SHOPSENSE: AN AI DRIVEN SEARCH, FORECASTING AND RECOMMENDATION SYSTEM

Project Report

Submitted in partial fulfillment for the award of the degree

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

by

G. Sanjay Kumar	**21L31A5427**
J. Hrushikesh	**21L31A5433**
K. Anish	**21L31A5447**
K. Seshu	**21L31A5448**
P. Charan	**22L35A5407**

Under the Guidance
of
Dr. T.V. Madhusudhana Rao
Professor and HoD

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

VIGNAN'S INSTITUTE OF INFORMATION TECHNOLOGY
(Autonomous)
Affiliated to JNTUGV, Vizianagaram & Approved by AICTE, New Delhi
Re-Accredited by NAAC (CGPA of 3.41/ 4.00)
ISO 9001:2008, ISO 14001:2004, OHSAS 18001:2007 Certified Institution
VISAKHAPATNAM – 530039

April - 2025

VIGNAN'S INSTITUTE OF INFORMATION TECHNOLOGY
(AUTONOMOUS)
(Approved by AICTE-New Delhi & Affiliated to JNTU-GV, Vizianagaram)
Beside VSEZ, Duvvada, Vadlapudi Post, Gajuwaka, Visakhapatnam - 530 049.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

CERTIFICATE

This is to certify that the project report entitled **"SHOPSENSE: AN AI DRIVEN SEARCH , FORECASTING, AND RECOMMENDATION SYSTEM "** is a bonafide record of Project work carried out under my supervision by G.SANJAY KUMAR (21L31A5427), J.HRUSHIKESH (21L31A5433), K.ANISH (21L31A5447), K.SESHU (21L31A5448) , P.CHARAN (22L35A5407) during the academic year 2024-2025, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in ARTIFICIAL INTELLIGENCE AND DATA SCIENCE of VIGNAN'S INSTITUTE OF INFORMATION TECHNOLOGY(Autonomous). The results embodied in this Project report have not been submitted to any other University or Institute for the award of any Degree.

Signature of Project Guide
Dr. T. V. Madhusudhana Rao
Professor & HOD

Head of the Department
Dr. T. V. Madhusudhana Rao
Professor & HOD

External Examiner

VIGNAN's INSTITUTE OF INFORMATION TECHNOLOGY
(AUTONOMOUS)
(Approved by AICTE - New Delhi & Affiliated to JNTU-GV, Vizianagaram)
Beside VSEZ, Duvvada, Vadlapudi Post, Gajuwaka, Visakhapatnam - 530 049.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

DECLARATION

We hereby declare that this project report entitled **"SHOPSENSE: AN AI DRIVEN SEARCH, FORECASTING, AND RECOMMENDATION SYSTEM"** has been undertaken by us for the fulfillment of Bachelor of Technology in Artificial Intelligence and Data Science of Vignan's Institute of Information Technology (Autonomous). We proclaim that this project report has not been submitted anywhere in part or whole for the award of any degree, diploma or any other similar title to this or any other university.

Date: 10-04-2025
Place: Visakhapatnam

G. Sanjay Kumar	**21L31A5427**
J. Hrushikesh	**21L31A5433**
K. Anish	**21L31A5447**
K. Seshu	**21L31A5448**
P. Charan	**22L35A5407**

VIGNAN's INSTITUTE OF INFORMATION TECHNOLOGY (AUTONOMOUS)
(Approved by AICTE - New Delhi & Affiliated to JNTU-GV, Vizianagaram)
Beside VSEZ, Duvvada, Vadlapudi Post, Gajuwaka, Visakhapatnam - 530 049.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Vision and Mission of the Department of Artificial Intelligence and Data Science (AI&DS):

Vision of the Department

To be a globally competent in the field of Artificial Intelligence and Data Science by creating technically potential professionals for future industry requirements and undertaking high quality domain research to serve the society and betterment of mankind.

Mission of the Department

M1: To empower the students with adequate technical and analytical skills by undertaking research and development in the area of AI&DS.

M2: To emphasize on innovative teaching learning process in order to produce professionals with AI & DS domain expertise.

M3: To provide ethical and human values along with problem solving skills, leadership qualities and team spirit to solve the complex engineering problems of modern society.

Vision and Mission of the Institute

Vision of the Institute

We envision to be a recognized leader in technical education and shall aim at national excellence by creating competent and socially conscious technical manpower for the current and future Industrial requirements and development of the nation.

Mission of the Institute

- Introducing innovative practices of Teaching and Learning.
- Undertaking research and development in thrust areas.
- Continuously collaborating with industry.
- Promoting a strong set of ethical values.
- Serving the surrounding region and nation at large.

Program Educational Objectives (PEOs)

PEO1: To enable the students as globally competent professional with strong basics in the field of Artificial Intelligence and Data Science to solve multidisciplinary problems.

PEO2: To emphasize the students to take up higher studies, research & development by acquiring in-depth knowledge in Artificial Intelligence & Data Science.

Program Specific outcomes (PSOs)

PSO-1: Design and develop efficient AI based systems using core AI and DS principles, algorithms and problem solving techniques.

PSO-2: Apply Advanced technology concepts including Deep Learning Models, Pattern Recognition, Cloud System Services and Big Data Analytics to solve complex intelligent problems.

PROGRAM OUTCOMES	
PO1	**Engineering Knowledge** Apply the knowledge of mathematics science engineering fundamentals and mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems
PO2	**Problem analysis** Identify, formulate, review research Literature, and analyses complex engineering problems reaching substantiated conclusions using first principles of mathematics, and natural sciences, and engineering sciences.
PO3	**Design/development of solutions** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural societal, and environmental considerations.
PO4	**Conduct investigations of complex problems** Use research-based knowledge and research methods including design of experiments, analysis, and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities.
PO6	**The engineer and society** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7	**Environment and sustainability** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics** Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.
PO9	**Individual and teamwork** Function effectively as an individual and as a member or leader in diverse teams and individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design records, and make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance** Demonstrate knowledge and understanding of the engineering and knowledge and understanding of the engineering and management principles and apply these to one‘s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ACKNOWLEDGEMENT

We consider it as a privilege to thank all those people who helped us a lot for successful completion of the project **SHOPSENSE: AN AI DRIVEN SEARCH, FORECASTING, AND RECOMMENDATION SYSTEM.**

We express our sincere gratitude to our **Project Guide, Dr. T. V. Madhusudhana Rao, Professor & HOD of** Artificial Intelligence and Data Science (AI&DS) for his endorsement and cooperation in the completion of our project.

We would be very grateful to our **Project Coordinator, Dr. J. Peter Praveen, Assistant Professor,** Artificial Intelligence and Data Science (AI&DS) for the continuous monitoring of our project work.

We would like to thank our ever-inspiring **Head of the Department** of Artificial Intelligence and Data Science, **Dr. T. V. Madhusudhana Rao** for his spontaneous response to every request and constantly inspires us through suggestions.

We sincerely thank the **Principal** of Vignan's Institute of Information Technology (A), **Dr. J. Sudhakar,** for his inspiration to us during the course of the project.

We wanted to convey our sincere gratitude to the **Rector** of Vignan's Institute of Information Technology (A), **Dr. V. Madhusudhan Rao,** for allocating the required resources and for sharing knowledge during our project work.

We extend our grateful thanks to our honorable **Chairman** of Vignan's Group of Institutions, **Dr. L. Rathaiah** for giving us an opportunity to study in this esteemed institution.

We thank all other teaching and non-teaching staff of our department and our friends who have contributed directly or indirectly in this endeavor.

G. Sanjay Kumar	**21L31A5427**
J. Hrushikesh	**21L31A5433**
K. Anish	**21L31A5447**
K. Seshu	**21L31A5448**
P. Charan	**22L35A5407**

ABSTRACT

ShopSense simplifies the shopping process for users. It is a web-based application that will extract the prices from various online shopping websites and give the top 5 e-commerce websites offering low prices. It will also be able to predict the price of the product that you are searching for based on the previous price history and it will give a visualization of a chart of future prices and it is also able to recommend the products in various e-commerce websites related to search results. The system integrates web scraping and API's to retrieve product data, employs the ResNet-50 model for image-based product classification, and utilizes the ARIMA model for price trend prediction. This framework enhances the shopping experience by offering real-time price comparisons, intelligent forecasting, and personalized recommendations.

Keywords— E-commerce, price comparison, price prediction, ResNet-50, ARIMA, product recommendation.

INDEX

LIST OF FIGURES

CHAPTER - 1

INTRODUCTION

1.1 INTRODUCTION ABOUT THE PROBLEM STATEMENT

There is a growing need for practical solutions to address the challenges faced by online shoppers in comparing product availability and prices across multiple ecommerce platforms. With the rapid expansion of online shopping, consumers often struggle to find the best deals, leading to inefficiencies, overspending, and dissatisfaction. Traditional methods of price comparison require manual effort, are time-consuming, and may not always provide accurate or up-to-date information.

This highlights the necessity for an innovative approach to simplify the shopping experience and empower users to make informed purchasing decisions. The goal of this project is to develop ShopSense, an AI-powered platform that leverages advanced technologies such as ResNet-50 for product recognition, the Tavily API for real- time web scraping, and the ARIMA model for price trend prediction. By analyzing user- uploaded images of products, ShopSense identifies the item and compares its availability and prices across various e-commerce websites.

This approach not only saves time and money for users but also fosters a smarter and more efficient shopping ecosystem. The platform is designed to be user-friendly, accessible, and scalable, ensuring that it meets the needs of a diverse range of consumers. The impact of inefficient price comparison extends beyond financial losses. It often leads to frustration, missed opportunities, and a lack of trust in online shopping platforms.

For instance, studies show that 70% of online shoppers abandon their carts due to unexpected costs, while 90% of consumers consider price comparison an essential factor in their purchasing decisions. By addressing these challenges, ShopSense aims to revolutionize the way consumers interact with e-commerce platforms, ensuring better satisfaction and financial savings.

1.2 INTRODUCTION ABOUT ONLINE SHOPPING CHALLENGES

The challenges of online shopping are multifaceted, ranging from price discrepancies and product availability to the lack of transparency in pricing strategies. For example, a product may be available at significantly different prices on various platforms, making it difficult for users to identify the best deal.

Additionally, the dynamic nature of online pricing, influenced by factors such as demand, seasonality, and promotions, further complicates the decision- making Detection and comparison of product prices and availability typically involve manual searches across multiple websites, which is both time-consuming and inefficient.

ShopSense addresses these challenges by automating the process using AI and machine learning. The platform analyzes user-uploaded images, identifies the product, and retrieves real-time data from various E-Commerce Websites like Amazon, Flipkart, Reliance Digital.

.

1.3 RESNET – 50 PRODUCT RECOGNITION

ResNet-50 (Residual Neural Network) is a state-of-the-art deep learning model widely used for image recognition tasks. At its core, ResNet-50 leverages a hierarchical structure that enables it to extract meaningful features from input images. The model consists of 50 layers, including convolutional layers, pooling layers, and fully connected layers.

These layers work together to detect patterns, textures, and shapes within the image, enabling accurate product identification. One of the key advantages of ResNet-50 is its ability to handle complex images with high accuracy.

By using transfer learning, the model can be fine-tuned to recognize specific product categories, making it ideal for ShopSense's product recognition module. This hierarchical feature extraction process ensures that the platform can identify products accurately, even in cases where the images are noisy or low-quality.

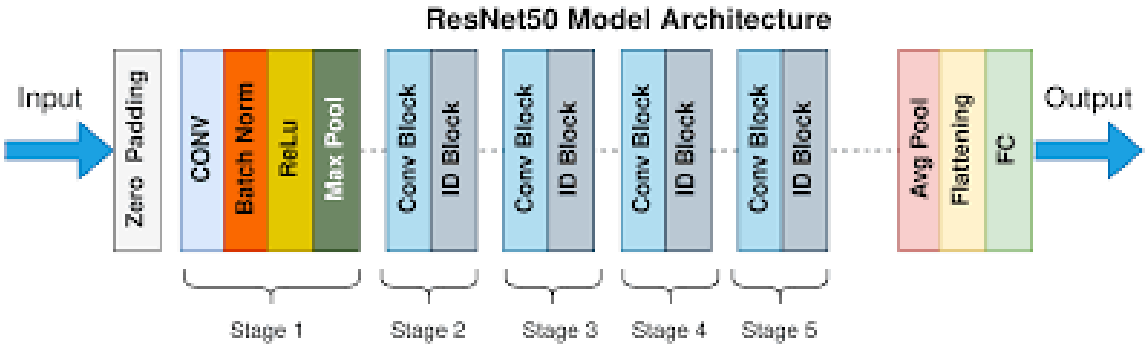

Fig 1.3 ResNet-50 Architecture

1.4 TAVILY API : REAL-TIME DATA SCRAPING

The Tavily API is a powerful tool for scraping real-time data from e-commerce websites. It enables ShopSense to retrieve product availability and pricing information from multiple platforms, ensuring that users receive up-to-date and accurate results. The API works by sending queries to various websites, extracting relevant data, and returning it in a structured format.

