CONTENT BASED RECOMMENDER SYSTEM FOR ONLINE STORES USING EXPERT SYSTEMS

A MINI PROJECT REPORT

Submitted by

K. Srikanth

(20841A6642)

B. Prashanth

(20841A6639)

S. Sainath Guptha

(20841A6630)

Guided by:

Mr. Atta Rahman Sofi

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

AURORA'S TECHNOLOGICAL & RESEARCH INSTITUTE

(Affiliated to JNTU, Hyderabad and Accredited by NAAC with 'A' grade)

Parvathapur, Uppal, Hyderabad-500039

DECEMBER, 2023

AURORA'S TECHNOLOGICAL & RESEARCH INSTITUTE

(Affiliated to JNTU, Hyderabad and Accredited by NAAC with 'A' grade)

Parvathapur, Uppal, Hyderabad-500 039

DECLARATION

We hereby declare that the work described in this project, entitled **CONTENT BASED RECOMMENDER SYSTEMS FOR ONLINE STORES USING EXPERT SYSTEMS** which is being submitted by us in partial fulfilment for the award of Bachelor of Technology in Computer Science and Engineering to **AURORA'S TECHNOLOGICAL & RESEARCH INSTITUTE** is the result of investigation carried out by us under the guidance of **Mr. Atta Rahman Sofi, Associate Professor, CSE.**

The work is original and has not been submitted for any degree this orany other university.

Place: Hyderabad

Date:

k. Srikanth(20841A6642)

B. Prashanth(20841A6639)

S. Sainath Gupta(20841A6630)

CERTIFICATE

Certified that this project report **CONTENT BASED RECOMMENDER SYSTEMFOR ONLINE STORES USING EXPERT SYSTEM** is bonified work **K. SRIKANTH (20841A6642), B. PRASHANTH(20841A6642), S. SAINATHGUPTA (20841A6630)** who carried out the project work under our supervision.

GUIDE

Mr. Atta Rahman Sofi
Associate Professor
Dept. of CSE

COORDINATOR

Mr. Krishna Rao
Dept. of CSE

HEAD OF THE DEPARTMENT

Mrs. A. Durga Pavani
Dept. of CSE

PRINCIPAL

Dr. A. Mahesh Babu

EXTERNAL EXAMINER

ACKNOWLEDGMENT

This work was done during the project period, and it was a very good opportunity to put theoretical knowledge into planned exercise with an aim to solve a real time problem and to develop confidence to face various practical situations.

We convey thanks to our project guide **Mr. Atta Rahman Sofi**, Department of Computer Science and Engineering, for providing encouragement, constant support, and guidance, which was of great help to complete this project successfully.

We express our sincere thanks to Project Coordinator **Mr. Ch Krishna Rao**, for helping us to complete our project work by giving valuable suggestions.

We would like to express our immense gratitude to **Mrs. A. Durga Pavani**, Head of the Department CSE, for her support, valuable suggestions, and kind attentionto us throughout the project.

We would like to express our sincere thanks to **Dr. A. Mahesh Babu,** Principal,Aurora's Technological and Research Institute for providing us with a congenial atmosphere and encouragement.

Finally, we would also like to thank the people who have directly or indirectly helped me, my parents, and my friends for their cooperation in completing the TechnicalSeminar work

K.Srikanth(20841A6642)

B.Prashanth(20841A6639)

S.Sainath Gupta(20841A6630)

ABSTRACT

The Content-Based Recommender System for Online Stores Using Expert Systems is an intelligent and personalized application designed to enhance the shopping experience for online customers by providing tailored product recommendations. Combining the principles of content-based recommendation and expert systems, the proposed system analyzes user preferences, product attributes, and expert knowledge to offer relevant and appealing product suggestions.

During the recommendation process, the system employs a content-based approach to analyze the user's past purchase history, browsing behavior, and expressed preferences. By comparing these user-specific attributes with the attributes of available products and applying system knowledge, the system generates personalized recommendations.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Overview

Content-Based Recommendation: This approach recommends items based on their attributes and the user's historical interactions. For online stores, this might include factors like product categories, descriptions, user reviews, and specifications.

Expert Systems: Expert systems are AI systems that emulate human expertise in a specific domain. In the context of online shopping, expert systems can leverage knowledge from experts in various product categories, such as fashion, electronics, or home decor. This knowledge can include style trends, technical specifications, compatibility, and more.

Integration: Combining content-based recommendation techniques with expert knowledge enhances the system's ability to provide personalized and informed suggestions. Expert systems can guide the recommendation engine in understanding nuanced user preferences.

Personalization: By analyzing a user's past interactions and preferences, a content-based recommender system with expert system integration can offer highly personalized recommendations. For instance, it can suggest clothing styles that align with a user's fashion taste or compatible accessories for a purchased product.

Complex Decision Support: In scenarios where product knowledge is extensive or specialized (e.g., high-end electronics or medical equipment), an expert system can assist users in making informed decisions by providing explanations and comparisons.

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 Software requirements

Windows 11 is the latest operating system developed by Microsoft. It features a redesigned user interface, with a centered Start menu and taskbar, rounded corners, and various other visual enhancements. Windows 11 provides a user-friendly environment for development and supports a wide range of applications and programming tools.

Integrated Development Environment (IDE): PyChar

PyCharm is a popular integrated development environment (IDE) specifically designed for Python development. Developed by JetBrains, PyCharm provides a rich set of features such as intelligent code completion, debugging, version control integration, and support for various web frameworks. It is widely used by Python developers for its robust set of tools that enhance productivity and code quality.

Programming Language: Python

Python is a high-level, interpreted programming language known for its readability and simplicity. It supports multiple programming paradigms and has a vast ecosystem of libraries and frameworks. Python is widely used for web development, data analysis, artificial intelligence, machine learning, and more. Its clean syntax makes it a preferred language for both beginners and experienced developers.

Libraries: scikit-learn (sklearn), NLTK (Natural Language Toolkit)

scikit-learn (sklearn): Scikit-learn is a powerful machine learning library for Python. It provides simple and efficient tools for data analysis and modeling, including classification, regression, clustering, and dimensionality reduction. Scikit-learn is widely used in the field of machine learning and data science for its ease of use and comprehensive documentation.

NLTK (Natural Language Toolkit): NLTK is a library for working with human language data. It includes various modules for tasks such as tokenization, stemming,

tagging, parsing, and more. NLTK is a valuable tool for natural language processing (NLP) tasks and is commonly used for text analysis and linguistic research.

Graphical User Interface (GUI):

Streamlit is a Python library for creating web applications with minimal effort. It is particularly well-suited for creating data-driven applications and interactive dashboards. With Streamlit, you can turn data scripts into shareable web apps without extensive web development knowledge. It simplifies the process of creating engaging and interactive user interfaces for data analysis and machine learning applications.

2.2 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

Architecture: All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture.

Processing power: The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this

processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

Secondary Storage: Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

Display Adapter: Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

Peripherals: Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

Processor: Dual-core processor (2.0 GHz or higher).

RAM: 4 GB RAM.

Storage: 100 GB HDD or SSD.

Network: Wi-Fi connectivity.

