph of user rating and relevance over time

Through this graph it can be seen that there is no consistency in recommendations, it was originally expected that the line graph would start with bad recommendations and slowly increase as time went on, but I failed to take into account that the users would input many varying image types, for example they would first input a dog, then a house, then a dragon, then a sunset. This meant that the image features logged in the user preferences varied wildly from each other and did not have a large impact on the recommendations.

Due to this I have also tested for the mean rating value of each relevance group (0 and 1) to get the average rating of all logged user interactions.

A graph with a bar and a number of bars

AI-generated content may be incorrect.

Figure 29: Mean user ratings

This shows that more often than not the image recommendation results are relevant.

To further test the system, I have done Analysis Of Variance (ANOVA) testing and Welch’s two sample T-test.

The ANOVA tests below are used to determine if there are any variations in the user rating data.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 30: ANOVA Testing results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Term** | **Df** | **Sum Sq** | **Mean Sq** | **F Value** | **P Value** |
| **Relevant** | 1 | 151.07 | 151.07 | 79.63 | 2.19e-07 |
| **Residuals** | 15 | 28.46 | 1.90 |  |  |

Table 4: ANOVA Results

As seen above in the ANOVA results, it is shown that a high number of ratings are deemed as relevant as the P value is below 0.05, the threshold for significance.

Welch’s two sample T-test is used to compare the means of two independent groups with differing sample sizes. In this case the relevant and non-relevant ratings.

A computer screen shot of a code

AI-generated content may be incorrect.

Figure 31: Welch's two sample T-test results

|  |  |
| --- | --- |
| **Metric** | **Value** |
| **T-Value** | -9.85 |
| **Degrees of Freedom (Df)** | 14.47 |
| **P-Value** | 8.38e-08 |
| **95% Confidence Interval** | [-7.37, -4.74] |
| **Mean (Not relevant)** | 2.142857 |
| **Mean (Relevant)** | 8.20 |

Table 5: T-test Results

As seen above in the T-test results, it is once again shown that a high number of ratings are deemed as relevant as the P value is below 0.05.

There is also a significant difference between the ratings for the relevant (above 5 points) and non-relevant images. The non relevant images were on average rated around 2 points, which is very low and shows that they did not match the expected result at all. The Relevant images on the other hand were on average rated around 8 points, this is a gap of 6 points.  
From this we can assume that when the image is relevant, it is exceedingly so and matches very closely with the user’s desired recommendation. Whilst when the image is not relevant it is completely off, being below average for the user’s expectations.

## Ethical, Societal, and Legal Considerations

* + Ethical: Data security, fairness, and openness are given top priority in the system's architecture. A diversified dataset helps to reduce algorithmic bias. All calculations are carried out locally on the user's device to secure user privacy, and no user data is saved without express permission. Users have the right to object to the keeping of their data and are notified of what information, if any, is being kept and why. In order to avoid recommending exploitative or manipulative content, the system's reasoning is clear and user-driven, and users have influence over their suggestions through the rating system.
  + Societal: There is recognition of the system's ability to affect user behaviour and patterns of content consumption. Users are exposed to a variety of viewpoints and are not confined to echo chambers because to the system's on-device design.
  + Legal: The GDPR and other relevant regulatory structures were taken into consideration when designing the system. Before any data is handled, users' informed consent is acquired. The system also handles all data locally on the user's device, minimising data collection and storage.

## Limitations and Future Work

* The system's accuracy is limited by the pre-trained MobileNetV2 model and the Tiny ImageNet 200 dataset. Using a fine-tuned model or a larger, more relevant dataset could improve the results.
  + The current implementation relies on pre-computed image features and Annoy indices due to it being the quickest to implement under my current time constraints. Future work could explore online indexing and feature extraction to support real-time updates to the image database or the implementation of a standard vector database like Neo4j or Weaviate.
  + The system could be enhanced to incorporate user preferences more directly, perhaps via allowing for the suer to choose a specific image feature they favour like shape or colour intensity, allowing for better personalised image recommendations.
  + Additional testing and evaluation, including user studies and quantitative metrics, would provide a more comprehensive assessment of the system's performance and usability.
  + Future iterations could use a broader dataset or make calls to the Google Search API to provide a broader and potentially more accurate set of results.