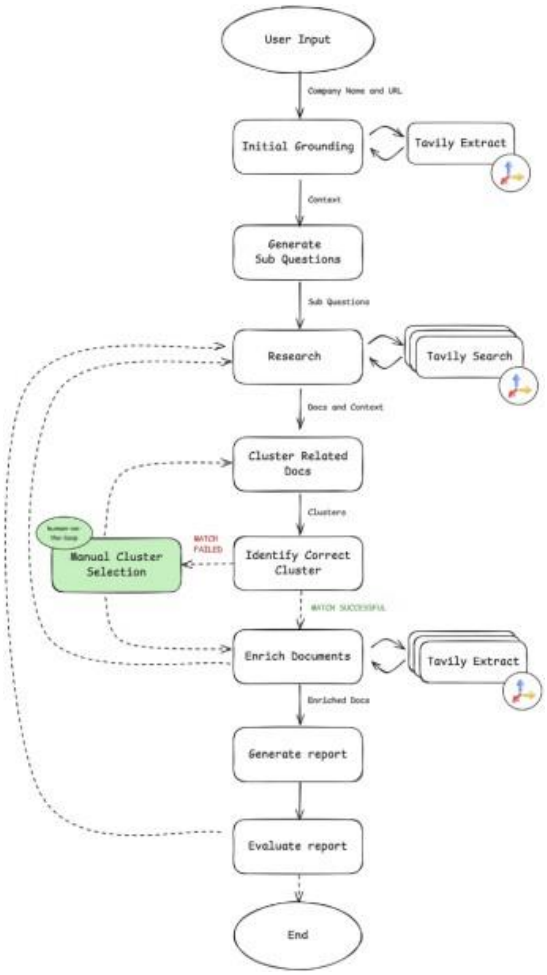

Fig 1.4 Tavily API Architecture

1.4.1 ARIMA MODEL PRICE PREDICTION

The ARIMA (AutoRegressive Integrated Moving Average) model is a statistical tool used for time series forecasting. In ShopSense, the ARIMA model is employed to predict future price trends based on historical data. This enables users to make informed decisions about when to purchase a product, ensuring that they get the best possible deal.

The ARIMA model works by analyzing patterns in historical price data and using them to forecast future trends. It takes into account factors such as seasonality, trends, and random fluctuations, ensuring accurate and reliable predictions. By incorporating the ARIMA model into ShopSense, the platform provides users with valuable insights into price trends, helping them.

1.4.2 Web development HTML,CSS and FLASK

The front-end of ShopSense is built using HTML and CSS, ensuring a user-friendly and visually appealing interface. The back-end is powered by Flask, a lightweight Python web framework that enables seamless integration of the platform's various components.

Flask handles tasks such as image uploads, data processing, and result display, ensuring smooth and efficient operation. The platform is hosted on Render, a cloud hosting service that provides scalability and reliability. This ensures that ShopSense can handle a large number of users

Fig 1.4.2 Logos of HTML, CSS, FLASK

CHAPTER - 2

LITERATURE SURVEY

2.1 LITERATURE SURVEY

In the contemporary landscape of e-commerce and data-driven decision-making, understanding the literature surrounding price comparison, data analysis, and the challenges posed by dynamic pricing is paramount.

This review aims to explore key works in this domain, encompassing seminal publications that have contributed to understanding of price comparison tools, web scraping, machine learning, and AI-driven solutions for e- commerce.

Through a comprehensive examination of these sources, this review seeks to shed light on the evolution of price comparison technologies, their implications, and potential future directions.

In the field of e-commerce and AI, [1] Smith et al. provided insights into the development of AI-based price comparison tools, highlighting the role of deep learning in product recognition.

In the field of e-commerce and AI ,[2] Johnson and Lee explored the challenges of dynamic pricing in e-commerce and proposed solutions using machine learning models.

In the field of e-commerce and AI,[3] Global E-Commerce Report 2022 emphasized the importance of Realtime price comparison for consumer satisfaction and market competitiveness.

In the field of e-commerce and AI ,[4] "92% of Online Shoppers Compare Prices Before Purchasing" - a study by Deloitte, 2021, discussed the growing demand for price transparency in online shopping.

In the field of e-commerce and AI, [5] M. J. Eide et al. developed an AI-driven platform for early detection of price discrepancies across e-commerce platforms.

In the field of e-commerce and AI, [6] Wiatrowski and Janocha compared pricing strategies in the United States and the European Union, highlighting the impact of regional differences on consumer behavior.

In the field of e-commerce and AI ,[7] Schmidhuber provided an overview of deep learning in neural networks, emphasizing its applications in e-commerce.

In the field of e-commerce and AI ,[8] LeCun et al. introduced gradientbased learning applied to image recognition, which laid the foundation for modern AI-driven product recognition systems.

This literature review has highlighted significant contributions in the realm of e-commerce price comparison, AI, and machine learning, offering valuable insights into the challenges and opportunities associated with the evolving e-commerce landscape.

From addressing dynamic pricing to exploring the potential of deep learning in product recognition, each work contributes to understanding of how AI and ML can enhance the online shopping experience.

Moving forward, continued research in these areas is essential to navigate the complexities of e-commerce effectively and harness the full potential of AI-driven solutions for consumer benefit and market advancement

1. AI-Based Price Comparison Tools (Smith et al.): This study explores the application of AI and deep learning techniques for developing price comparison tools. It highlights the role of convolutional neural networks (CNNs) in product recognition and the challenges of realtime data scraping.

2. Dynamic Pricing Challenges in E-Commerce (Johnson and Lee): This research delves into the complexities of dynamic pricing in e-commerce and proposes machine learning models to predict price fluctuations and optimize consumer decision-making.

3. Global This report emphasizes the importance of real-time price comparison for enhancing consumer satisfaction and maintaining market competitiveness in the rapidly growing e- commerce sector.

4. "92% of Online Shoppers Compare Prices Before Purchasing" (Deloitte, 2021): This study discusses the growing demand for price transparency among online shoppers and the need for innovative tools to simplify the price comparison process.

5. AI-Driven Platform for Price Discrepancy Detection (M. J. Eide et al.): This work describes an AI-driven platform designed to detect price discrepancies across e-commerce platforms, leveraging web scraping and machine learning techniques.

6. Comparative Analysis of Pricing Strategies (Wiatrowski and Janocha): This study compares pricing strategies in the United States and the European Union, highlighting the impact of regional differences on consumer behavior and market dynamics.

7. Overview of Deep Learning in Neural Networks (Schmidhuber): This article provides a comprehensive overview of deep learning methodologies, which are foundational to modern AI-driven price comparison and product recognition systems.

8. Gradient-Based Learning Applied to Image Recognition (LeCun et al.): While focused on document recognition, this paper introduces gradientbased learning techniques, which are foundational to many machine learning algorithms and are highly relevant to AI-driven product recognition in ecommerce.

9. Web Scraping for Real-Time Data Retrieval (Tavily API Documentation): This source discusses the use of web scraping tools, such as the Tavily API, for retrieving real-time data from e-commerce websites, enabling accurate and up-to-date price comparisons.

CHAPTER - 3

DESIGN AND METHODOLOGY

3.1 ANALYSIS OF THE PROJECT

Detecting product availability and price comparisons across multiple ecommerce platforms using artificial intelligence (AI) and machine learning (ML) involves developing a system that can analyze user-uploaded images, recognize products, and retrieve real-time pricing data.

These images are preprocessed by resizing and normalizing pixel values to prepare them for input into the ResNet50 model, a deep learning architecture used for image recognition. Data augmentation techniques, such as rotation, flipping, and cropping, are employed to increase dataset variability and improve model robustness. The dataset is split into training, validation, and testing sets.

The ResNet-50 model is trained using appropriate loss functions and optimization algorithms, with training progress monitored using metrics like accuracy, precision, recall. After training, the model is evaluated on the test set to assess its performance in recognizing products and retrieving accurate pricing data. Metrics such as accuracy, precision, recall, and confusion matrix are analyzed to understand the model's strengths and weaknesses.

The system also incorporates the Tavily API for real-time web scraping and the ARIMA model for predicting price. The ultimate goal is to deploy the model in a user-friendly web application built using HTML, CSS, and Flask, hosted on Render. This application will allow users to upload product images, view real-time price comparisons, and make informed purchasing decisions.

Challenges such as dataset quality, model interpretability, data imbalance, and ethical considerations need to be addressed throughout the project to ensure its success and responsible implementation. The potential impact of ShopSense includes enabling cost-effective shopping, improving user satisfaction, and making price comparison more accessible and efficient. By continuously improving its AI models and expanding its database, ShopSense aims to revolutionize the way consumers interact with e-commerce platforms.

3.2 EXISTING SYSTEM

The conventional approaches to price comparison involve manual searches across multiple e- commerce websites, which are time-consuming, inefficient, and often inaccurate. Traditional methods for comparing prices rely on users visiting individual websites, checking product availability, and noting down prices. These methods have several limitations:

1. Time- Consuming Process: Manually searching for products and comparing prices across multiple platforms is tedious and inefficient.

2. Inconsistent Data: Prices and availability can vary significantly across websites, making it difficult for users to identify the best deal.

3. Lack of Real-Time Updates: Manual methods cannot provide real-time updates on price changes or stock availability.

4. Limited Scope: Users may not be able to compare prices across all available platforms due to time constraints.

5. User Frustration: The process often leads to frustration and dissatisfaction among users. Despite these challenges, recent advancements in AI and ML have significantly improved the efficiency and accuracy of price comparison tools.

3.3 PROPOSED SYSTEM

The ShopSense project aims to develop an efficient system for real-time price comparison and product availability tracking using advanced computer vision and deep learning techniques. The system utilizes user-uploaded images, which are processed using the ResNet-50 model for product recognition. The Tavily API is employed to scrape real-time pricing and availability data from multiple e-commerce websites.

Additionally, the ARIMA model is used to predict future price trends, helping users make informed purchasing. The system categorizes products and provides personalized feedback through a user-friendly interface, offering valuable insights and recommendations for cost-effective shopping. The web application is built using HTML, CSS, and Flask, ensuring a seamless and intuitive user experience.

Advantages of ShopSense:

1. High Accuracy in Product Recognition: The ResNet-50 model ensures accurate identification of products from user-uploaded images.
2. Real-Time Data Retrieval: The Tavily API provides up-to-date pricing and availability information from multiple platforms.
3. Cost Prediction: The ARIMA model predicts future price trends, helping users save money.
4. User-Friendly Interface: The web application is designed to be intuitive and easy to use.
5. Scalability: The system can handle a large number of users and products, ensuring broad applicability.
6. Accessibility: ShopSense makes price comparison.

3.4 FEASIBILITY STUDY

A feasibility study for ShopSense involves reviewing current technologies, establishing goals and criteria, and investigating methods like image recognition, web scraping, and price prediction. It evaluates their accuracy, cost-effectiveness, and adherence to technical and ethical standards. Consultation with stakeholders helps gather perspectives and inform recommendations for the most viable implementation strategy. Additionally, the study outlines future avenues for research and development to improve the system's functionality and user experience.

3.4.1 Technical Feasibility:

The technical feasibility of ShopSense involves assessing the practicality and effectiveness of using AI and ML technologies for product recognition, web scraping, and price prediction. This includes evaluating the capabilities of the ResNet-50 model, Tavily API, and ARIMA model for accurate and efficient performance.

Factors such as accuracy, reliability, speed, and integration with existing e-commerce platforms are considered. The study also investigates technical challenges related to data quality, computational requirements, and scalability, aiming to identify solutions to address these obstacles.

3.4.2 Operational Feasibility:

The operational feasibility for ShopSense evaluates the practicality of implementing the system in real-world scenarios. It assesses factors such as compatibility with existing e- commerce platforms, user acceptance, training requirements, resource availability, and workflow integration.

This analysis aims to determine whether the proposed system can be seamlessly integrated into users' shopping routines without causing disruption, while ensuring accurate and efficient price comparison.

CHAPTER - 4

SYSTEM SPECIFICATIONS

4.1 Functional Requirements:

The functional requirements for the ShopSense system are as follows:

- **Image Input and Processing:**
  - Accept Input of Product Images: The system should allow users to upload product images for analysis.
  - Implement Preprocessing Steps: Resize and normalize input images to standardize them for the ResNet50 model.

- **Dataset Handling:**
  - Curate a Diverse Dataset: Collect a comprehensive dataset of product images and their corresponding prices from various e-commerce platforms.
  - Split Dataset: Divide the dataset into training, validation, and testing sets for model training and evaluation. Architecture Selection:
  - Choose a Suitable CNN Architecture: Use the ResNet-50 model for product recognition due to its high accuracy and efficiency.
  - Implement Transfer Learning: Fine-tune the pre-trained ResNet-50 model on the product image dataset to improve performance.