CHAPTER 3

LITERATURE SURVEY

3.1 Title: “Item-Based Collaborative Filtering Recommendation Algorithms”

Author: Badrul Sarwar, et al.i (2001)

Description: The paper presents an item-based collaborative filtering approach to recommendation systems. Collaborative filtering is a technique used to make automatic predictions (filtering) about user preferences by collecting information from many users (collaborating). In this approach, user-item interactions are analyzed to identify patterns and similarities among items that users have interacted with. These patterns are then used to make personalized recommendations to users.

Background and context:

• Badrul Sarwar's work in content-based recommender systems focuses on improving online shopping experiences.

• Online stores often struggle with personalized product recommendations for users.

• The research aims to address this challenge by utilizing expert systems.

• expert systems utilize knowledge from human experts to solve complex problems.

Theoretical Framework:

• Content-based recommender systems rely on the analysis of item features and user profiles to make recommendations.

• Expert systems incorporate knowledge from domain experts to enhance recommendation accuracy.

• Sarwar's work likely combines these principles to create a robust recommendation system.

Research Gap Identification:

• Sarwar's research likely identifies gaps in current online store recommendation systems.

• These gaps may include limited personalization, low accuracy, or challenges in adapting to evolving user preferences.

• While the concept of integrating expert systems into content-based recommender systems is promising, there's a need for more research on the practical implementation challenges

Methodology:

• The methodology may involve data collection, feature extraction, and user profiling.
• It could incorporate expert knowledge to improve recommendation algorithms.

• Evaluation metrics like precision, recall, and user satisfaction might be used to assess system performance.

• Set specific goals, such as enhancing user engagement, click-through rates, or conversions through personalized image-based recommendations.

Main themes and trends:

• Key themes include personalization, user modeling, and knowledge integration from experts.

• Trends may involve the use of machine learning, natural language processing, and deep learning techniques.

• Pre-trained CNN models, such as VGG, ResNet, and Inception, were commonly used for image feature extraction.

• The integration of content-based image analysis with collaborative filtering or other recommendation techniques to create hybrid recommender systems.

Synthesis and Analysis:

• This section would likely summarize the results of the research.

• It may present findings related to recommendation accuracy, user satisfaction, and the impact of expert knowledge

Discussion:

• The discussion section may interpret the results in the context of the research goals.

• It might explore the implications of the findings for online retailers and users.

• Potential limitations of the research may also be discussed.

Conclusion:

• The conclusion would provide a summary of the key findings.

• It may reiterate the significance of incorporating expert systems in content-based recommendation.

• Future research directions or practical applications might also be mentioned.

3.2 Title:“Matrix Factorization Techniques for Recommender Systems”

Author: Yehuda Koren, Robert Bell, and Chris Volinsky (2009)

Description: The paper addresses the challenge of improving recommendation accuracy in collaborative filtering-based recommendation systems. Collaborative filtering relies on user-item interaction data to make recommendations, and matrix factorization techniques are proposed as a way to model this data effectively. The authors focus on the Netflix Prize dataset, a largescale real-world dataset, to demonstrate the effectiveness of their approach.

Background and Context:

• The paper likely discusses the challenges faced by recommender systems in providing accurate recommendations.

• It may touch upon the increasing importance of recommendation systems in various online platforms.

Theoretical Framework:

• The authors might introduce matrix factorization as a key theoretical concept in building recommender systems.

• They may discuss how matrix factorization can be applied to collaborative filtering, a common recommendation technique.

Research Gap Identification:

• The paper may identify limitations or challenges in existing recommender systems, particularly in collaborative filtering.

• It could highlight the need for more effective techniques to handle sparse data or improve recommendation quality.

Methodology:

• Details about the matrix factorization techniques used in the research, such as Singular Value Decomposition (SVD) or Alternating Least Squares (ALS), may be described.

• Data preprocessing and model training methods may also be outlined.

Main Theme and Trends:

• Main themes may include matrix factorization, collaborative filtering, and dimensionality reduction.

• Trends might involve the incorporation of implicit feedback or hybrid recommendation systems.

Synthesis and Analysis:

• This section could present experimental results, showing how matrix factorization techniques improve recommendation accuracy.

• Comparison with other recommendation methods might be included to demonstrate the effectiveness of the proposed approach.

Discussion:

• The discussion section may analyze the implications of the findings.

• It could address the strengths and limitations of matrix factorization in recommender systems.

• Practical considerations for implementing these techniques may also be discussed.

Conclusion:

• The conclusion would summarize the key contributions of the paper.

• It may reiterate the importance of matrix factorization in improving recommendation systems.

• Future research directions in this area might also be suggested

3.3 Title: “A Fast Parallel SGD for Matrix Factorization in Shared Memory Systems”

Author: Yifan Hu, et al. (2015)

Description: The paper addresses the need for scalable and efficient matrix factorization techniques for collaborative filtering-based recommendation systems. It introduces parallel stochastic gradient descent (SGD) algorithms designed to optimize matrix factorization models efficiently, leveraging shared memory systems. The focus is on improving the training speed of recommendation models while maintaining recommendation quality.

Background and Context:

• Matrix factorization is a fundamental technique used in various applications, including recommendation systems, image processing, and data compression.

• It involves decomposing a large matrix into smaller matrices to extract meaningful patterns or reduce dimensionality. performing matrix factorization on massive datasets poses significant computational challenges, often requiring time-consuming computations.

• The paper addresses the need for efficient matrix factorization techniques, particularly in shared memory systems.

Theoretical Framework:

• The paper builds upon the theoretical foundations of Stochastic Gradient Descent (SGD), a widely used optimization technique for minimizing loss functions.

• SGD iteratively updates model parameters to find the optimal factorized representation of a matrix.

• The key idea behind SGD is to approximate the gradient of the loss function using a random subset of the training data, making it computationally efficient.

Research Gap Identification:

• One of the primary challenges in matrix factorization is the need for scalable and parallelizable algorithms, especially when dealing with large datasets.

• Existing techniques often struggle to achieve high performance in shared memory systems, where multiple processors work collaboratively.

• This paper identifies the research gap in developing a fast and parallelizable SGD algorithm tailored for shared memory systems.

Methodology:

• The proposed methodology in the paper introduces a novel approach to matrix factorization using a modified SGD algorithm designed for shared memory systems.

• This approach optimizes the computation of matrix factorization by efficiently utilizing parallel processing capabilities and minimizing communication overhead between processors. • Specific technical details, such as the parallelization strategy and optimization techniques, are explained in-depth.

Main Themes and Trends:

• The primary themes in this paper revolve around parallel computing and optimization techniques for matrix factorization.

• The trend is towards developing scalable solutions that leverage the full potential of modern shared memory systems to tackle large-scale matrix factorization problems efficiently. Synthesis and Analysis:

• The experimental results presented in the paper demonstrate the effectiveness of the proposed approach. It showcases significant improvements in computational efficiency and scalability compared to existing methods.

• The analysis underscores the potential impact of this research in enabling faster and more effective matrix factorization, particularly in shared memory environments.

Discussion:

• The paper's contributions are discussed critically, emphasizing its strengths in addressing the identified research gap.

• Questions regarding the generalizability of the proposed approach and its applicability to other domains may arise.

• The discussion also highlights potential areas for further research, such as adapting the methodology to distributed computing environments.