Overall, the image recommendation system achieves its primary objectives and provides a solid foundation for future development. The system has effectively integrated several technologies to deliver a functional and user-friendly image search experience. The design of the system prioritises ethical considerations, user privacy, and control.

Further work, including more rigorous testing using user data, searching for a larger dataset, and a better form of user preference integration, could enhance the system's performance and usability.

# Conclusion

This chapter seeks to reflect on the project as a whole, summarising the work done, acknowledging its limits, and identifying potential areas for future work. The reflections presented provide an assessment of the areas where the project excelled, as well as where it did not meet the goals I set initially. I will also give some suggestions on how any future progression on the system could look.

This chapter will also theorise which features could be added to the product in the future, evaluate their implementation difficulty, user advantages, and the presence of similar functions in existing applications.

## Completed Work

The goal of this project was to develop an on-device image recommendation system that allows users to upload an image and receive visually similar images from a local dataset. This was achieved through the integration of several key technologies.

MobileNetV2 was used for efficient feature extraction on the client side, reducing the load on the server.

The All-MiniLM-L6-v2 sentence transformer model was employed to generate semantic embeddings, enabling a more nuanced understanding of image content.

Annoy was utilised for fast, approximate nearest neighbour search, allowing for efficient retrieval of similar images from the dataset.

A client-server architecture, built with Node.js, Python and Typescript, facilitated seamless communication and data flow between the different components. Furthermore, the system has implemented a user feedback mechanism, allowing users to rate the relevance of the returned images, which has an impact on future recommendations.

In alignment with the research direction of "Empowering Users Through Transparent and Controllable On-Device Image Recommendation" this project has successfully delivered an image recommendation system that differs from traditional non transparent, third-party solutions. The system performs all processes and computations locally and uses a feedback-driven interface that implements elements of XAI through graphs and charts to explain how the algorithm works, the system successfully empowers users with greater control over the image recommendation process.

Specifically, the project addressed the research questions in the following ways:

* **Interaction & Attention Focus:** The image recommendation interface was designed to reshape users' interactions with visual content by providing real-time, personalised recommendations. The user feedback mechanism allows users to actively participate in the curation process, influencing future recommendations and potentially increasing their sense of control.
* **Perceived Value and Intrusiveness:** The system's on-device nature and transparent design result in a working algorithm that meets the intended goal whilst also minimising intrusiveness as the user is aware of all data used. Through avoiding any reliance on external, or third party, servers and giving users control over the recommendations through the rating function, the system seeks to be as helpful as possible whilst also taking the context of then uploaded image into account.
* **Trust, Transparency & User Control:** The project has prioritised these aspects by implementing transparent keyword-based recommendations, visible to the user through the predicted keywords section where the top 3 keywords as well as the percentage of similarity to the uploaded image are shown as well as by incorporating user ratings and ensuring that all computations and processes of images and data are performed locally on the host device.

## Limitations

Despite the successful implementation of the system’s core functionalities, the project encountered several limitations.

One of the significant limitations was the reliance on the ‘Tiny ImageNet 200’ dataset which was picked due to its small size with efficiency in mind, with only 200 classes the scope of the recommended images and the diversity of the image types was severely limited. The images from the dataset only included real objects, so inputting any images that don’t exist in the real world such as a dragon resulted in bad recommendations.

Another limitation is the use of the pre-trained MobileNetV2 model, though efficient, lightweight and suitable for this project where the purpose was to show the principle of image recommendation algorithms and their effect on user trust, a production ready version of the system could leverage the use of more advanced models which offer greater image feature extraction capabilities at the cost of efficiency and resource management. Similarly to the dataset, this model only excels in recognising real world objects, something like a dragon would return the predicted keyword of ‘fish’ or ‘barracuda’.

Additionally, the system's reliance on pre-computed image features and Annoy indices means that the image database cannot be updated in real-time. This can be remedied through the implementation of a traditional vector database.

## Future Work

In the future I can make several possible improvements to the system such as expanding the dataset, this would significantly improve the diversity and accuracy of the search results due to the larger library of images and classes available.

Alternatively, this could be achieved by making calls to the Google Search API, which would allow images to be retrieved from google images, though these images may not use the same ‘Labels’ and ‘WNID’ elements in their metadata that my system references in its code so changes will need to be made taking this into account.