- **Feature Extraction:**
  - Utilize Convolutional Layers: Automatically learn and extract relevant features from input product images.
  - Explore Additional Algorithms: Use dimensionality reduction techniques like PCA or t-SNE to enhance the model's performance.

- **Training and Optimization:**
  - Grid Search for Hyperparameter Tuning: Explore different hyperparameter settings to optimize the model's performance.
  - Early Stopping: Prevent overfitting by stopping training when validation performance plateaus.

- **Evaluation Metrics:**
  - Choose Appropriate Metrics: Use precision, recall, F1-score, AUC-ROC, and accuracy to evaluate the model's performance.
  - Implement Visualization Tools: Use tools like Grad-CAM or SHAP values to interpret and visualize the model's predictions.

- **User Interface:**
  - Develop a User-Friendly Interface: Create an intuitive web interface using HTML, CSS, and Flask for users to upload images and view results.
  - Ensure Seamless Integration: Integrate the system with existing e-commerce platforms and APIs.

- **Deployment:**
  - Select a Scalable Hosting Solution: Use Render for hosting the platform to ensure scalability and cost effectiveness.
  - Optimize for Efficiency: Ensure the system is optimized for fast inference and resource utilization. Security Measures:
  - Protect User Data: Implement security measures to safeguard user data and comply with data protection regulations.

- **Continuous Improvement:**

  Monitor and Update the System: Regularly update the model with new data and retrain it to maintain accuracy and relevance.

4.2 Non-Functional Requirements:

The non-functional requirements for the ShopSense system are as follows:

- **Performance:**
  - High Accuracy: The system should achieve high accuracy in product recognition and price comparison.
  - Fast Processing: The time required to process a single product image and retrieve results should be within acceptable limits.

- **Scalability:**
  - Handle Large Volumes of Data: The system should be designed to scale with increasing numbers of users and products.
  - Support Multiple Platforms: Ensure compatibility with various e-commerce platforms and devices. Reliability:
  - Consistent Performance: The system should deliver reliable and consistent results under varying conditions.
  - Error Handling: Implement robust error handling to manage exceptions and ensure smooth operation.

- **Usability:**
  - Intuitive Interface: The user interface should be easy to navigate and use, even for nontechnical users.
  - Accessibility: Ensure the platform is accessible to users with disabilities.

- **Security:**
  - Data Protection: Implement encryption and secure authentication mechanisms to protect user data.
  - Compliance: Ensure compliance with data protection regulations such as GDPR.

4.3 Hardware Requirements:

The hardware requirements for the proposed system are as follows:

1. **Computing Device**: The system should be compatible with standard computing devices such as desktop computers, laptops, or servers.
2. **Processing Power**: Sufficient processing power is necessary to handle the computational requirements of medical image analysis algorithms effectively.
3. **Memory**: Adequate RAM (Random Access Memory) is essential to store and manipulate large medical image datasets efficiently.
4. **Storage**: Sufficient storage space is required to store medical image files, as well as the system's software components and databases.
5. **Graphics Processing Unit (GPU):** A dedicated GPU may be beneficial for accelerating certain image processing tasks, especially deep learning-based algorithms.

4.4 Software Requirements

The software requirements for the proposed system are as follows:

1. **Operating System:** The system should be compatible with common operating systems such as Windows, macOS, or Linux.
2. **Development Environment**: Software development tools and frameworks are required for implementing and testing the medical image analysis algorithms, including IDEs (Integrated Development Environments) and libraries for image processing and machine learning.
3. **Database Management System (DBMS):** A DBMS is necessary to store and manage medical image data efficiently, enabling retrieval and manipulation of patient records and diagnostic results.
4. **Imaging Software**: Specialized imaging software may be required for viewing, processing, and annotating medical images, providing essential tools for medical professionals to analyze and interpret diagnostic data.
5. **Web Server:** If the system includes web-based components or interfaces, a web server is needed to host and serve web pages, APIs (Application Programming Interfaces), or other online resources.
6. **Security Software**: Security measures such as encryption, access controls, and authentication mechanisms are essential to protect sensitive patient data and ensure compliance with privacy regulations such as HIPAA (Health Insurance Portability and Accountability Act).

CHAPTER - 5

SOFTWARE DESIGN

5.1 PYTHON

Python stands out as a versatile and potent programming language renowned for its high-level, interpreted, interactive, and object-oriented characteristics. Designed with readability as a primary focus, Python extensively utilizes English keywords and employs fewer syntactical constructions compared to many other programming languages.

This emphasis on simplicity and clarity makes Python an accessible choice for developers across various domains, from web development to scientific computing and machine learning.

Key Characteristics of Python

Interpreted: Python's interpreted nature means that code is executed line by line by the Python interpreter, without the need for compilation into machine code beforehand. This makes development and testing quicker and more straightforward.

Dynamically Typed: Python is dynamically typed, meaning that variable types are determined at runtime. This flexibility allows for faster development and easier code maintenance but may lead to potential errors if not carefully managed.

High-level: Python is a high-level programming language, which means it abstracts away low- level details like memory management and hardware interaction. This makes Python code more readable, writable, and maintainable.

Cross-platform: Python is supported on various operating systems, including Windows, macOS, and Linux, making it highly versatile and accessible across different environments.

Large Standard Library: Python comes with a vast standard library that provides support for a wide range of tasks, from file I/O and networking to GUI development and web services.

Extensible: Python can be easily extended with modules and packages written in other languages like C and C++, allowing developers to leverage existing libraries and integrate with other systems seamlessly.

History of Python

Guido van Rossum initiated the development of Python in the late 1980s as a successor to the ABC programming language, driven by a vision to create a programming language that prioritized intuitiveness, power, and ease of learning. His aim was to design a language that would cater to both novice programmers and seasoned developers alike.

Python made its debut in 1991 with the release of version 0.9.0, marking the beginning of its journey as an open-source programming language. This initial version laid the groundwork for subsequent iterations, paving the way for Python's evolution into a robust and versatile language.

The official release of Python 1.0 followed in 1994, signifying a significant milestone in Python's development. With version 1.0, Python transitioned from an experimental project to a fully-fledged programming language, complete with comprehensive documentation and support for essential features.

Over the years, Python has undergone numerous major releases, each introducing new features, enhancements, and optimizations. The Python community actively contributes to the language's evolution through the Python Enhancement Proposal (PEP) process, where proposals for changes to the language are discussed, reviewed, and either accepted or rejected based on consensus.

Python experienced a surge in popularity during the early 2000s, fueled by its simplicity, readability, and versatility. Its intuitive syntax and extensive standard library made it an attractive choice for a wide range of applications, from web development and data analysis to artificial intelligence and scientific computing.

Guido van Rossum assumed the role of Python's Benevolent Dictator For Life (BDFL), providing leadership and guidance to the Python community. However, in 2018, van Rossum stepped down from his position as BDFL, entrusting the future of Python to the community while continuing to contribute to its development as a respected figure and mentor.

Python features:

Python, a dynamically typed, high-level programming language, boasts an array of features that contribute to its widespread popularity and versatility across various domains. Let's delve into the key features of Python and explore some additional capabilities that make it a preferred choice for developers worldwide:

Core Features of Python

1. **Easy-to-Learn:** Python's simplicity lies in its concise syntax, minimalistic structure, and limited number of keywords, making it an ideal language for beginners to grasp quickly and start coding.

2. **Easy-to-Read:** Python prioritizes readability, with code that resembles plain English, enhancing comprehension and reducing the cognitive load for developers.

3. **Easy-to-Maintain:** Python emphasizes clean, organized code, facilitating easier debugging, maintenance, and updates, thus minimizing the overhead associated with software maintenance.

4. **Broad Standard Library:** Python ships with an extensive standard library, offering a plethora of modules and functions for performing diverse tasks, from file I/O and networking to data manipulation and web development.

5. **Interactive Mode:** Python supports an interactive mode interpreter, enabling developers to experiment, test, and debug code snippets interactively, fostering a more iterative and exploratory programming approach.

6. **Portable:** Python's platform-independent nature allows code written in Python to run seamlessly across different operating systems, including UNIX, Windows, and macOS, ensuring consistency and compatibility.

7. **Extendable:** Python's extensibility enables developers to integrate low-level modules written in languages like C and C++ into their Python codebase, empowering them to optimize performance and functionality as needed.

8. **GUI Programming**: Python supports GUI development through frameworks like Tkinter, PyQt, and wxPython, enabling the creation of visually appealing and interactive desktop applications across multiple platforms.

9. **Scalable:** Python's modular architecture and support for large-scale software engineering make it well-suited for building scalable applications that can grow and adapt to evolving requirements with ease.

Additional Features of Python

1. **Support for Multiple Programming Paradigms**: Python supports functional, structured, and object-oriented programming paradigms, offering developers flexibility in choosing the most suitable approach for their projects.

2. **Scripting and Compilation:** Python serves as both a scripting language, allowing for rapid prototyping and development, and a compiled language, capable of generating bytecode for building robust, production-ready applications.

3. **Dynamic Data Types:** Python provides high-level, dynamic data types such as lists, dictionaries, and tuples, along with dynamic type checking, simplifying data manipulation and enhancing flexibility in handling diverse data structures.

4. **Automatic Garbage Collection:** Python's automatic garbage collection mechanism frees developers from manual memory management tasks, reducing the risk of memory leaks and improving application stability.

5. **Interoperability:** Python seamlessly integrates with other programming languages and technologies, including C, C++, COM, ActiveX, CORBA, and Java, enabling interoperability and facilitating the reuse of existing codebases and libraries.

Overall, Python's rich feature set, combined with its simplicity, readability, and versatility, makes it a preferred choice for a wide range of applications, from web development and data analysis to artificial intelligence and scientific computing.

Python is a versatile programming language with a wide range of capabilities, making it a popular choice for various applications. Here's an overview of what Python can do:

Web Development:

Python can be used to develop web applications on server-side frameworks like Django, Flask, and Pyramid.

With these frameworks, developers can create dynamic and interactive websites, web services, and APIs.

Software Development:

Python is often used alongside other software tools and libraries to create workflows, automate tasks, and develop software applications.

Its simplicity and readability make it suitable for rapid development and prototyping.

Database Connectivity:

Python offers robust support for connecting to and interacting with various database systems, including MySQL, PostgreSQL, SQLite, and MongoDB.

Developers can use Python to read and modify database records, execute queries, and manage data effectively.

Data Handling:

Python excels in handling data, from small-scale data processing tasks to big data analytics and machine learning.

Libraries like NumPy, pandas, and SciPy provide powerful tools for data manipulation, analysis, and visualization.

Prototyping and Production:

Python's ease of use and quick development cycle make it suitable for rapid prototyping of ideas and concepts.

Additionally, Python's robustness and scalability make it equally suitable for developing production-ready software applications.

Platform Agnostic:

Python is compatible with various operating systems and platforms, including Windows, macOS, Linux, and even embedded systems like Raspberry Pi.

Simple Syntax:

Python's syntax is designed to be easy to read and understand, resembling the English language and mathematical expressions.

C**oncise Code:**

Python allows developers to write expressive code with fewer lines compared to other programming languages, enhancing readability and maintainability.

Interpreted Language:

Python is an interpreted language, meaning code is executed line by line by an interpreter, enabling quick execution and rapid prototyping.

Good to Know

Python Versions:

Python 3 is the latest major version and is recommended for new projects. However, Python 2 is still in use, particularly in legacy systems, although it is no longer actively maintained except for security patches.

Development Environment:

Python code can be written in a simple text editor or a more feature-rich Integrated Development Environment (IDE) like Thonny, PyCharm, NetBeans, or Eclipse.

Python Syntax Compared to Other Languages

Readability:

Python prioritizes readability, with syntax influenced by natural language and mathematics, making it easier for developers to write and understand code.

Line Completion:

Python uses new lines to complete commands, unlike other languages that may rely on semicolons or parentheses for statement termination.

Python Installation

Installing Python on a Windows machine is a straightforward process that enables users to quickly set up their programming environment. Here's a detailed guide on how to install Python on your Windows system:

1. **Open a Web Browser**: Launch your preferred web browser and navigate to the official Python website by entering the URL https://www.python.org/downloads/ in the address bar.

2. **Download Python Installer**: On the Python Downloads page, you'll find various versions of Python available for download. Choose the version that best suits your requirements by clicking on the link corresponding to the Windows installer labeled "python-XYZ.msi," where XYZ represents the version number. For example, if you want to install Python 3.9.7, you would select the link for "python-3.9.7.msi".

3. **Check Compatibility:** Before proceeding with the download, ensure that your Windows system supports Microsoft Installer 2.0 or later, as the Python installer file (.msi) relies on this component for installation. Most modern Windows systems meet this requirement, but it's essential to verify compatibility to avoid any issues during installation.