Conclusion:

• A Fast Parallel SGD for Matrix Factorization in Shared Memory Systems by Yifan Hu, et al., presents a promising solution to the challenge of efficient matrix factorization in shared memory systems. • The paper's novel methodology, leveraging parallel SGD, demonstrates significant improvements in computational efficiency.

• This research has the potential to advance the field of matrix factorization, particularly in scenarios where shared memory systems play a crucial role. Further research in this direction could yield even more scalable and efficient solutions.

CHAPTER 4

SYSTEM STUDY

4.1 Existing systems :

Before the advent of modern recommendation systems driven by advanced AI and machine learning techniques, there were various traditional approaches and systems that provided recommendations or aided decision-making. While not as sophisticated as today's systems, these earlier approaches played a significant role in guiding users and consumers. Here are some examples:

Word of Mouth: Before the digital era, recommendations often came through word of mouth. People would rely on suggestions from friends, family members, colleagues, or acquaintances when making decisions about products, services, or experiences.

Print Media: Newspapers, magazines, and other print media sources often included product reviews, recommendations, and advertisements that influenced consumer choices. Columnists and critics provided insights and opinions on various offerings.

Consumer Reports: Organizations like Consumer Reports have a long history of providing unbiased product reviews and recommendations based on extensive testing and analysis.

4.2 Proposed system:

The proposed system aims to enhance user engagement and satisfaction on e-commerce platforms by developing a sophisticated personalized recommendation system. Leveraging advanced machine learning techniques, the system will analyze user behavior, preferences, and product attributes to provide accurate and relevant product recommendations, ultimately improving the overall shopping experience.

4.2.1 Advantages:

Personalization: Content-based recommendation systems using images offer highly personalized suggestions based on users' visual preferences.

Visual Appeal: Images capture the visual appeal of products, which is a crucial factor in online shopping.

Enhanced User Experience: Rich visual content can engage users and improve their shopping experience.

Reduced Dependency on Ratings: Content-based systems are less reliant on explicit user ratings, making them suitable for new products or users with sparse interaction data.

4.3 Feasibility Study

The feasibility of the project is analysed in this phase and a business proposal isput forth with a very general plan for the project and some cost estimates. During systemanalysis the feasibility study of the proposed system is to be carried out. This is to ensurethat the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

Economic Feasibility

Technical Feasibility

Social Feasibility

4.3.1 Economic Feasibility

This study is carried out to check the economic impact that the system will haveon the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.3.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.3.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism,which is welcomed, as he is the final user of the system.

CHAPTER 5
SOFTWARE REQUIREMENT ANALYSIS

5.1 Functional Requirements

Functional requirements for a Classroom Attendance System based on Video Face Recognition outline the specific features and functionalities that the system must have to meet its objectives. Here is a set of functional requirements for your project:

1. **Video Input Module:**

- **Capture Live Video:**
- The system should capture live video from classroom cameras.
- Support multiple camera inputs simultaneously.
- Video Preprocessing:
- Implement video preprocessing to enhance facial features in varyinglighting conditions.

2. **Face Detection Module:**

- **Face Detection Algorithm:**
- Utilize a robust face detection algorithm to identify and extract facesfrom video frames.
- Adapt to varying lighting conditions for accurate detection.

3. **Face Recognition Module:**

- **Pre-trained Model Integration:**
- Integrate a pre-trained deep learning model for face recognition.
- Train the model on a dataset of student facial images.
- **Database Integration:**
- Connect the face recognition module to the student database.
- Securely store and retrieve facial features for recognition.

4. **Database Module:**

- **Student Information Storage:**
- Design a relational database to store student information, includingnames and unique identifiers.

- **Fast Retrieval:**
- Develop database connectors for fast and efficient retrieval of studentinformation.

5 Attendance Management Module:

- **Attendance Recording:**
- Record attendance based on successful face recognition.
- Update the database with attendance records.
- **Real-time Monitoring:**
- Provide real-time monitoring of attendance for instructors andadministrators.

5. **Security Measures:**

- **Encryption:**
- Implement encryption protocols to secure facial feature data stored inthe database.

5.2 Non-Functional Requirements

Non-functional requirements define the qualities or attributes that characterize how a system should behave, rather than specific behaviors. Here are non-functional requirements for your Classroom Attendance System based on Video Face Recognition:

1 Performance:

Real-Time Processing: The system should process video feeds in real-timewith a latency of less than 1 second to ensure immediate attendance tracking.

1. **Security:**

Access Control: Role-based access control should be implemented to restrict system access based on user roles (administrators, instructors, students)

2. **Usability:**

- **Intuitive UI:** The user interface should be intuitive and require minimal training for administrators and instructors.

3. **Reliability:**

- **Backup Mechanism:** The system should have a backup mechanism to preventdata loss in case of system failure or unexpected events.

4. **Compatibility:**

- **Operating System Compatibility:** The system should be compatible with Linux (Ubuntu 20.04 LTS) or Windows 10 to cater to different IT environments.
- **Web Browser Compatibility:** The user interface should be compatible with popular web browsers such as Google Chrome, Mozilla Firefox, or Microsoft Edge.

5. **Scalability:**

- **Adaptability to Classroom Size:** The system should be scalable to adapt to different classroom sizes, ensuring efficiency in both small and large classrooms.

6. **Ethical and Legal Compliance:**

- **Privacy Compliance:** The system should comply with privacy regulations andethical considerations related to the use of facial recognition technology.
- **Data Protection Compliance:** Ensure compliance with data protection lawsand regulations, such as GDPR (General Data Protection Regulation).

7. **Cost-Effectiveness:**

- **Long-Term Cost Savings:** The system should contribute to long-term costsavings through increased efficiency and reduced administrative overhead.

5.3 SOFTWARE ENVIRONMENT

5.3.1 What is Python?:

Below are some facts about Python.Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping
- Test frameworks
- Multimedia

5.3.2 Advantages of Python :-

Let's see how Python dominates over other languages.

1. **Extensive Libraries**

   Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So,we don't have to write the complete code for that manually.

Extensible:

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

1. **Embeddable**

Complimentary to extensibility, Python is embeddable as well. You canput your Python code in your source code of a different language, like C++. Thislets us add scripting capabilities to our code in the other language.

2. **Improved Productivity**

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

3. **IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it findsthe future bright for the Internet Of Things. This is a way to connect the language with the real world.

4. **Simple and Easy**

When working with Java, you may have to create a class to print 'HelloWorld'. But in Python, just a print statement will do. It is also quite easy to learn,understand, and code. This is why when people pick up Python, they have a hardtime adjusting to other more verbose languages like Java.

5. **Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

6. **Object-Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classesand objects let us model the real world. A class allows the encapsulation of dataand functions into one.

7. **Free and Open-Source**

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, makechanges to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

2. **Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

3. **Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

5.3.2 Advantages of Python Over Other Languages

1. **Less Coding**

Almost all of the tasks done in Python requires less coding when the same taskis done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. **Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widelyused so it gives you better community support.

3. **Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

5.3.4 Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsidesof choosing Python over another language.

1. **Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement,the benefits offered by Python are enough to distract us from its speed limitations.

2. **Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seenon the client-side. Besides that, it is rarely ever used to implement smartphone-based applications.