Instead of MobileNet V2, a more advanced feature extraction model could be used to improve the accuracy of the image similarity comparisons, though as stated earlier this would come at the cost of additional required processing power if the future version of the system still does all computations locally instead of on the cloud to protect user privacy.

The Annoy index database could be swapped out for a standard vector database like Neo4j or Weaviate to improve the scalability of the system in preparation for future expansions to the system.

Furthermore, I would explore online indexing and feature extraction techniques, this would allow for the system to support real-time updates to the image database. A more interactive form of using the user preferences, such as having the user refine the returned images based on an entered string like “I would like the recommended images to all be taken in low light conditions at night” could allow for a more personalised and user-centric experience and improve both the accuracy and relevancy of recommended images. The effectiveness of this change can be seen through the image generation functions of AI chatbot models such as Microsoft’s Copilot or OpenAI’s ChatGPT, which can fine tune the generated images based on user input. I believe that incorporating this into the image recommendation algorithm will result in greater accuracy and user satisfaction rates.

After implementing these future changes to the system, it will be necessary to perform additional testing and evaluation metrics, including user studies through feedback forms and quantitative metrics such as Regression Analysis and Root Mean Squared Error tests against a ground data set, to provide a thorough assessment of the new system's capabilities.

These additions would benefit the end user by making the retrieved images more accurate and catered to them as an individual, implementing them would not be difficult but under the current time constraints they were not done and were removed from the project’s scope.

The Documentation of the system could also be improved with a swagger page for the endpoints being created.

Some suggestions from the user feedback below could also be taken into account and implemented.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 32: User suggestions for improvements from the feedback form

## Hindsight

Reflecting on the project as a whole, greater emphasis on the initial design phase would have been ideal and would have undoubtedly led to a more coherent and robust final product.

Allowing for a larger time frame would have also led to a larger more expansive system that better met the requirements.

Focusing less on the theme of transparency and user privacy through local computation and using cloud computing functions may have allowed for a better recommendation system, though at the cost of user trust and efficiency.

## Closing Remarks

To summarise, this dissertation paper has covered the creation of an on device, lightweight and transparent image recommendation system that was designed with the core theme of explainable AI and transparency for the purpose of fostering user trust. I believe that it has met this goal. However, by taking into account the limitations met and the suggestions for future work it could be enhanced.