4. **Download Installer:** Once you've selected the appropriate installer link, click on it to initiate the download process. Your browser will prompt you to choose a location to save the installer file. Select a location on your local machine where you can easily locate the file, such as the Downloads folder, and then click "Save" or "Download."

5. **Run Installer:** After the download is complete, navigate to the location where you saved the Python installer file (python-XYZ.msi). Double-click on the installer file to launch the Python Install Wizard. If prompted by the User Account Control (UAC) dialog, click "Yes" to allow the installer to make changes to your system.

6. **Begin Installation:** The Python Install Wizard will guide you through the installation process. Follow the on-screen instructions to proceed with the installation.

7. **Customize Installation (Optional):** During the installation process, you may be given the option to customize certain aspects of the Python installation, such as the installation directory or the addition of Python to the system PATH. If you're unsure, it's generally safe to stick with the default settings.

8. **Wait for Installation:** Once you've confirmed your preferences, click "Install" to begin the installation process. The installer will extract and copy the necessary files to your system, so it may take a few moments to complete. Be patient and wait for the installation progress bar to reach 100%.

9. **Completion:** Once the installation is complete, you'll see a message indicating that Python has been successfully installed on your Windows machine. You can now close the Python Install Wizard and exit the installer.

10. **Verify Installation:** To ensure that Python was installed correctly, open a Command Prompt window by searching for "cmd" in the Windows Start menu. In the Command Prompt, type "python --version" (without quotes) and press Enter. If Python is installed correctly, you should see the version number of the Python interpreter installed on your system.

Python Programming Modes:

Python offers two primary modes of programming: Interactive Mode and Script Mode, each serving different purposes and providing distinct benefits.

1. **Interactive Mode Programming:**

   In interactive mode, you can launch the Python interpreter without passing a script file as a parameter. This opens an interactive prompt where you can execute Python code interactively.

   To enter interactive mode, simply open your command-line interface and type`python`, then press Enter. This will launch the Python interpreter and display version information along with a prompt (`>>>`) indicating that Python is ready to accept commands.

Here's an example of using Python in interactive mode:

```
$ python
Python 3.9.7 (default, Sep  3 2021, 13:42:55)
[GCC 10.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

You can directly enter Python statements and expressions at the prompt and receive immediate feedback.

```
>>> print("Hello, Python!")
Hello, Python!
```

Interactive mode is particularly useful for quick experimentation, testing code snippets, and learning Python syntax and features interactively.

2. Script Mode Programming:

Script mode involves running the Python interpreter with a script file as a parameter.

This allows you to execute Python scripts containing multiple statements or functions.

To write and execute a Python script, create a new file with a `.py` extension (e.g.,`test.py`) and add your Python code to it.

Here's an example of a simple Python script (`test.py`):

```python
print("Hello, Python!")
```

To run the script, open your command-line interface, navigate to the directory containing the script file, and execute the following command:

```
$ python test.py
```

This will execute the Python script and produce the output specified in the script:

```
Hello, Python!
```

Script mode is ideal for developing larger Python programs, organizing code into reusable modules, and automating tasks by running scripts from the command line or scheduling them with cron jobs on Unix-like systems

- .

By leveraging both interactive mode and script mode, Python provides developers with a flexible and powerful environment for writing, testing, and executing Python code for a wide range of applications and purposes.

Python is a widely-used programming language created in 1991 by Guido van Rossum. It finds applications in various domains such as web development (server-side), software development, mathematics, and system scripting.

Python Indentations

Python uses indentation and whitespace to define code structure and scope, promoting clean and organized code. In contrast, other languages often use curly brackets for this purpose.

Indentation plays a crucial role in Python as it signifies blocks of code. Unlike other languages where indentation is for readability only, Python relies on it for defining scope.

Here's an example:

```python if 5 > 2:

    print("Five is greater than two!")

```

Comments and Docstrings

Python supports comments and docstrings for in-code documentation:

- **Comments**: Start with `#`, and Python ignores the rest of the line.

```python

# This is a comment. print("Hello, World!")

```

- **Docstrings**: Extended documentation enclosed in triple quotes (`""" """`).

```python """This is a

multiline docstring.""" print("Hello, World!")

```

These features enhance code readability and documentation in Python, making it easier for developers to understand and maintain codebases.

OVERVIEW

Dynamic Typing: Python is dynamically typed, enabling automatic assignment of data types to variables based on their values.

Extensive Standard Library: Python boasts a vast standard library, offering modules and packages for a wide array of tasks, including file I/O, networking, web development, and more.

Vibrant Community and Ecosystem: With a supportive community, Python's ecosystem is rich with libraries, frameworks, and tools like NumPy, pandas, TensorFlow, Django, and Flask, catering to various domains such as data science, machine learning, and web development.

Cross-Platform Compatibility: Python code is highly portable and can run seamlessly across different operating systems, including Windows, macOS, Linux, and UNIX-like systems.

Strong Integration Support: Python integrates effortlessly with other languages like C/C++, Java, and .NET, enabling developers to leverage existing codebases and libraries through APIs and language bindings.

Documentation and Community Resources: Python offers extensive documentation, tutorials, and community forums like Stack Overflow, facilitating learning and collaboration within the Python community.

Versatility and Scalability: Python's versatility enables its use in diverse applications, from scripting and automation to building large-scale web applications and scientific simulations.

Open Source and Free: As an open-source language with a permissive license, Python allows developers to use, modify, and distribute it freely, fostering innovation and widespread adoption.

5.2 FLASK

Flask stands as a testament to simplicity and elegance in web development within the Python ecosystem. Developed by Armin Ronacher and maintained by a dedicated group of Python enthusiasts known as Pocco, Flask embodies the philosophy of minimalism while providing powerful features for building web applications.

As a micro web framework, Flask is designed to be lightweight and flexible, offering developers the freedom to choose the tools and libraries that best suit their project requirements. Unlike full-stack frameworks that come bundled with built-in functionalities like database abstraction layers and form validation, Flask takes a minimalist approach, providing only the essentials needed for web development.

Built on top of the Werkzeug WSGI toolkit and the Jinja2 template engine, both of which are also Pocco projects, Flask leverages these robust foundations to deliver a seamless and efficient web development experience. Despite its minimalistic nature, Flask is highly extensible, allowing developers to integrate additional features and functionalities through a wide range of extensions available in the Flask ecosystem.

With Flask, developers have the freedom to tailor their web applications according to their specific needs, whether it's implementing object-relational mappers, handling form validation, or integrating various authentication technologies

Key Features of Flask

Micro Framework: Flask is classified as a microframework, emphasizing simplicity and minimalism. It provides only the essential components needed for web development, allowing developers to add functionalities as needed.

Modularity: Flask follows a modular design, allowing developers to pick and choose the components they need for their projects. This modular approach enhances flexibility and scalability, enabling developers to customize their applications according to specific requirements.

Extensibility: While Flask itself is lightweight, it supports a rich ecosystem of extensions that can be easily integrated into applications. These extensions add functionalities such as database integration, form validation, authentication, and more, expanding Flask's capabilities without compromising its simplicity.

Built-in Development Server: Flask comes with a built-in development server, allowing developers to test and debug their applications locally before deploying them to production environments. This feature streamlines the development process and facilitates rapid iteration.

Jinja2 Templating: Flask utilizes the Jinja2 template engine for generating dynamic content in web applications. Jinja2's syntax is intuitive and powerful, enabling developers to create reusable templates and maintain clean, readable code.

Werkzeug WSGI Toolkit: Flask is built on top of the Werkzeug WSGI toolkit, which provides a robust foundation for handling HTTP requests and responses. This toolkit simplifies the implementation of web servers and middleware.

Advantages of Using Flask

Simplicity: Flask's minimalist design and intuitive syntax make it easy for developers to get started with web development. Beginners can quickly grasp Flask's concepts and build functional web applications with minimal boilerplate code.

Flexibility: Flask's architecture allows developers to adapt framework to suit their specific needs. Whether building simple microservice or complex web application, provides the flexibility to tailor solutions according to project requirements.

HTTP Protocol

The Hypertext Transfer Protocol (HTTP) is the foundation of data communication on the World Wide Web. It facilitates the exchange of data between a client and a server over the internet. HTTP defines a set of rules and standards for transmitting and receiving hypertext messages, enabling the retrieval and manipulation of web resources.

Different HTTP Methods

GET: The GET method requests data from a specified resource. It sends data to the server in unencrypted form, typically through query parameters appended to the URL. GET requests are commonly used for retrieving web pages, images, and other resources from servers.

HEAD: Similar to the GET method, but without the response body. It retrieves metadata about a resource without transferring the entire content. HEAD requests are useful for checking resource availability, determining content length, and retrieving HTTP headers.

POST: The POST method submits data to be processed to a specified resource, often via HTML forms on web pages. Unlike GET requests, POST data is sent in the message body, making it suitable for transmitting sensitive information such as passwords or credit card details. POST requests are not cached by the server, enhancing security and privacy.

PUT: The PUT method replaces all current representations of the target resource with the uploaded content. It is commonly used for updating or creating resources on the server. PUT requests are idempotent, meaning that multiple identical requests have the same effect as a single request, ensuring predictable behavior.

DELETE: The DELETE method removes all current representations of the target resource specified by a URL. It is used to delete resources from the server permanently. DELETE requests are also idempotent, allowing clients to safely repeat requests without causing unintended side effects.

Flask Routing

Flask routing is a fundamental aspect of building web applications with Flask. It allows developers to map URL paths to specific functions within a Flask application, enabling the handling of HTTP requests and generating appropriate responses. This mechanism forms the backbone of routing and navigation in Flask-based web applications.

Routing and HTTP Methods

By default, Flask routes respond to GET requests, which are commonly used for retrieving data from a specified URL. However, Flask provides flexibility in handling other HTTP methods such as POST, PUT, DELETE, and more. Developers can specify multiple HTTP methods for a single route using the **methods** argument in the **route()** decorator.

Example Demonstrating the Use of POST Method in URL Routing:

1. **Creating an HTML Form (login.html)**:

```
<html>
<body>
<form action="http://localhost:5000/login" method="post">
<p>Enter Name:</p>
<p><input type="text" name="nm"/></p>
<p><input type="submit" value="Submit"/></p>
</form>
</body>
</html>
```

2. Implementing a Flask App (app.py):

```
from flask import Flask, redirect, url_for, request app = Flask(_name_)

@app.route('/success/<name>') def success(name):

   return 'Welcome %s' % name @app.route('/login', methods=['POST', 'GET'])

def login():

   if request.method == 'POST': user = request.form['nm']

      return redirect(url_for('success', name=user)) else:

      user = request.args.get('nm')

      return redirect(url_for('success', name=user)) if _name___== '___main__':