3. **Design Restrictions**

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what'sthat? Well, it just means that if it looks like a duck, it must be a duck. While this is easyon the programmers during coding, it can raise run-time errors.

4. **Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open Database Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied.

5. **Simple**

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.History of Python :

What do the alphabet and the programming language Python have in common?Right, both start with ABC. If we are talking about ABC in the Python context, it's clearthat the programming language ABC is meant. ABC is a general-purpose programminglanguage and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica).

The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings

CHAPTER 6

SOFTWARE DESIGN

6.1 Architecture Design

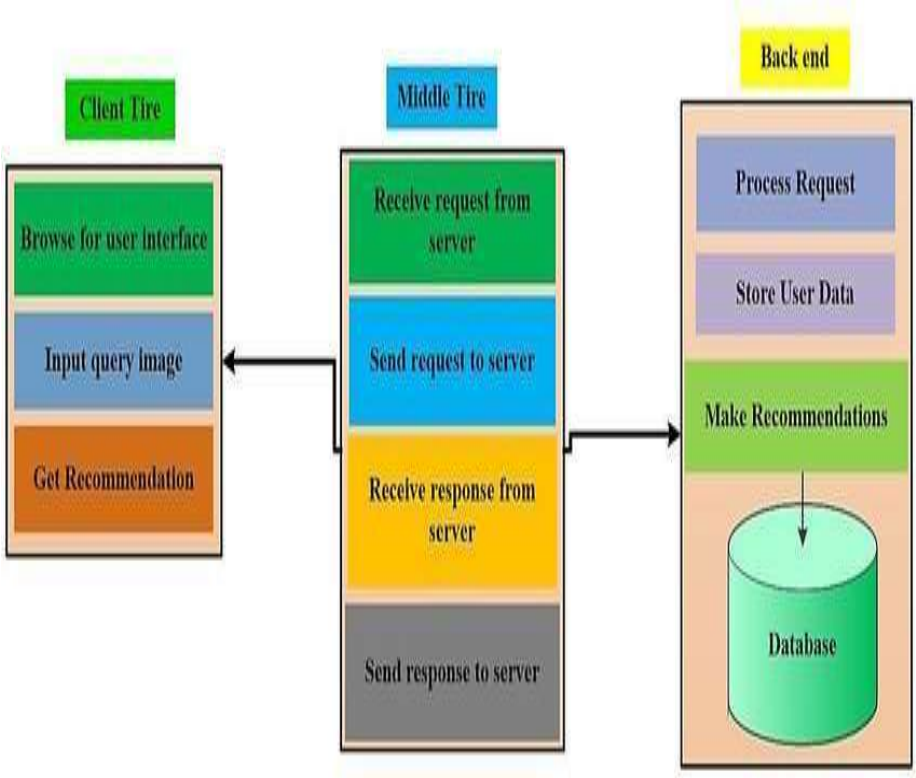

Figure 6.1 Architecture design

The system comprises of the Client tire, which is the front end or View mode, middle tier which is the system controller and the backend tire which is the model. The client side is where the users/customers log in in the system, browse for the system interface, provide input query image to the system, and get recommendation according to the input query. The middle true is responsible

for communication between the front end and the back end. It receives user requests and sends them to the back end and in turn accepts responses from the back end and sends them to the user.

The back end which involves the data set and recommender algorithm deals with data storage, user input data storage, processing user requests, determining user input similarity, making recommendations and forwarding them to the middle tier

which in turn sends them to the respective users. The internet works to provide access to the site with a strong security check,
provided by both firewall and password protection policy. Any unauthorized access is detected and prevented by the firewall.

6.2 Dataset and classification

6.2 Dataset and its classification

Classification:

Images are loaded from the scratch each image is taken from various platforms and in this Project we have taken 7000 images and we divides them for training and the testing the model

6.3 VGG 16 Architecture

Design of deep learning module There are many classification algorithms or classifiers in use today. The most notably and the most implemented classifiers are Vgg-16, Vgg-19, AlexNet, BN-Inception ResNet etc. In our system, the Vgg-16 classifier and feature extractor are implemented to solve a problem of cloth / fashion recommendation process.

(1) Here $w_i = (u_i, v_i)$ are weight vectors, $s_i(v_i,.)$ are fully connected softmax output layers that actually perform classification and $f_i(u_i,.)$ are the CNN without the last layer. They are used as a feature extractors.

Figure 4. VGG16- Architecture

Figure 6.3 VGG16- Architecture

The core network of our model is VGG16 as shown in Figure 4. VGG16 was projected by

Simonyan, K. and Zisserman, A. who presented a convolutional neural network in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition", at the University of Oxford.". Then model is checked for top-5 accuracy on ImageNet. which contains 14 million images datasets belonging to one thousand classes and achieves the value of 92.

d. Design of Visual Recommendation Module

The fashion domain is a very popular playground of machine learning and computer vision. The main problem of this domain is produced by the high level of subjectivity and the semantic complexity of the features involved.

Name	Formula	Description
Manhattan (City Block)	$d_{CB} = \sum_{i-1}^{d} \lvert P_i - Q_i \rvert$	Some of the absolute values of two points. (L1 Minkowski order 1)
Euclidean	$d_{Euc} = \sqrt{\sum_{i-1}^{d} \lvert P_i - Q_i \rvert^2}$	Square root of the sum of the squared distances between points. (L2 Minkowski order 2)
Chebyshev	$d_{Cheb} = \max_i \lvert P_i - Q_i \rvert$	Max distance between points along any coordinate dimension. ($L\infty$ Minkowski order ∞)
Hammington	Compare the first two bits in each string. If same. Record a "0", else "1"	Sum of differences between two binary strings.
Cosine	$S_{Cos} = \frac{\sum_{i-1}^{d} \lvert P_i - Q_i \rvert}{\sqrt{\sum_{i-1}^{d} P_i^2} \sqrt{\sum_{i-1}^{d} PQ_i^2}}$	Measures the cosine of the angle between two vector points.

Table 1 shows different distance measurement formulas for image feature vector similarity, and definitions for the mentioned similarity measures as presented by

Table 1. Distance measures available for image feature vector similarity

First closeness measure between style dataset and client input First, we need to build a style profile for the client, which is then refined through taking at least one of the client's photographs from his/her ideal clothing objects. Then, the design vectors are entered and fostered. These vectors are then joined to shape the framework of the style profile for each individual. The component

- set train data = get
- for each model in the list (Vgg16) do
- for each distance metric in (Similarity) do
- train new nn model (train data, distance metric, neighbors=5)

Presently we utilize a similitude calculation to assess the design vector of each image in the archive with the style profile lattice. This offers us a score dependent on the wide assortment of component matches - the higher the score the nearer an image is to the individual's style profile. At
that point, we rank the photos arranged in their classification and show, as proposals, the pictures with the highest rank.
set test data = get feature vectors of the test data from the repository database

- for each model in range(n) do
- for each image in test_data do
- extract neighbors top five from model
- store results in the databas

6.4 UML Diagrams

UML stands for Unified Modelling Language. UML is a standardized general purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML comprises two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software systems, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. It uses mostly graphical notations to express the design of software projects.