References

1. Hwang, K. H., Lee, H. and Choi, D. (2012) ‘Medical Image Retrieval: Past and Present.’ Healthcare Informatics Research, 18(1) p. 3.
2. What is the ‘semantic gap’ problem and how does semantic search address it? (2025) Milvus.io. [Online] [Accessed on 29th April 2025] https://milvus.io/ai-quick-reference/what-is-the-semantic-gap-problem-and-how-does-semantic-search-address-it. ‌
3. Li, X., Yang, J. and Ma, J. (2021) ‘Recent developments of content-based image retrieval (CBIR).’ *Neurocomputing*, 452, September, pp. 675–689.
4. Hong, R., Li, L., Cai, J., Tao, D., Wang, M. and Tian, Q. (2017) ‘Coherent Semantic-Visual Indexing for Large-Scale Image Retrieval in the Cloud.’ *IEEE Transactions on Image Processing*, 26(9) pp. 4128–4138.
5. ‌ Hameed, I. M., Abdulhussain, S. H. and Mahmmod, B. M. (2021) ‘Content-based image retrieval: A review of recent trends.’ Pham, D. T. (ed.) Cogent Engineering, 8(1) p. 1927469. ‌
6. Piras, L. and Giacinto, G. (2017) ‘Information fusion in content based image retrieval: A comprehensive overview.’ Information Fusion, 37, September, pp. 50–60. ‌
7. Suganyadevi, S., Seethalakshmi, V. and Balasamy, K. (2021) ‘A review on deep learning in medical image analysis.’ International Journal of Multimedia Information Retrieval, 11(1) pp. 19–38. ‌
8. Tzelepi, M. and Tefas, A. (2018) ‘Deep convolutional learning for Content Based Image Retrieval.’ Neurocomputing, 275, January, pp. 2467–2478. ‌
9. Rubner, Y., Tomasi, C. and Guibas, L. J. (1998) ‘A metric for distributions with applications to image databases.’ International Conference on Computer Vision, January. ‌
10. Wang, W., Feng, F., He, X., Nie, L. and Chua, T.-S. (2021) ‘Denoising Implicit Feedback for Recommendation.’ Proceedings of the 14th ACM International Conference on Web Search and Data Mining, March. ‌
11. Angelov, P. P., Soares, E. A., Jiang, R., Arnold, N. I. and Atkinson, P. M. (2021) ‘Explainable artificial intelligence: an analytical review.’ WIREs Data Mining and Knowledge Discovery, 11(5). ‌
12. Dwivedi, R., Dave, D., Naik, H., Singhal, S., Rana, O., Patel, P., Qian, B., Wen, Z., Shah, T., Morgan, G. and Ranjan, R. (2022) ‘Explainable AI (XAI): Core Ideas, Techniques and Solutions.’ ACM Computing Surveys, 55(9). ‌
13. Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R. and Herrera, F. (2020) ‘Explainable Artificial Intelligence (XAI): Concepts, taxonomies, Opportunities and Challenges toward Responsible AI.’ Information Fusion, 58(1) pp. 82–115. ‌
14. Fernández, M., Cantador, I., López, V., Vallet, D., Castells, P. and Motta, E. (2011) ‘Semantically enhanced Information Retrieval: An ontology-based approach.’ Journal of Web Semantics, 9(4) pp. 434–452. ‌
15. Yin, C. and Zhang, Z. (2024) ‘A Study of Sentence Similarity Based on the All-minilm-l6-v2 Model With “Same Semantics, Different Structure” After Fine Tuning.’ Advances in computer science research. Atlantis Press, January, pp. 677–684. ‌
16. Stankevičius, L. and Lukoševičius, M. (2024) ‘Extracting Sentence Embeddings from Pretrained Transformer Models.’ Applied Sciences, 14(19) p. 8887. ‌
17. Hovy, D. and Prabhumoye, S. (2021) ‘Five sources of bias in natural language processing.’ Language and Linguistics Compass, 15(8). ‌
18. Li, Y., Luo, J. and Mei, T. (2014) ‘Personalized image recommendation for web search engine users.’ 2022 IEEE International Conference on Multimedia and Expo (ICME), July, pp. 1–6. ‌
19. Zhang, J., Yang, Y., Tian, Q., Zhuo, L. and Liu, X. (2017) ‘Personalized Social Image Recommendation Method Based on User-Image-Tag Model.’ IEEE Transactions on Multimedia. Institute of Electrical and Electronics Engineers (IEEE), 19(11) pp. 2439–2449. ‌
20. Mouadh Guesmi, Mohamed Amine Chatti, Shoeb Joarder, Qurat Ul Ain, Siepmann, C., Hoda Ghanbarzadeh and Rawaa Alatrash (2023) ‘Justification vs. Transparency: Why and How Visual Explanations in a Scientific Literature Recommender System.’ Information. Multidisciplinary Digital Publishing Institute, 14(7) pp. 401–401. ‌
21. Paz-Ruza, J., Alonso-Betanzos, A., Guijarro-Berdiñas, B., Cancela, B. and Eiras-Franco, C. (2024) ‘Sustainable transparency on recommender systems: Bayesian ranking of images for explainability.’ Information Fusion. Elsevier BV, 111, May, pp. 102497–102497. ‌
22. Yong Rui, Huang, T. S., Ortega, M. and Mehrotra, S. (1998) ‘Relevance feedback: a power tool for interactive content-based image retrieval.’ IEEE Transactions on Circuits and Systems for Video Technology, 8(5) pp. 644–655. ‌
23. Abdollahpouri, H., Mansoury, M., Burke, R., Mobasher, B. and Malthouse, E. (2021) ‘User-centered Evaluation of Popularity Bias in Recommender Systems.’ Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization, June.
24. Panda, D.K. and Ray, S. (2022). Approaches and algorithms to mitigate cold start problems in recommender systems: a systematic literature review. Journal of Intelligent Information Systems. doi:https://doi.org/10.1007/s10844-022-00698-5. ‌
25. Zhou, Y., Wang, H., He, J. and Wang, H. (2024) ‘Review-Based Explainable Recommendations: A Transparency Perspective.’ ACM Transactions on Recommender Systems, October. ‌