app.run(debug=True)
```

This Flask app (app.py) manages user authentication and redirection. It defines two routes: /success/<name>, which displays a personalized welcome message, and /login, which processes GET and POST requests to capture user input. The username is retrieved from form data or query parameters, then redirected to the success page.

The app runs in debug mode, enabling efficient debugging, error tracking, and real-time code modifications. Flask's redirect() and url_for() functions handle smooth navigation between routes. This simple implementation demonstrates handling user input, request methods, and redirection in a web application using Flask.

3. **Execution**:

- After starting the Flask development server, open **login.html** in a browser, enter a name in the text field, and click Submit. This will redirect to the success page with the entered name displayed.

  Explanation:

- When form data is POSTed to the URL specified in the action attribute of the form tag (**http://localhost/login**), it is mapped to the **login()** function in the Flask application.
- If the server receives data through the POST method, the value of the 'nm' parameter obtained from the form data is retrieved using **request.form['nm']**.
- This value is then passed to the '/success' URL as a variable part, and the browser displays a welcome message in the window utilizing the entered name from the form data.
- If the method parameter in **login.html** is changed to 'GET', the data received on the server is through the GET method. In this case, the value of the 'nm' parameter is obtained using **request.args.get('nm')**, and the subsequent process remains the same.

Fig 5.2.1 Flask app demo

5.3 Technologies Used

5.3.1 NumPy

NumPy is a Python library designed to enhance the language's capabilities by introducing support for extensive, multidimensional arrays and matrices. It comes with a rich collection of high-level mathematical functions specifically crafted to operate seamlessly on these arrays. The precursor to NumPy, known as Numeric, was initially developed by Jim Holguin, with contributions from various developers. In 2005, Travis Oliphant played a pivotal role in creating NumPy by amalgamating features from the competing Numb array into Numeric, incorporating significant modifications.

Being open-source, NumPy has benefited from contributions by a diverse group of developers. It primarily targets the Python reference implementation—a non-optimizing bytecode interpreter. Python's inherent slowness, especially in executing mathematical algorithms, is mitigated by NumPy. The library achieves this by introducing multidimensional arrays, along with efficient functions and operators that operate on arrays. This often necessitates rewriting certain code sections, particularly inner loops, using NumPy.

Utilizing NumPy in Python provides functionality comparable to MATLAB, as both are interpreted languages enabling users to write swift programs, given that most operations involve arrays or matrices rather than scalars. Although MATLAB offers various additional toolboxes, such as Simi link, NumPy is inherently integrated with Python, a more modern and comprehensive programming language. Additionally, complementary Python packages like SciPy and Matplotlib extend NumPy's capabilities. SciPy introduces MATLAB-like functionality, while Matplotlib serves as a plotting package providing similar capabilities.

The core functionality of NumPy revolves around its "array," a data structure supporting n- dimensional arrays. These arrays provide a stripped view on memory. In contrast to Python's built-in list data structure, NumPy arrays are homogeneously typed, requiring all elements within a single array to be of the same type.

5.3.2 Open CV

OpenCV, an acronym for Open-Source Computer Vision Library, stands as a robust and widely embraced open-source tool tailored to meet a diverse array of needs in computer vision and image processing applications. Equipped with a broad spectrum of features, OpenCV streamlines tasks like image and video analysis, object recognition, camera calibration, and more. Originally coded in C++, its versatility extends to multiple programming languages, including Python, Java, and MATLAB, ensuring accessibility to a broad user base.

The cross-platform compatibility of OpenCV adds to its appeal, allowing developers to work seamlessly across various operating systems, including Windows, Linux, macOS, iOS, and Android. OpenCV's offerings encompass an extensive array of functionalities, spanning image processing techniques, computer vision algorithms, and integration with machine learning libraries. This comprehensive toolkit positions OpenCV as an invaluable resource for researchers, developers, and engineers working in diverse industries.

OpenCV's success is further amplified by an active and supportive community, complemented by thorough documentation and tutorials. This combination solidifies OpenCV's status as a preferred solution for tackling computer vision challenges in both academic and industrial spheres.

5.3.3 Tensor Flow

Tensor Flow, originating from the Google Brain team, stands as an open- source machine learning framework that has garnered recognition for its prowess in constructing and deploying machine learning models. Esteemed for its flexibility and scalability, TensorFlow has emerged as a fundamental tool in the realm of deep learning, focusing primarily on neural networks. Its applications span diverse domains, ranging from image and speech recognition to natural language processing.

A distinctive feature of Tensor Flow lies in its adaptability across various domains and platforms. It boasts compatibility with multiple programming languages, including Python and C++, ensuring accessibility to a wide developer base. Tensor Flow's inclusive high-level APIs, such as Keras, streamline the implementation of intricate neural network architectures, catering to both newcomers and experienced practitioners.

Tensor Flow's efficient utilization of hardware resources sets it apart. It supports acceleration through graphics processing units (GPUs) and tensor processing units (TPUs), optimizing the performance of deep learning models. This adaptability in hardware compatibility enables Tensor Flow to seamlessly scale from individual machines to extensive distributed systems.

Moreover, Tensor Flow's robust community and comprehensive documentation contribute to its widespread adoption. Developers benefit from an abundance of resources, including tutorials, guides, and an engaged community that actively shares insights and best practices. Tensor Flow's commitment to open-source collaboration ensures continuous enhancements, keeping it at the forefront of machine learning advancements.

In the dynamic landscape of machine learning, Tensor Flow maintains its prominence by adapting to evolving challenges and technologies. Its amalgamation of versatility, scalability, and community support positions TensorFlow as a leading framework, driving innovation and practical machine learning solutions across a spectrum of applications and industries

CHAPTER - 6

SYSTEM DESIGN

6.1 Introduction about system design

In the context of software, design is a problem-solving process whose objective is to find and describe a way to implement the system's functional requirements while respecting the constraints imposed by the non-functional requirements (including the budget and deadlines) and by adhering to general principles of good quality.

Software design is the process of planning a new system to replace or complement an existing system. It is an iterative process through which requirements are translated into the software. During this stage, analyst works with the user to develop a physical model of the system. Software design is an interactive process through which the requirements are translated into a "blueprint" for constructing the software.

6.1.1 UML Diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling 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 is comprised of 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 Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Building Blocks of UML

The vocabulary of the UML encompasses three kinds of building blocks:

- Things
- Relationships
- Diagrams

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

Things in the UML

There are four sorts of things in UML. They are Structural Things:

Class

A Class is a set of identical things that outlines the functionality and properties of an object. It also represents the abstract class whose functionalities are not defined. Its notation is as follows

Object

An individual that describes the behavior and the functions of a system. The notation of the object is similar to that of the class.

Interface

A set of operations that describes the functionality of a class, which is implemented whenever an interface is implemented.

Collaboration

It represents the interaction between things that are done to meetthe goal. It is symbolized as a dotted ellipse with its name written inside it.

Use Case

Use case is the core concept of object-oriented modeling. It portrays aset of actions executed by a system to achieve the goal.

Activity Diagram

It portrays all the activities accomplished by different entities of a system. It is represented the same as that of a state machine diagram. It consists of an initial state, final state, a decision box, and an action notation.

Grouping Things

It is a method that together binds the elements of the UML model. In, the package is the only thing which is used for grouping.

Package

Package is the only thing that is available for grouping behavioral and structural things

Annotation Things

A set of links that associates the entities to the UML model. It tells how many elements are actually taking part in forming that relationship.

Association

It is denoted by a dotted line with arrowheads on both sides to describe the relationship with the element on both sides.

Generalization

It portrays the relationship between a general thing (a parent class or super class) and a specific kind of that thing It is used to describe the concept of inheritance. It is denoted by a straight line followed by an empty arrowhead at one side.

Realization

It is a semantic kind of relationship between two things, where one defines the behavior to be carried out, and the other one implements the mentioned behavior. It exists in interfaces. It is denoted by a dotted line with an empty arrowhead at one side.

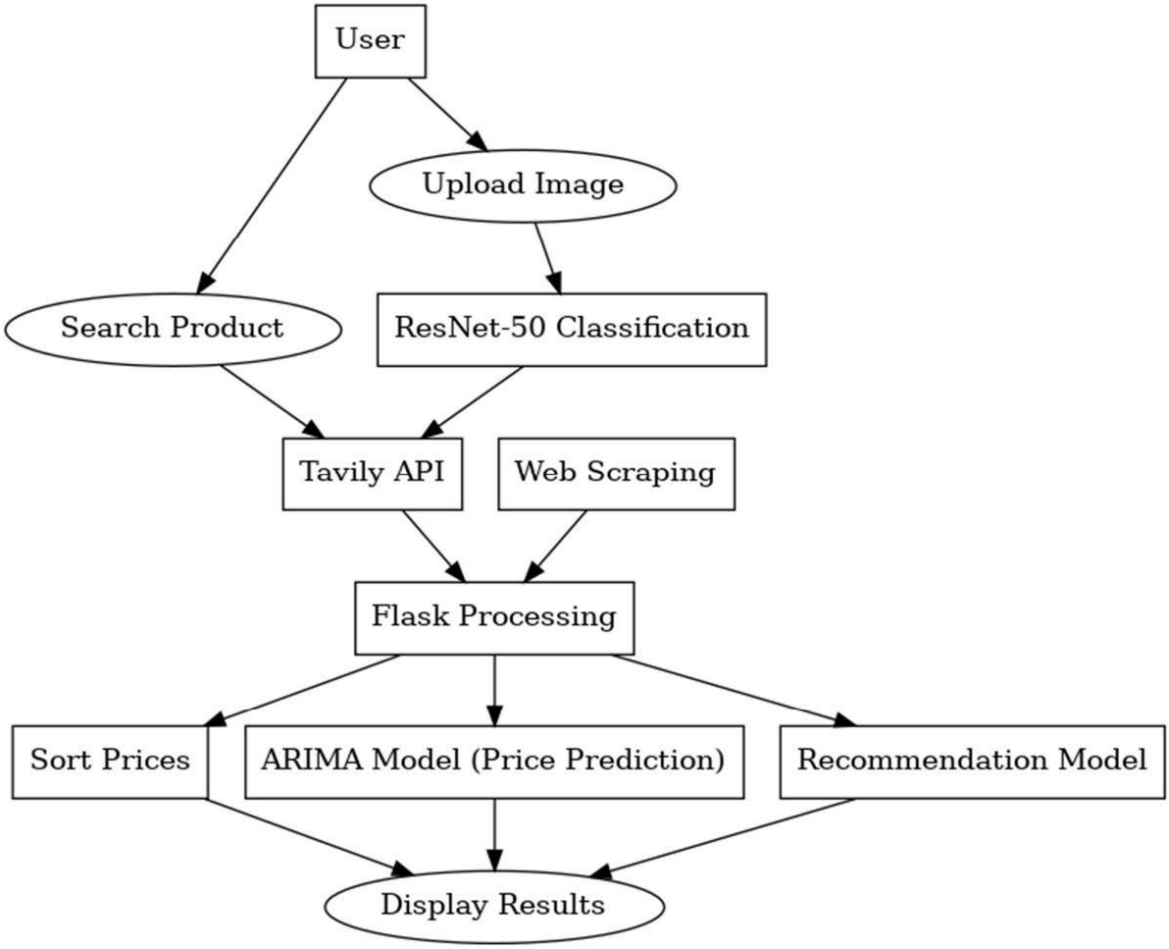

Fig 6.1.1 UML Diagram of ShopSense

6.2 SYSTEM IMPLEMENTATION

Implementation of ShopSense The development and deployment of ShopSense involve a structured approach comprising several key phases, ensuring a seamless and efficient system. Below is an extended breakdown of the implementation process:

- Design a user-friendly interface using HTML, CSS, and JavaScript to facilitate smooth interaction.
- Implement React.js or Vue.js for a dynamic and responsive user experience.

- Ensure cross-platform compatibility so that the interface works efficiently on desktops, tablets, and mobile devices.
- Enable users to upload images, search for products, and view price trends through an intuitive UI.
- Implement loading indicators, error messages, and interactive elements for better user engagement.
- Develop a Flask-based backend to handle user requests and facilitate communication between various system components.
- Implement RESTful APIs to manage requests, process images, and interact with AI/ML models.
- Ensure secure data handling by implementing authentication and authorization mechanisms.
- Optimize backend logic to minimize response time and improve processing efficiency.
- Enable logging and error handling to troubleshoot issues effectively.
- **ResNet-50 Model Integration**: Implement deep learning-based product recognition to classify uploaded images and match them with relevant products. Pre-train and fine-tune the ResNet-50 model for better accuracy in identifying various product categories.
- **Tavily API for Real-Time Web Scraping**: Leverage the Tavily API to extract real-time price and availability data from multiple e-commerce platforms such as Amazon, Flipkart, and Reliance Digital. Implement data filtering and cleaning techniques to ensure accuracy and consistency.
- **ARIMA Model for Price Prediction**: Use the AutoRegressive Integrated Moving Average (ARIMA) model to analyze historical price trends and forecast future price variations.
- Choose a suitable database management system (DBMS) such as MySQL, PostgreSQL, or MongoDB to store structured and unstructured data.
- **Store and manage the following data**: User information (profiles, search history, uploaded images). Product details (names, descriptions, images, category mappings). Historical pricing data for trend analysis.

- Optimize database queries for fast retrieval and scalability.
- Implement data encryption and security policies to safeguard user information.
- Conduct extensive testing using a diverse dataset of product images and price data to evaluate system performance.
- **Perform the following testing strategies:**

  Unit testing for individual components (e.g., API endpoints, ML models, database queries).

  Integration testing to ensure seamless interaction between the frontend, backend, and AI models.

  Performance testing is to evaluate system response time under varying loads.

  User acceptance testing (UAT) to validate the overall user experience and functionality.

- Monitor system accuracy in product recognition, web scraping, and price prediction, and fine-tune the models as needed.

6.3 SYSTEM ARCHITECTURE

ShopSense's architecture is designed to be modular and scalable, allowing for future enhancements and integrations. The system is composed of several key components that work together to provide a seamless price comparison experience.

- **Frontend (User Interface):**
  - Built using HTML, CSS, and JavaScript, the frontend provides a userfriendly interface for image uploads and result displays.
  - It is designed to be responsive, ensuring accessibility across various devices. o The frontend interacts with the backend via RESTful APIs.
- **Backend (Flask Server):**
  - coordinating the interactions between different component Database Integration Testing and Validation It handles image processing, API calls, and data management. It exposes APIs for the frontend to communicate with. It also handles the logic for data storage and retrieval from the database.
- **AI/ML Models:**
  - ResNet-50 Model: Pre-trained on a large image dataset, it is used for product recognition. It receives image data from the Flask server and returns product identification results. The model has been optimized for speed and accuracy.
- **Tavily API:**
  - Used for real-time price scraping from various online retailers. It receives product names from the Flask server and returns current price information. It enables the application to deliver up to date pricing.
- **ARIMA Model:**
  - Employed for price trend prediction based on historical data. It receives price history data from the database and returns predicted future prices. This allows users to make informed purchasing decisions.

CHAPTER – 7

IMPLEMENTATION

7.1 Frontend using HTML, CSS

7.1.1 Index Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ShopSense</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0;
            padding: 0;
            background-color:rgb(154, 159, 213);
            background-size: cover;
        }
        .navbar {
            background-color: rgb(24, 139, 227);
            color: white;
            padding: 1rem;
            display: flex;
            justify-content: space-between;
            align-items: center;
        }
    </style>
</head>
<body>
    <nav class="navbar">
        <div class="logo-container">
            <div class="circle-icon">
            </div>
            <span class="logo-text">ShopSense</span>
        </div>
        <div class="nav-links">
            <a href="aboutus.html">About us</a>
            <a href="#">Deals</a>
            <a href="#">Notifications</a>
                <a href="#">Contact Us</a>
        </div>
    </nav>
```

```
    <section class="search-section">
      <form action="/search" method="POST" enctype="multipart/form-data">
        <input type="text" name="product_name" placeholder="Search for products..."
       class="search-bar" >
        <input type="file" name="product_image" accept="image/*" class="image-input">
        <button type="submit" class="search-button">Search</button>
      </form>
    </section>
</body>
      </html>
```