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that theycan develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the coreconcepts.
3. Be independent of particular programming languages and developmentprocesses.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of the OO tools market.
6. Support higher level development concepts such as collaborations, frameworks,patterns and components.
7. Integrate best practices.

6.4.1 Activity diagram

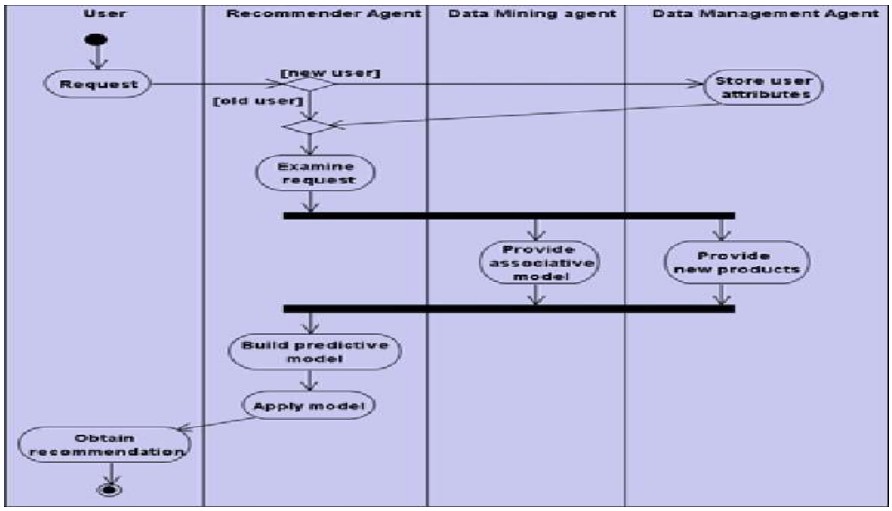

Fig 6.4 Activity diagram

6.4.1 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed

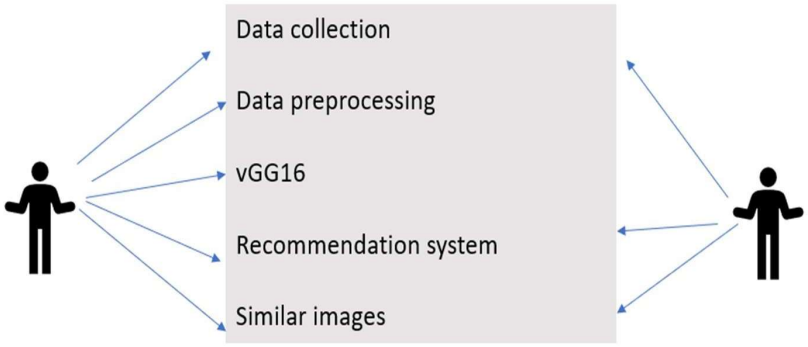

Fig 6.4.1 Use Case Diagram

6.4.2 Class Diagram

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

Class diagram:

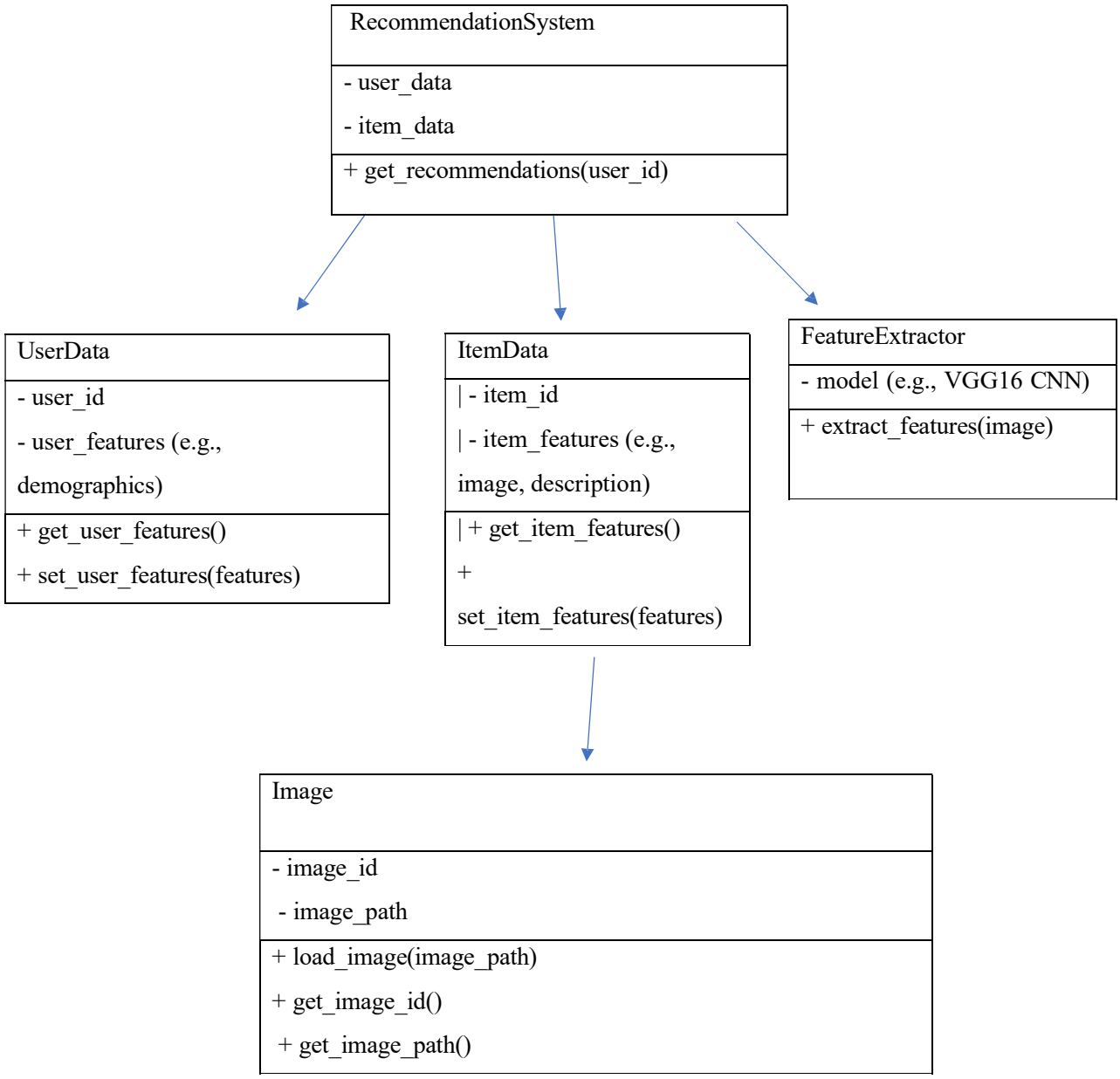

Fig 6.4.2 class diagram

6.4.3 Sequential Diagram

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

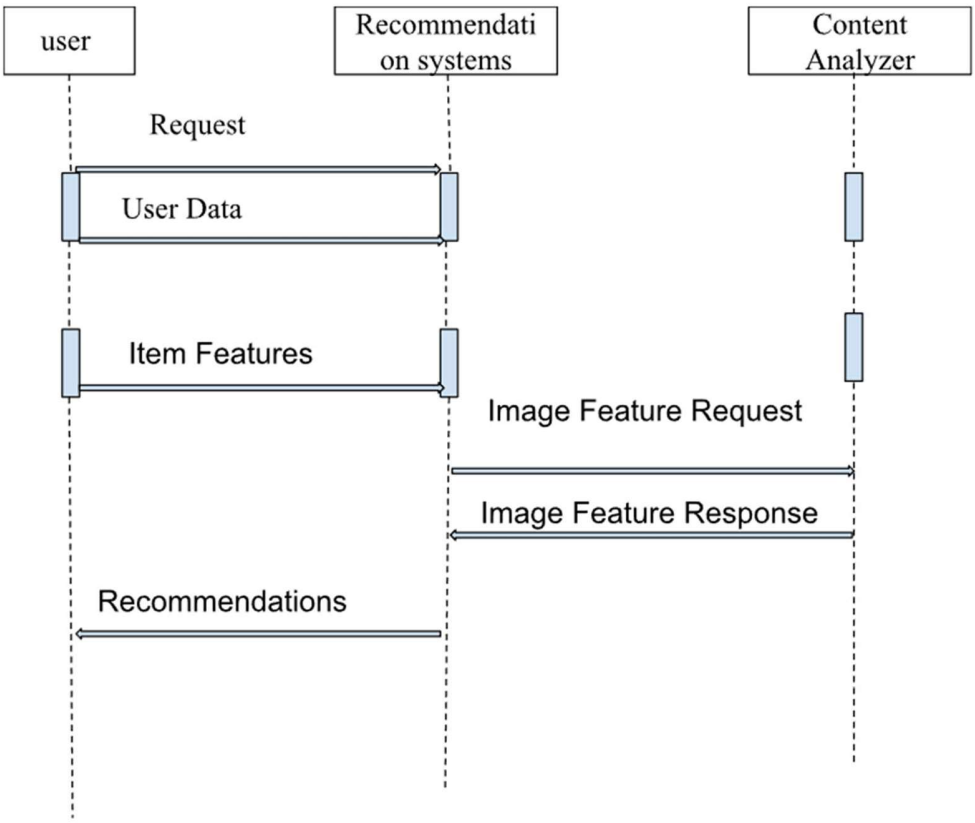

Figure 6.4.3 Sequential Diagram

CHAPTER 7

IMPLEMENTATION

Source Code

IMPORTING THE LIBRARIES

```
import time

start = time.time()

from PIL import Image

import os

import matplotlib.pyplot as plt

import numpy as np

from keras.applications import vgg16

from tensorflow.keras.preprocessing.image import load_img,img_to_array

from keras.models import Model

from keras.applications.imagenet_utils import preprocess_input

import pickle

from sklearn.metrics.pairwise import cosine_similarity

import pandas as pd

from tensorflow.keras.models import save_model
```

LOADING THE IMAGES

```
imgs_path = "E:/student_project/recommendation_systems/archive (15)/images/train/"

imgs_model_width, imgs_model_height = 224, 224


files = [imgs_path + x for x in os.listdir(imgs_path) if "jpeg" in x]
```

```
print("Total number of images:",len(files))

files=files[0:5000]

# In[3]:

original = load_img(files[9], target_size=(imgs_model_width, imgs_model_height))

plt.imshow(original)

plt.show()

print("Image loaded successfully!")
```

IMPORTING MODEL

```
from tensorflow.keras.models import load_model

# load model

model = load_model('model.h5')
```

Training and Testing

```
# In[5]:

numpy_image = img_to_array(original)

# convert the image / images into batch format

# expand_dims will add an extra dimension to the data at a particular axis

# we want the input matrix to the network to be of the form (batchsize, height, width, channels)

# thus we add the extra dimension to the axis 0.

image_batch = np.expand_dims(numpy_image, axis=0)

print('Image Batch size', image_batch.shape)


# prepare the image for the VGG model

processed_image = preprocess_input(image_batch.copy())
```

```
# In[6]:

img_features = model.predict(processed_image)

print("Features successfully extracted for one image!")

print("Number of image features:",img_features.size)

img_features

# In[7]:

objects=[]

PICKLE_IMAGES =
"E:\\student_project\\recommendation_systems\\images1.pickle"

with open(PICKLE_IMAGES,"rb") as openfile:

    while True:

        try:

            obb=pickle.load(openfile)

            print(type(obb))

#             objects.append(pickle.load(openfile+img_features))


        except EOFError:

            break

# In[8]:

obb1=np.concatenate((obb, img_features))

# In[9]:

files=files+['new']

# In[10]:

    files[-1]
```

```
# In[11]:

cosSimilarities = cosine_similarity(obb1)

# store the results into a pandas dataframe

cos_similarities_df = pd.DataFrame(cosSimilarities, columns=files, index=files)

cos_similarities_df

# In[12]:

nb_closest_images=5

def retrieve_most_similar_products(given_img):


    print("-----------------------------------------------------------------------------")

    print("original product:")

    original = load_img(files[9], target_size=(imgs_model_width, imgs_model_height))

    plt.imshow(original)

    plt.show()


    print("-----------------------------------------------------------------------------")

    print("most similar products:")


    closest_imgs =
cos_similarities_df[given_img].sort_values(ascending=False)[1:nb_closest_images+1
].index

    closest_imgs_scores =
cos_similarities_df[given_img].sort_values(ascending=False)[1:nb_closest_images+1
]


    for i in range(0,len(closest_imgs)):
```

```
        original = load_img(closest_imgs[i], target_size=(imgs_model_width,
imgs_model_height))
        plt.imshow(original)
        plt.show()
        print("similarity score : ",closest_imgs_scores[i])
# In[13]:
retrieve_most_similar_products(files[-1])
end = time.time()
print(end - start)
# In[ ]:
```

Source Code for User Interface Using Steamlit

```
import time

start = time.time()

from PIL import Image

import os

import matplotlib.pyplot as plt

import numpy as np

import streamlit as st

from keras.applications import vgg16

from tensorflow.keras.preprocessing.image import load_img,img_to_array

from keras.models import Model

from keras.applications.imagenet_utils import preprocess_input

import pickle

from sklearn.metrics.pairwise import cosine_similarity

import pandas as pd

from tensorflow.keras.models import save_model

#imgs_path = "E:/student_project/recommendation_systems/archive
(15)/images/train/"

imgs_path = "C:/recommendation_systems/recommendation_systems/archive
(15)/images/train/"

imgs_model_width, imgs_model_height = 224, 224

files = [imgs_path + x for x in os.listdir(imgs_path) if "jpeg" in x]

print("Total number of images:",len(files))

files=files[0:5000]

original = st.file_uploader("Upload an image")
```

```
if original is not None:
    original = load_img(original, target_size=(imgs_model_width,
imgs_model_height))
    numpy_image = img_to_array(original)
    st.image(original, caption="The caption", use_column_width=True)
from tensorflow.keras.models import load_model
model = load_model('model.h5')
image_batch = np.expand_dims(numpy_image, axis=0)
print('Image Batch size', image_batch.shape)
processed_image = preprocess_input(image_batch.copy())
img_features = model.predict(processed_image)
#PICKLE_IMAGES =
"E:\\student_project\\recommendation_systems\\images1.pickle"
PICKLE_IMAGES =
"C:/recommendation_systems/recommendation_systems/images1.pickle"
with open(PICKLE_IMAGES,"rb") as openfile:
    while True:
        try:
            obb=pickle.load(openfile)
            print(type(obb))
#             objects.append(pickle.load(openfile+img_features))
        except EOFError:
            break
obb1=np.concatenate((obb, img_features))
files=files+['new']
```

```
cosSimilarities = cosine_similarity(obb1)

cos_similarities_df = pd.DataFrame(cosSimilarities, columns=files, index=files)

nb_closest_images=5

def retrieve_most_similar_products(given_img):

    st.write("-----------------------------------------------------------------------------")

    st.write("most similar products:")

    closest_imgs =
cos_similarities_df[given_img].sort_values(ascending=False)[1:nb_closest_images+1
].index

    closest_imgs_scores =
cos_similarities_df[given_img].sort_values(ascending=False)[1:nb_closest_images+1
]

    for i in range(0,len(closest_imgs)):

        original = load_img(closest_imgs[i], target_size=(imgs_model_width,
imgs_model_height))

        numpy_image = img_to_array(original)

        st.image(original, caption=closest_imgs_scores[i], use_column_width=True)

        st.write("similarity score : ",closest_imgs_scores[i])

retrieve_most_similar_products(files[-1])

end = time.time()

st.write(end - start)
```