Bibliography

<https://www.pluralsight.com/courses/ng-conf-19-machine-learning-tensorflow-js>

<https://www.linkedin.com/learning/software-design-modeling-with-uml>

<https://r-graph-gallery.com/violin.html>

<https://r-graph-gallery.com/barplot.html>

<https://r-graph-gallery.com/line-plot.html>

<https://statsandr.com/blog/anova-in-r/>

<https://www.geeksforgeeks.org/welchs-t-test-in-r/>

Appendices

Appendix A:

GitHub Repository link

<https://github.com/Husnain-Ah/Image_recommendation_Engine_Project>

Appendix B:

Google Form

<https://docs.google.com/forms/d/1rdHq5CUnQzQ2tWxYdh-uVJGqHk4x8CqV-rzn6wHNDQM/>

Appendix C:

build\_annoy\_index.py script

This is a one time run script that creates an inverted index that maps the images in the dataset to labels so they can be queried and searched for.

import os

import numpy as np

import tensorflow as tf

from annoy import AnnoyIndex

from PIL import Image

import json

BASE\_DIR = 'C:/Users/Husnain/Desktop/git for uni work/Image\_recommendation\_Engine\_Project/tiny-imagenet-200/'

TRAIN\_DIR = os.path.join(BASE\_DIR, 'train')

WORDS\_FILE = os.path.join(BASE\_DIR, 'words.txt')

FEATURE\_DIM = 1280

TOP\_K = 10

MAX\_IMAGES\_PER\_CLASS = 50 # only load 50 pics for now to cut dowen load time in developmnet (change if load times are too long)

label\_map = {}

with open(WORDS\_FILE, 'r') as f:

    for line in f:

        wnid, label = line.strip().split('\t')

        label\_map[wnid] = label

model = tf.keras.Sequential([

    tf.keras.applications.MobileNetV2(include\_top=False, weights='imagenet', input\_shape=(224, 224, 3)),

    tf.keras.layers.GlobalAveragePooling2D()

])

model.trainable = False

def preprocess\_image(img\_path, target\_size=(224, 224)):

    img = Image.open(img\_path).resize(target\_size).convert("RGB")

    img = np.array(img) / 255.0

    return img

def extract\_embeddings(image\_paths):

    embeddings = []

    metadata = []

    for img\_path in image\_paths:

        img = preprocess\_image(img\_path)

        img = np.expand\_dims(img, axis=0)

        embedding = model.predict(img, verbose=0).flatten()

        parts = img\_path.split(os.sep)

        wnid = parts[-3]

        filename = parts[-1]

        label = label\_map.get(wnid, wnid)

        embeddings.append(embedding)

        metadata.append({

            "filename": filename,

            "path": img\_path,

            "label": label,

            "wnid": wnid

        })

    return np.array(embeddings), metadata

image\_paths = []

for wnid in os.listdir(TRAIN\_DIR):

    wnid\_dir = os.path.join(TRAIN\_DIR, wnid, 'images')

    if not os.path.isdir(wnid\_dir):

        continue

    image\_files = [f for f in os.listdir(wnid\_dir) if f.endswith('.JPEG')][:MAX\_IMAGES\_PER\_CLASS]

    for file in image\_files:

        image\_paths.append(os.path.join(wnid\_dir, file))

print(f"Processing {len(image\_paths)} images...")

embeddings, metadata = extract\_embeddings(image\_paths)

annoy\_index = AnnoyIndex(FEATURE\_DIM, 'angular')

for i, embedding in enumerate(embeddings):

    annoy\_index.add\_item(i, embedding)

annoy\_index.build(10)

annoy\_index.save("annoy\_data/image\_annoy\_index.ann")

neighbors = {}

for i in range(len(embeddings)):

    indices = annoy\_index.get\_nns\_by\_item(i, TOP\_K + 1)

    current\_file = metadata[i]['filename']

    neighbors[current\_file] = [

        metadata[idx]['filename'] for idx in indices if metadata[idx]['filename'] != current\_file

    ]

with open('annoy\_data/annoy\_neighbors.json', 'w') as f:

    json.dump(neighbors, f, indent=2)

with open('annoy\_data/metadata.json', 'w') as f:

    json.dump(metadata, f, indent=2)

print("Annoy index and metadata saved successfully.")

Appendix D:

Consent Form

A close-up of a form

AI-generated content may be incorrect.

Appendix E:

Participant Information Sheet

A close-up of a questionnaire

AI-generated content may be incorrect.