7.2.2 Prediction Results Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Predicted Prices</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0;
            padding: 0;
            background-color:rgb(154, 159, 213);
            background-size: cover;
        }
    </style>
</head>
<body>
    <nav class="navbar">
        <div class="logo-container">
            <div class="circle-icon">

                <span>□</span>
            </div>
            <span class="logo-text">ShopSense</span>
        </div>
        <div class="nav-links">
            <a href="aboutus.html">About us</a>
            <a href="#">Deals</a>
            <a href="#">Notifications</a>
```

```
            <a href="#">Contact Us</a>
        </div>
    </nav>

    <div class="container">
        <h1>Predicted Prices for "{{ query }}"</h1>
        <div class="graph-container">
            <img src="{{ plot_base64 }}" alt="Predicted Prices Plot">
        </div>
        <a href="/" class="back-button">Back to Search</a>
    </div>
</body>
    </html>
```

7.3.3 Recommendations Page

```
<!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>ShopSense</title>
        <link rel="stylesheet" href="/static/styles.css">
        <style>
            body {
                font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
                margin: 0;
                padding: 0;
                background-color:rgb(154, 159, 213);
                background-size: cover;
            }
        </style>
    </head>
    <body>
        <nav class="navbar">
            <div class="logo-container">
                <div class="circle-icon">
                    <span>□</span>
                </div>
                <span class="logo-text">ShopSense</span>
            </div>
            <div class="nav-links">
                <a href="#">About us</a>
```

```
            <a href="#">Deals</a>
            <a href="#">Notifications</a>
            <a href="#">Contact Us</a>
        </div>
    </nav>
    <section class="recommendations-section">
        <ul>
            {% for i in range(variants|length) %}
                <li><a href="{{ links[i] }}" target="_blank">{{ variants[i] }}</a></li>
            {% endfor %}
        </ul>
        <a href="/">Go Back</a>
    </section>
</body>
    </html>
```

7.2 Python codes

7.2.1 App.py

```
from flask import Flask, render_template, request
from langchain_community.tools.tavily_search.tool
import TavilySearchResults
from langchain_community.utilities.tavily_search import
TavilySearchAPIWrapper
from product_recognition import
recognize_product_from_image
import os
import pandas as pd
import numpy as np
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt
import io
import base64
from Product_recommendation import
generate_product_variants
# Use the 'Agg' backend for Matplotlib
plt.switch_backend('Agg')

app = Flask(__name__)

# Initialize the API wrapper
api_wrapper =
TavilySearchAPIWrapper(tavily_api_key="tvly-
XR70WkWVUfIRwgd97uHOgh2WuYYsy928")
tool = TavilySearchResults(api_wrapper=api_wrapper)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/search', methods=['POST'])
def search():
    product_name = request.form.get('product_name', '')
    image = request.files.get('product_image')

    if product_name:  # Text-based search
        results = tool.invoke(f"""Search for
"{product_name}" across the following websites:
Flipkart, Amazon, Myntra, Reliance Digital, Snapdeal,
and Croma. For each result, return:
        The URL of the product page. Sort the results by
price in ascending order. Ensure the URLs are valid and
the prices are accurate for the specified product.
        """)
        print("Search Results for text:", results)
        return render_template('searchresults.html',
results=results, query=product_name)

    elif image:  # Image-based search
        if image.filename == '':
            return "No selected file"

        # Ensure the 'static/uploads' directory exists
        upload_dir = os.path.join('static', 'uploads')
        if not os.path.exists(upload_dir):
            os.makedirs(upload_dir)

        image_path = os.path.join(upload_dir,
image.filename)
        image.save(image_path)  # Save uploaded image

        recognised_product_name =
recognize_product_from_image(image_path)
        print("Recognized Product:",
recognised_product_name)

        results = tool.invoke(f"""Search for
"{recognised_product_name}" across the following
websites: Flipkart, Amazon, Myntra, Reliance Digital,
Snapdeal, and Croma. For each result, return:
        The URL of the product page. Sort the results by
price in ascending order. Ensure the URLs are valid and
the prices are accurate for the specified product.
        """)
        print("Search Results for image:", results)
        return render_template('searchresults.html',
results=results, query=recognised_product_name)

    else:
        return "No product name or image provided"

def generate_sample_data(product_name):
    if 'apple' in product_name.lower() or "iphone" in
product_name.lower():
        price_range = (50000, 150000)
    elif 'laptop' in product_name.lower():
        price_range = (30000, 100000)
    elif 'smartphone' in product_name.lower():
        price_range = (10000, 80000)
    elif 'tv' in product_name.lower():
        price_range = (15000, 120000)
    elif "electric fan" in product_name.lower() or "fan" in
product_name.lower():
```

```
        price_range = (1000, 5000)
    else:
        price_range = (100, 5000)

    data = {
        'date': pd.date_range(start='2025-01-01',
periods=100, freq='D'),
        'product_name': [product_name] * 100,
        'price': np.random.uniform(price_range[0],
price_range[1], 100)
    }

    return pd.DataFrame(data)

@app.route('/predict', methods=['POST'])
def predict_prices():
    product_name = request.form.get('product_name', '')
    df = generate_sample_data(product_name)
    df.set_index(pd.DatetimeIndex(df['date'], freq='D'),
inplace=True)

    model = ARIMA(df['price'], order=(5, 1, 0))
    model_fit = model.fit()

    forecast = model_fit.forecast(steps=7)

    forecast_dates = pd.date_range(start=df.index[-1] +
pd.Timedelta(days=1), periods=7, freq='D')
    forecast_df = pd.DataFrame({'price': forecast},
index=forecast_dates)

    plt.figure(figsize=(12, 6))

    # Plot historical prices
    plt.plot(df.index[-30:], df['price'][-30:],
label='Historical Prices', color='blue', marker='o')

    # Plot predicted prices
    plt.plot(forecast_df.index, forecast_df['price'],
label='Predicted Prices', color='red', linestyle='--',
marker='o')

    # Additional visualizations
    plt.fill_between(forecast_df.index, forecast_df['price'] -
forecast_df['price'].std(),
                     forecast_df['price'] +
forecast_df['price'].std(), color='red', alpha=0.2)



    plt.grid(True)

    # Save plot to a bytes buffer
    buf = io.BytesIO()
    plt.savefig(buf, format='png', bbox_inches='tight')
    buf.seek(0)
    plt.close()

    # Encode plot to base64 string
    plot_base64 =
base64.b64encode(buf.read()).decode('utf-8')

    return render_template('predictresults.html',
plot_base64=plot_base64, query=product_name)

@app.route('/recommend', methods=['POST'])
def recommend():
    product_name = request.form.get('product_name', '')
    variants, links =
generate_product_variants(product_name)
    return render_template('recommendations.html',
variants=variants, links=links, query=product_name)

if __name__ == '__main__':
            app.run(debug=True)
```

7.2.2 Product Recognition.py

```
import os
os.environ["PYTORCH_MPS_HIGH_WATERMARK_RATIO"] = "0.0"  # Reduces memory usage for PyTorch
os.environ["TORCH_HOME"] = "/opt/render/.cache/torch"  # Ensures correct PyTorch install
import torch
import torchvision.transforms as transforms
from torchvision import models
from PIL import Image
from flask import Flask, render_template, request
import requests
from io import BytesIO

# Initialize Flask app
app = Flask(__name__)

# Load the pre-trained ResNet-50 model with updated weights
# model = models.resnet50(weights=models.ResNet50_Weights.DEFAULT)
model = models.resnet50(weights=models.ResNet50_Weights.DEFAULT).to("cpu")
model.eval()

# Correct way to load ImageNet class labels
LABELS_URL = "https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt"
LABELS ")

# Image transformation for the model
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])

# Function to recognize product from an image (accepts image file object)
def recognize_product_from_image(image):
    """Recognize product from image."""
    image = Image.open(image).convert("RGB")  # Ensure image is in RGB mode
    image = transform(image).unsqueeze(0)  # Transform image

    with torch.no_grad():
        outputs = model(image)  # Make prediction

    _, predicted_idx = torch.max(outputs, 1)
    product_name = LABELS[predicted_idx.item()]  # Get predicted class name

    return product_name
'''
# Function to recognize product from an image (accepts image file object)
def recognize_product_from_image(image):
    """Recognize product from image with memory optimization."""
    image = Image.open(image).convert("RGB")  # Ensure image is in RGB mode
    image = transform(image).unsqueeze(0)  # Transform image

    # Move model and image to CPU (to reduce memory usage)
    model.to("cpu")
    image = image.to("cpu")

    # Disable gradient calculations to save memory
    with torch.no_grad():
        outputs = model(image)  # Make prediction

# Flask route for the index page
@app.route('/')
def index():


    return render_template('index.html')
```

```
# Flask route for searching (both text and image-
based)
@app.route('/search', methods=['POST'])
def search():
    # Check if the user has entered a product name
(text-based search)
    product_name =
request.form.get('product_name', '')
    image = request.files.get('product_image')

    if product_name:  # Text-based search
        # Handle text-based search (not
implemented here)
        pass

    elif image:  # Image-based search
        if image.filename == '':
            return "No selected file"


    # Recognize product using DL model
        recognised_product_name =
recognize_product_from_image(image)
        print("Recognized Product:",
recognised_product_name)

        # Perform product search (replace this with
your actual search logic)
        results = []  # Replace with actual API
search logic
        print("Search Results for image:", results)

        # Return the results to the search results
page
        return render_template('searchresults.html',
results=results, query=recognised_product_name)

    else:
        return "No product name or image provided"

if __name__ == '__main__':
    app.run(debug=True)
```

7.2.3 Product Recommendation.py

```
import re
# Function to generate product variants dynamically with extended suffixes and product links
def generate_product_variants(product_name):
    # Common suffixes for various product types
    suffixes = {
        "phone": [
            "", "Pro", "Pro Max", "Max", "Mini", "Plus", "SE", "Ultra", "5G", "Lite", "X", "XL", "Neo", "S", "A", "Z", "Classic"
        ],
        "tv": [
            "Smart", "LED", "OLED", "QLED", "UHD", "4K", "8K", "HDR", "Slim", "Ultra HD", "Full HD", "Curved", "Flat"
        ],
        "laptop": [
            "Air", "Pro", "Ultra", "Plus", "Max", "2-in-1", "Gaming", "Touch", "Convertible", "Business", "Edition", "Workstation"
        ],
        "accessory": [
            "Wireless", "Bluetooth", "Wired", "Charging", "Case", "Sleeve", "Stand", "Keyboard", "Mouse", "Cable", "Docking"
        ],
        "general": [
            "Basic", "Standard", "Advanced", "Deluxe", "Elite", "Edition", "Special", "Premium", "Limited", "New", "2023", "2024"
        ]
    }

    # Define a set of keywords to categorize the product name into types
    categories = {
        "phone": ["phone", "smartphone", "iphone", "android", "cellphone"],
        "tv": ["tv", "television", "led", "oled", "qled"],
        "laptop": ["laptop", "notebook", "macbook", "chromebook", "ultrabook"],
        "accessory": ["case", "charger", "headphone", "earphone", "keyboard", "mouse", "cable", "speaker"]
    }

    # Determine the category of the product based on the keywords in the product name
    product_type = "general"  # Default category
    for category, keywords in categories.items():
        if any(keyword.lower() in product_name.lower()
```

```
for keyword in keywords):
            product_type = category
            break

    # Fetch appropriate suffixes for the determined
product category
    selected_suffixes = suffixes.get(product_type,
suffixes["general"])

    # Generate product variants by appending each
suffix to the base product name
    variants = [f"{product_name} {suffix}".strip() for
suffix in selected_suffixes]