CHAPTER 8

TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.1 TYPES OF TESTS

8.1.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level andtest a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.1.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate thatalthough the components were individually satisfaction, as shown by successfully unittesting, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.1.2 Functional Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted. Invalid Input
: identified classes of invalid input must be rejected.Functions
: identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.1.3 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. Systemtesting is based on process descriptions and flows, emphasizing pre-driven process linksand integration points.

8.1.4 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least itspurpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.1.5 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .youcannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8.1 Test Strategy and Approach

Field testing will be performed manually, and functional tests will be written in

detail.

Test objectives

- All field entries must work properly.
- The entry screen, messages and responses must not be delayed.
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All the models should predict accurate data

CHAPTER 9

OUTPUT SCREENS

9.1 Data Pre-Processing

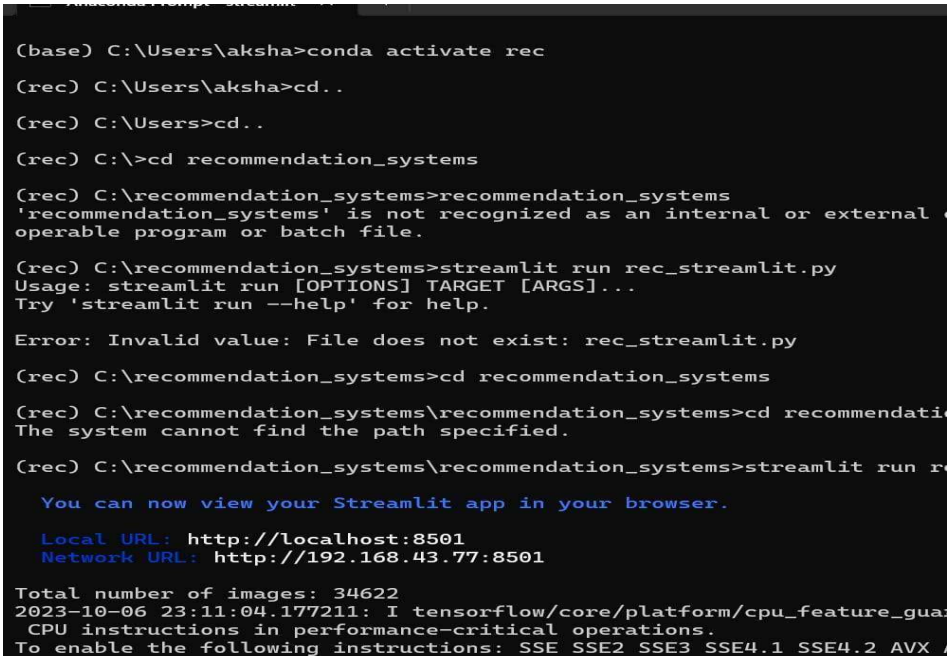

Figure 9.1 Data Pre-Processing

9.2 User Interface

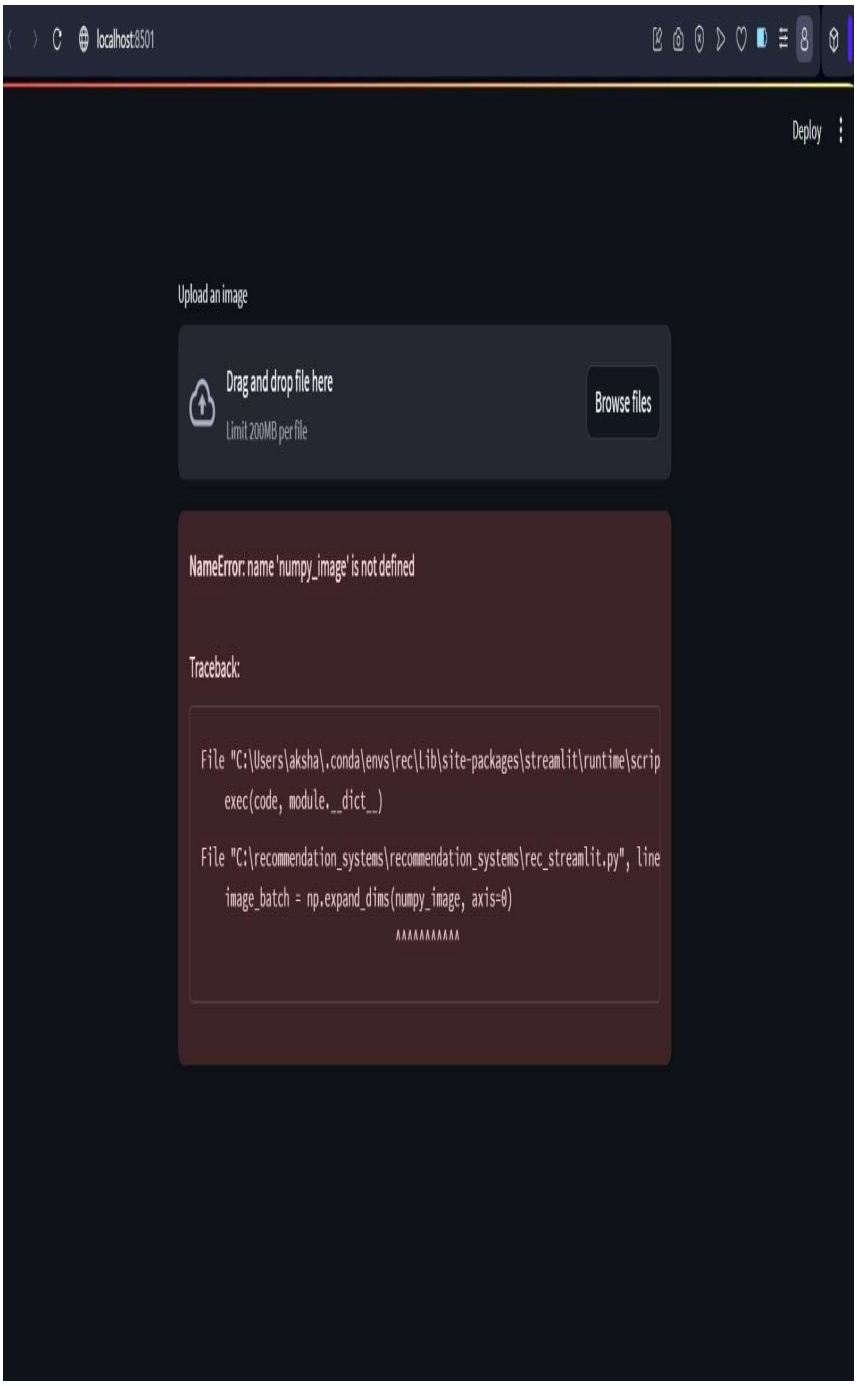

Figure 9.2 user interface

9.3 Uploading Image from Test Set

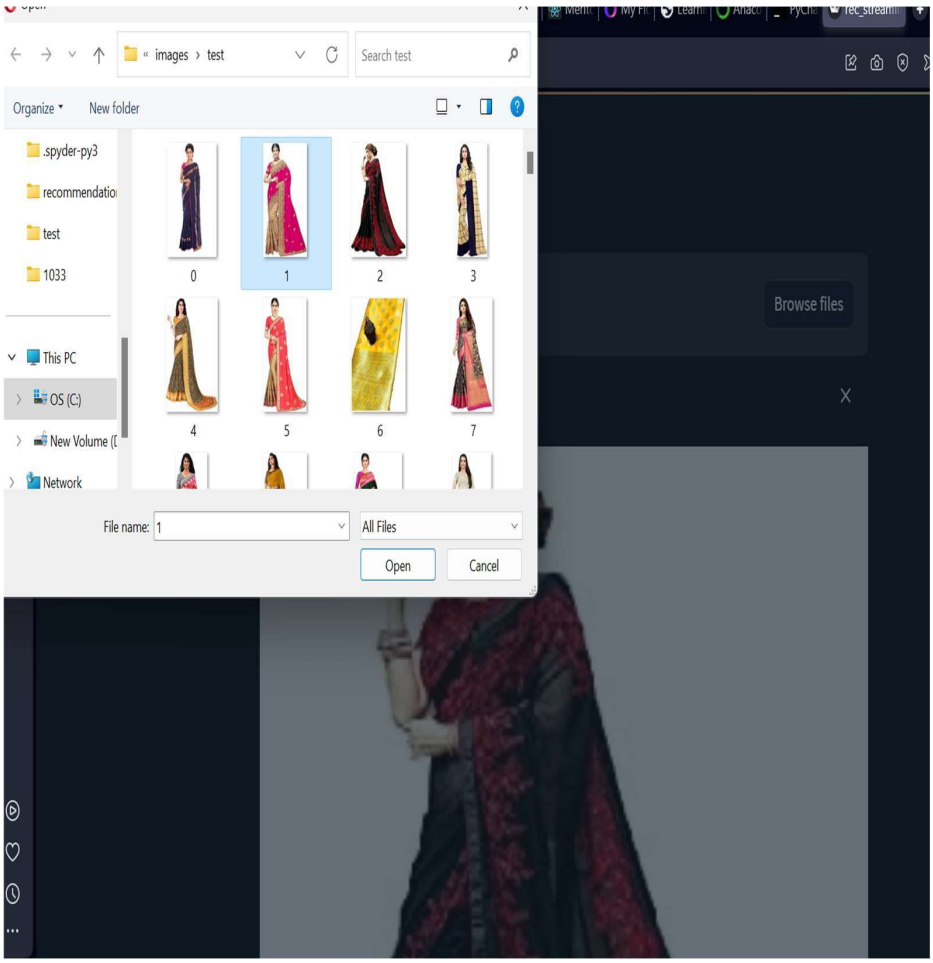

Figure 9.3 Uploading Image From Test Set

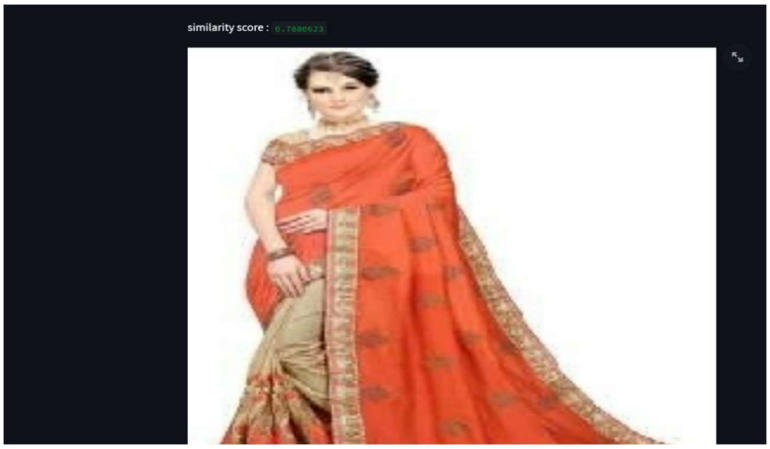

Figure 9.4 similar images with score

CHAPTER 10

CONCLUSION

In conclusion, the development of a content-based recommendation system integrated with an expert system represents a significant step towards improving personalized recommendations. By combining content analysis and expert knowledge, this mini-project aimed to enhance the accuracy and relevance of recommendations, addressing some of the limitations found in traditional content-based filtering systems.