A white paper with black text

AI-generated content may be incorrect.

A paper with text on it

AI-generated content may be incorrect.

A screenshot of a document

AI-generated content may be incorrect.

Appendix F:

Feasability study

A blue background with black text

AI-generated content may be incorrect.

A close-up of a blue and white page

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A blue and white page with black text

AI-generated content may be incorrect.

A blue and white document with text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A blue and white document with black text

AI-generated content may be incorrect.

A blue and white text on a page

AI-generated content may be incorrect.

A blue and white information page

AI-generated content may be incorrect.

A close-up of a text

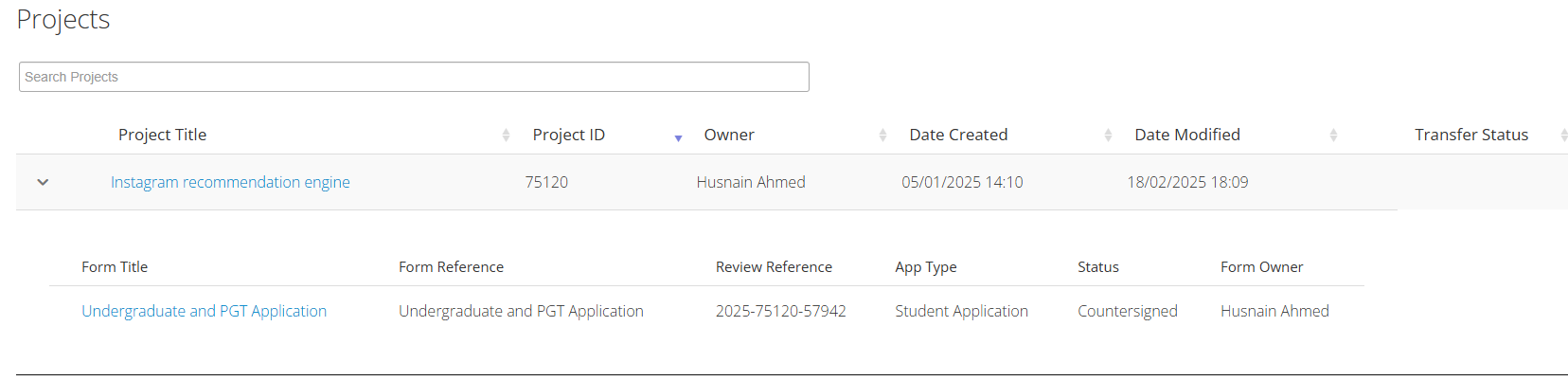
AI-generated content may be incorrect.