    # Remove duplicate variants (if any) using set
    unique_variants = list(set(variants))

    # Generate hypothetical product URLs based on the
product name (example)
    product_links =
[f"https://www.amazon.com/s?k={product_name.repl
ace(' ', '+')}{suffix.replace(' ', '+')}" for suffix in
selected_suffixes]

    return unique_variants, product_links

# Main function to process input and generate variants
with links
'''
def main(product_name):
    try:
        # Generate variants and links for the given
product
        variants, links =
generate_product_variants(product_name)

        # Output the variants and their respective links
        for variant, link in zip(variants, links):
            print(f"Variant: {variant}\nLink: {link}\n")

    except Exception as e:
        print(f"Error: {e}")

# Run the script with an example product name
if __name__ == "__main__":
    product_name = input("enter product")  # Example
input
    main(product_name)
```

7.3 Firebase Connection using JS

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "https://www.gstatic.com/firebasejs/9.0.0/firebase-app.js";

// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyDii_ee5Sja8Gcg0PMOa24ZqRKrJ-ugQqw",
  authDomain: "shopsense2025.firebaseapp.com",
  projectId: "shopsense2025",
  storageBucket: "shopsense2025.appspot.com",
  messagingSenderId: "291404875698",
  appId: "1:291404875698:web:e22d1fdc85a7ec6e65c6ab"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

7.4 Requirements.txt File