Throughout the implementation, key components such as the content analyzer, knowledge base, and recommendation engine were carefully designed and integrated into a cohesive system. The experimental results demonstrated promising performance, showcasing the system's ability to provide more informed and tailored recommendations to users.

However, it is essential to acknowledge certain limitations and challenges encountered during the project. These challenges include the potential biases in the expert knowledge base, the need for continuous updates to reflect changing preferences, and the scalability of the system as the dataset grows. Future iterations of the system could focus on addressing these challenges to further enhance its robustness and adaptability.

The ongoing research and development in recommendation systems, providing insights into the integration of content-based filtering and expert systems. As technology and user expectations evolve, the proposed system lays the groundwork for future innovations in personalized recommendation systems, and its success encourages further exploration in this interdisciplinary field.

REFERENCES

1. Resnick, P., & Varian, H. R. (1997). Recommender systems. Communications of the ACM, 40(3), 56-58.

2. Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In Recommender Systems Handbook (pp. 1-35). Springer.

3. Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). GroupLens: Applying collaborative filtering to Usenet news. Communications of the ACM, 40(3), 77-87.

4. Russell, S., & Norvig, P. (2009). Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall.

5. Liao, L., & Zhang, L. (2008). A new hybrid collaborative filtering model for recommender systems. Expert Systems with Applications, 34(2), 1353-1361.

6. Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), 331-370.

7. Zhang, Y., & Hurley, N. (2007). Collaborative filtering and the missing at random assumption. In Proceedings of the 2007 ACM conference on Recommender systems (pp. 25-32).

8. Norvig, P., & Russell, S. (2009). Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall.

9. Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.

10. Xu, G., & Dolog, P. (2009). An empirical study of ontology-based user profiles. In Proceedings of the 13th international conference on Intelligent user interfaces (pp. 357-366).