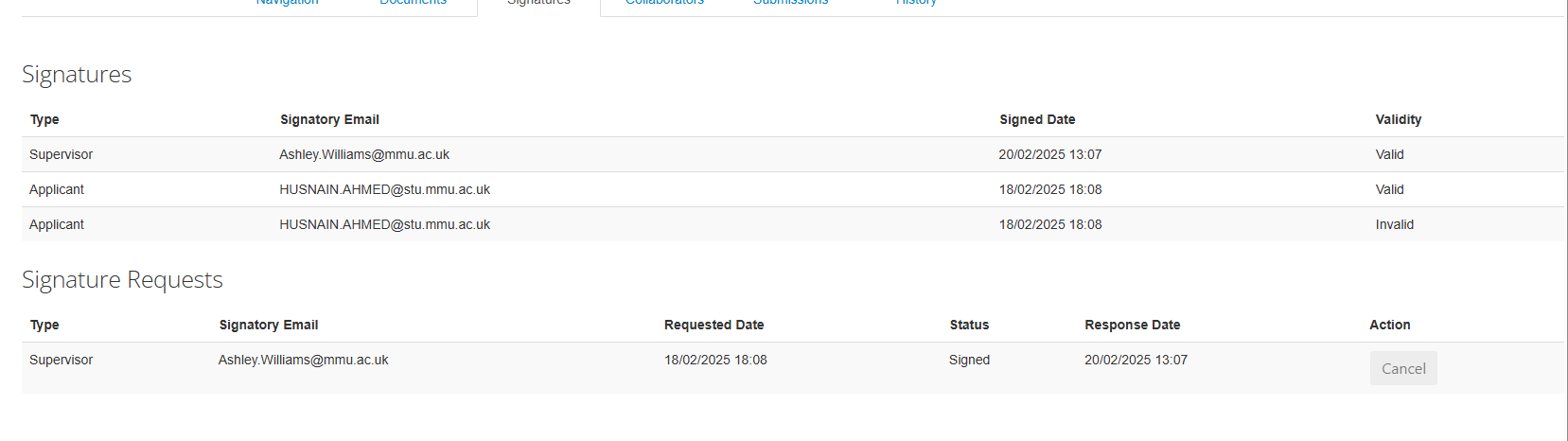
A black and blue logo

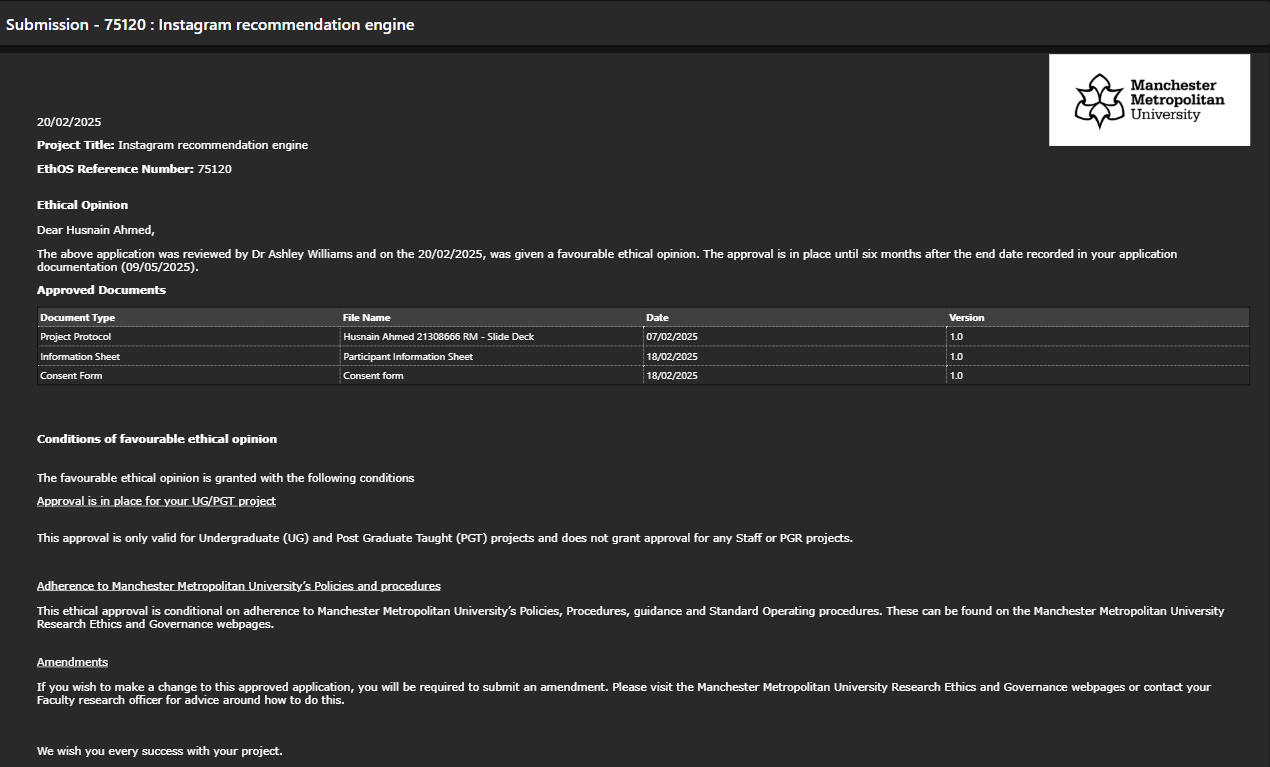
AI-generated content may be incorrect.

Appendix E:

Ethos Form proof







Appendix F:

Project showcase presentation slides

A blue rectangular object with black text

AI-generated content may be incorrect.

A screenshot of a computer project

AI-generated content may be incorrect.

A screenshot of a questionnaire

AI-generated content may be incorrect.

A close-up of a project

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A white and blue rectangle with black text

AI-generated content may be incorrect.

A blue and white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A black and blue logo

AI-generated content may be incorrect.

Link to presentation video:

hghgkhjk