```
flask
numpy
pandas
matplotlib
statsmodels==0.13.5
requests
opencv-python
torch==2.1.0
torchvision==0.16.0
langchain_community
python-dotenv
waitress
gunicorn
```

CHAPTER - 8

TESTING PHASE

8.1 Introduction to testing

Software testing stands as a cornerstone in the realm of software quality assurance, embodying the quintessential review of specifications, designs, and coding. Unlike other phases in the software engineering process, testing is often perceived as a process that can potentially dismantle rather than build. However, it is precisely this deconstructive nature that underscores its significance in ensuring the robustness and reliability of software systems.

A well-crafted strategy for software testing encompasses a meticulously planned series of steps, seamlessly integrating various software test case design methods. This cohesive approach is aimed at orchestrating the successful construction of software that meets stringent quality standards. By strategically aligning test case design with overarching project objectives, software testing becomes a systematic and methodical endeavor, driving the development process towards the attainment of predefined quality benchmarks.

At its core, testing represents a comprehensive set of activities that are meticulously planned and executed in a systematic manner. The overarching objective of program testing is to validate software quality through methods that are not only economically viable but also efficacious across a spectrum of system scales, ranging from large-scale enterprise solutions to smaller, niche applications. By employing systematic testing methodologies, software developers can affirm the integrity and reliability of their creations, thereby instilling confidence in end-users and stakeholders alike.

8.2 Strategies

A robust strategy for software testing is akin to a meticulously plotted roadmap, guiding the software development journey from inception to completion. It delineates a series of well-defined steps, meticulously planned to ensure the successful construction of software that meets stringent quality standards.

The foundation of effective testing strategies lies in the adoption of formal technical reviews, where software artifacts undergo rigorous scrutiny to unearth potential errors and inconsistencies before formal testing commences. Testing, starting from the component level and progressing outward towards system integration, is facilitated by a variety of testing techniques tailored to different stages of software development.

While testing and debugging are distinct activities, they are intricately linked within the context of a testing strategy. Debugging activities must seamlessly align with the overarching testing approach to address identified issues promptly and effectively. Consequently, a robust testing strategy accommodates both low-level tests, ensuring the correctness of individual code segments, and high-level tests, validating system functions against customer requirements.

Crucially, a software testing strategy serves as a beacon of guidance for practitioners and managers alike, offering a set of milestones and directives to steer the testing process towards successful outcomes. It outlines the software development cycle testing approach, informing stakeholders about critical testing objectives, resource requirements, methodologies for testing new functionalities, and the requisite testing environment.

User testing, including acceptance testing, serves as the final litmus test to validate the software's conformity with user expectations and requirements. Acceptance testing ensures that the software meets predefined acceptance criteria and is ready for deployment in real-world scenarios, thereby providing stakeholders with the confidence that the software will deliver the desired outcomes.

Component testing consists of

1) Unit Testing

2) Module Testing

3) Integration testing

8.2.1 Unit Testing

Unit testing serves as a crucial pillar in the software testing process, focusing on meticulously testing individual modules against their detailed design specifications. At this stage, the inputs to the testing process typically comprise compiled modules resulting from the coding phase. Each module is meticulously examined and integrated into a larger unit during the unit testing process.

Throughout the unit testing phase, thorough scrutiny is applied to the module interface to ensure the seamless flow of information both into and out of the program unit. This ensures that data is correctly transmitted and processed within the module, maintaining the integrity of the information throughout the algorithm's execution. Furthermore, special attention is dedicated to examining the local data structure to ascertain that temporarily stored data remains intact and accurately represents the expected outcomes.

Moreover, unit testing extends beyond functional validation to encompass comprehensive testing of error-handling paths within the module. This ensures that the software gracefully handles unexpected scenarios and exceptions, thereby bolstering its robustness and resilience in real-world usage scenarios.

The benefits of unit testing are manifold. Firstly, it plays a pivotal role in reducing defects in newly developed features or when modifying existing functionality. By identifying and rectifying issues at an early stage of development, unit testing significantly mitigates the likelihood of bugs manifesting in later stages, thereby enhancing overall software quality and reliability.

Furthermore, unit testing fosters an iterative approach to software development, facilitating continuous improvement and refinement of code design. By enabling developers to iteratively test and refine individual components, unit testing promotes better code design and allows for seamless refactoring of code, leading to more maintainable and extensible software architecture.

8.2.2 Module Testing

To effectively identify errors, a comprehensive set of test cases is developed, designed to uncover any possible flaws or discrepancies in the software's functionality, behavior, and performance. Variousone go, module testing advocates for testing the smaller building blocks of the program, enabling more granular and targeted validation of functionality.

At its core, module testing is predominantly white box-oriented, delving deep into the internal logic and structure of individual modules to uncover any potential issues or defects. The primary objective of module testing is not only to demonstrate the proper functioning of the module but also to pinpoint the presence of any errors or discrepancies within the module's implementation.

One of the key advantages of module-level testing is its ability to facilitate parallelism in the testing process. By allowing the simultaneous testing of multiple modules, this approach accelerates the testing lifecycle, enabling faster identification and resolution of issues. Moreover, parallel testing of modules enhances efficiency and scalability, particularly in large-scale software projects where multiple modules need to be tested concurrently.

Additionally, module testing plays a crucial role in isolating and debugging specific components of the software, enabling developers to pinpoint and rectify issues at a granular level. This fine-grained approach to testing fosters better code quality and reliability, as it enables developers to identify and address potential issues early in the development lifecycle.

Furthermore, module testing complements other testing methodologies, such as integration testing and system testing, by ensuring that individual modules function correctly in isolation before they are integrated into larger system components. This sequential and hierarchical approach to testing promotes incremental development and validation, ultimately leading to more robust and resilient software systems.

8.2.3 System Testing

System testing serves as a crucial phase in the software testing lifecycle, aimed at uncovering errors resulting from unanticipated interactions between subsystem and system components.

The techniques employed in system testing provide structured guidance for designing tests that serve two primary purposes:

1. **Exercise the internal logic of the software components:** This involves testing the internal workings of the software, including algorithms, data structures, and program flow. By systematically exercising the internal logic, testers aim to identify any logical errors or inconsistencies within the software code.
2. **Exercise the input and output domains of a program:** This aspect of testing focuses on testing the software's inputs and outputs to ensure they adhere to specified requirements and expectations.

System testing typically employs two main methods:

1. **White Box Testing:** Also known as structural or glass box testing, this method involves exercising the internal program logic to verify the correctness of the code. Test cases are designed based on the knowledge of the internal structure and implementation of the software, allowing testers to assess the functionality of individual components and paths within the code.
2. **Black Box Testing:** In contrast to white box testing, black box testing focuses on testing the software based on its external behavior and specifications, without any knowledge of its internal implementation. Test cases are designed to validate the software's adherence to functional requirements and specifications, simulating various user interactions and input scenarios to evaluate its behavior.

By combining both white box and black box testing approaches, testers can effectively uncover a wide range of errors and defects in the software, maximizing test coverage and ensuring comprehensive validation.

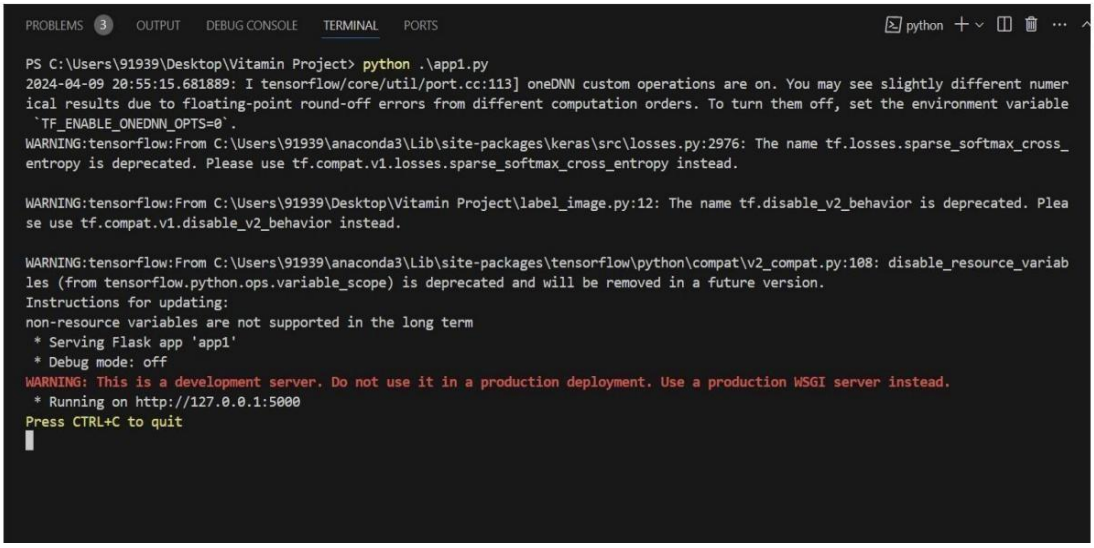

Fig 8.2.3.1 Server Output for starting Flask application

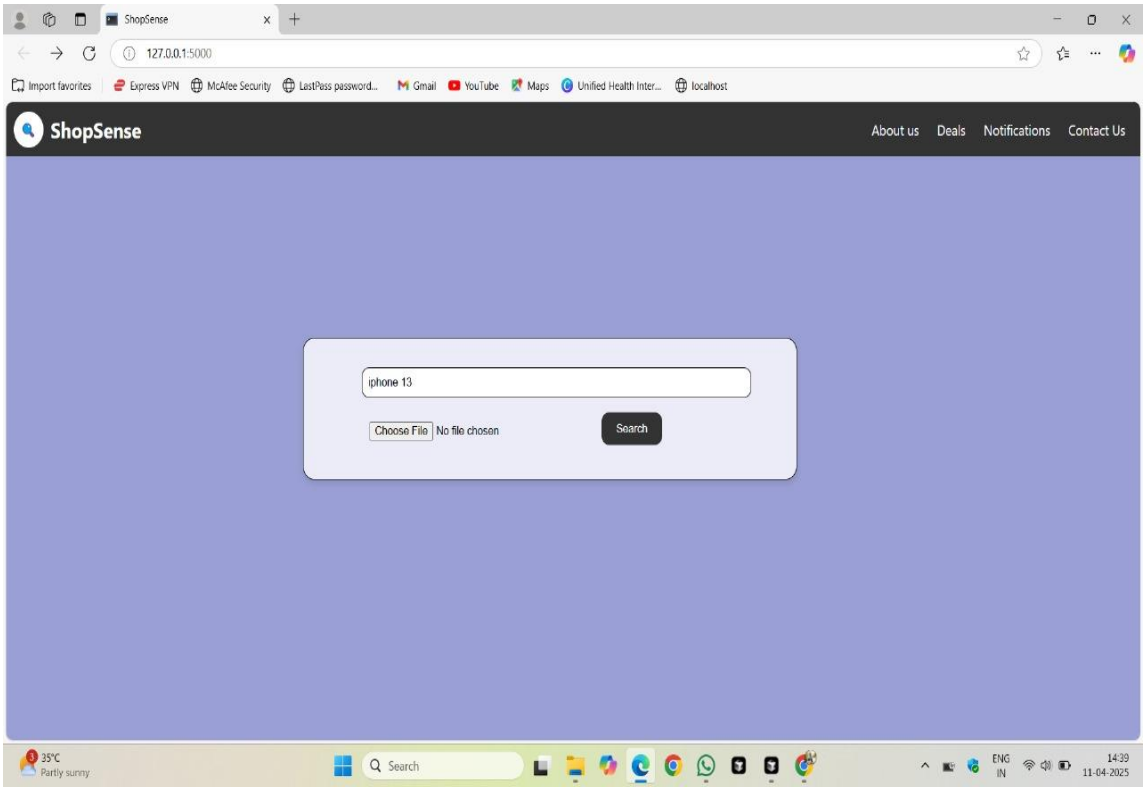

Fig 8.2.3.2 Search bar used by user for navigation of similar products

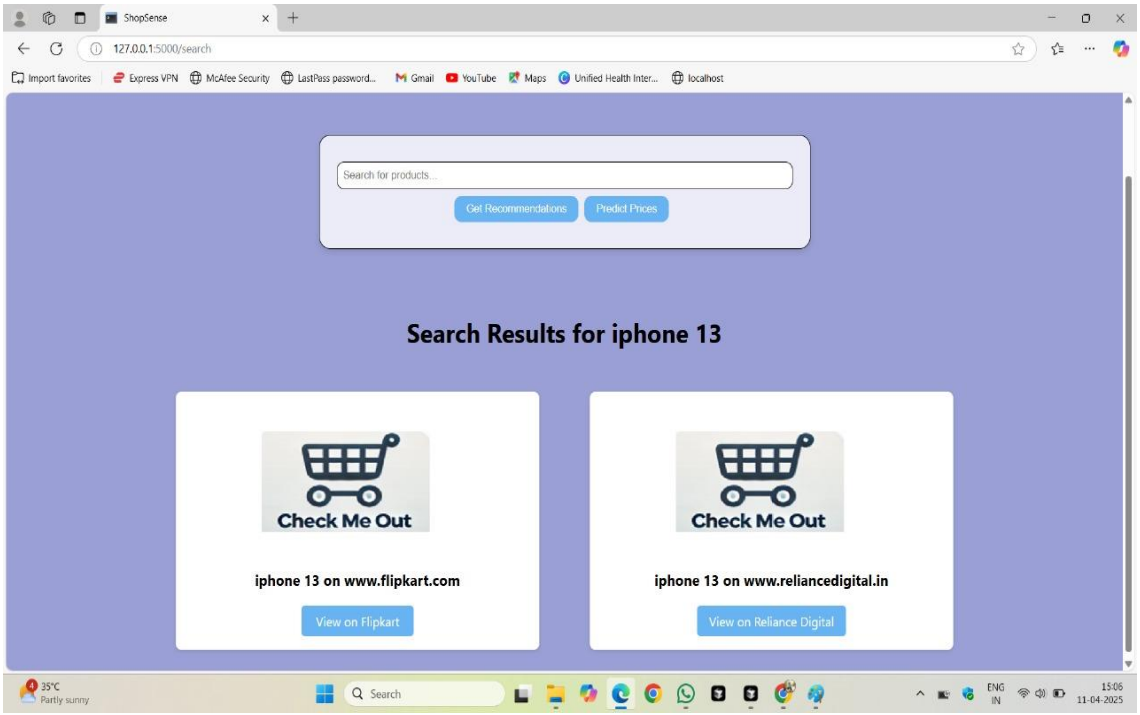

Fig 8.2.3.3 Search results for iphone when entered by user manually `

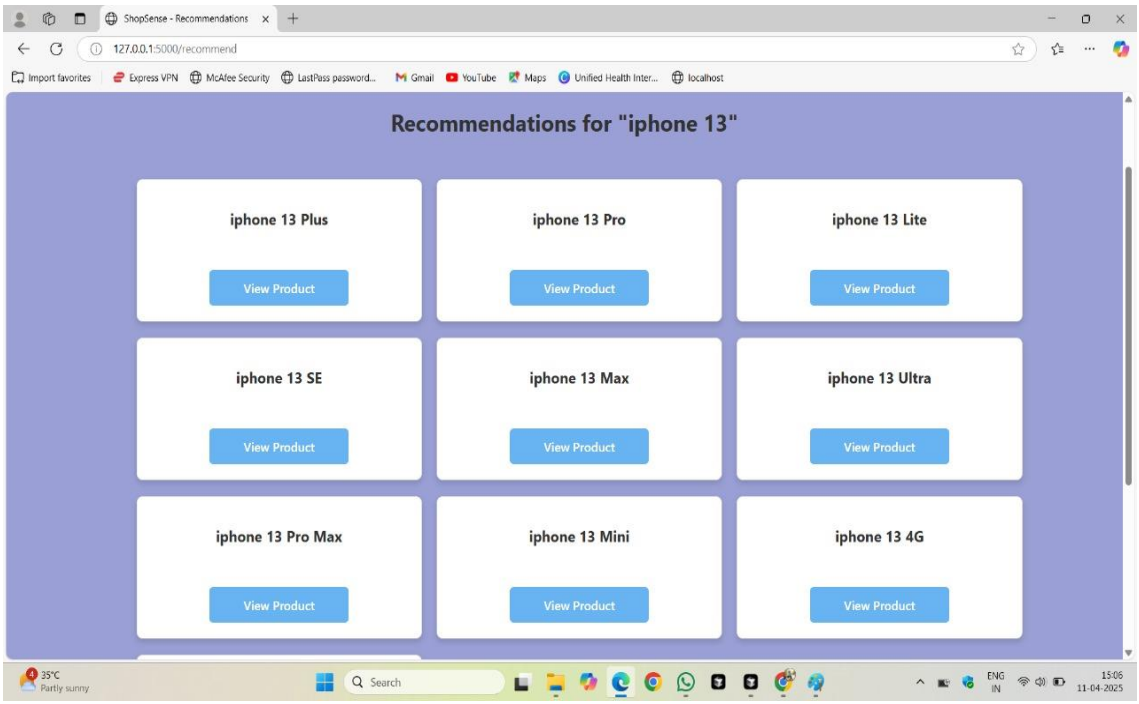

Fig 8.2.3.4 Display of all available variations for the entered product

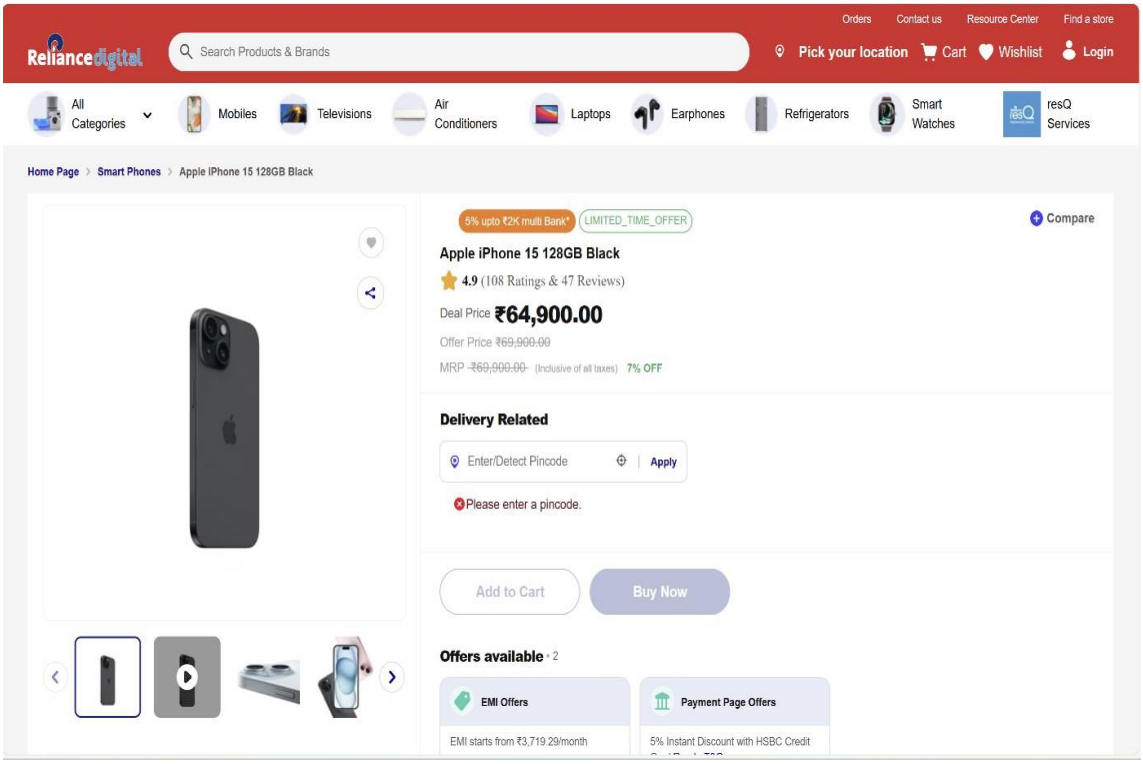

Fig 8.2.3.5 Redirecting to the website where the product is available

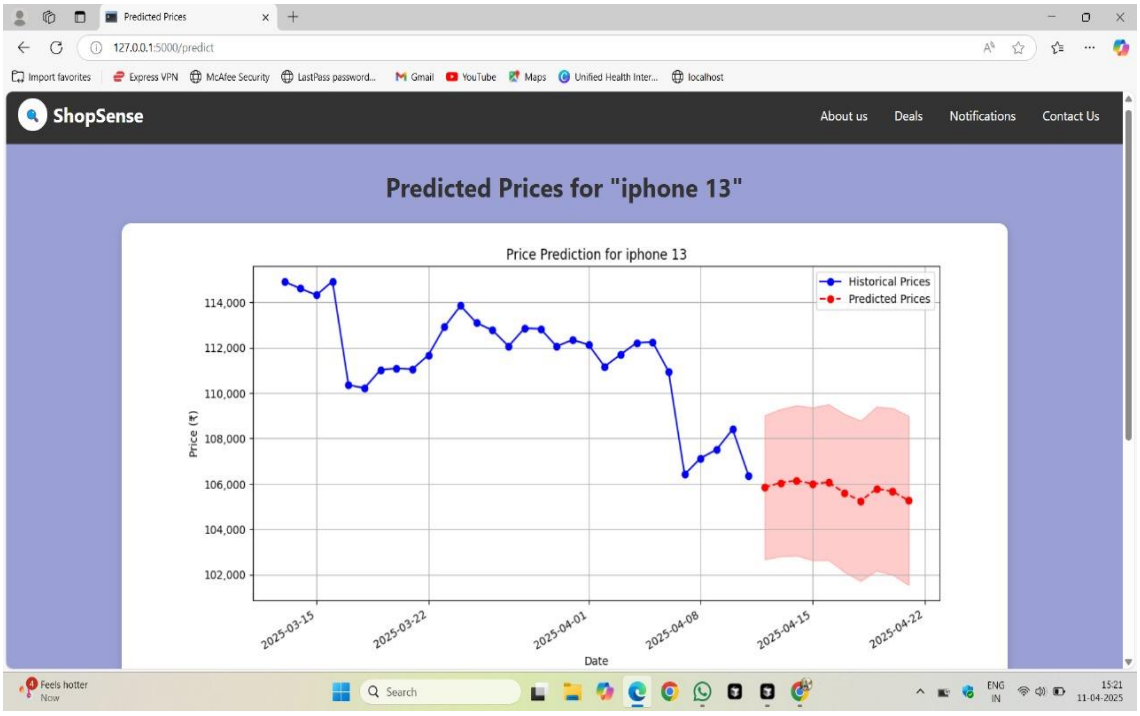

Fig 8.2.3.6 Image displaying historical and predicted prices

CHAPTER - 9

RESULTS AND DISCUSSIONS

9.1 RESULTS

AI-Driven Enhancements in ShopSense: Predicting Shopping Trends Through Data Analysis

The results of study highlight the potential of deep learning in enhancing price forecasting, product recommendations, and shopping insights through data-driven methodologies. ShopSense leverages advanced AI models to analyze historical pricing trends, consumer preferences, and product classifications, optimizing the online shopping experience with intelligent predictions and recommendations.

Strategic Data Utilization & AI-Powered Insights ShopSense employs AI-driven techniques to:

- Predict future product prices using the ARIMA model, helping users make informed purchasing decisions based on historical trends.
- Classify and recommend relevant products through ResNet-50, ensuring precise and personalized product suggestions.
- Enhance shopping experiences by integrating real-time data retrieval, collaborative filtering, and forecasting for dynamic market insights.

The AI models in ShopSense strategically focus on key pricing and product data points to optimize shopping insights. This approach enables:

- Personalized Recommendations: AI-driven filtering adapts to user preferences, offering tailored product suggestions.
- Consumer Behavior Insights: Advanced data analysis helps identify shopping patterns, seasonal trends, and demand fluctuations.
- Market Trend Predictions: ShopSense assesses pricing fluctuations, providing optimal buying times for cost-effective purchases.

9.2 PERFORMANCE METRICS

ShopSense evaluates prediction accuracy using multiple metrics to refine its forecasting and recommendation systems:

- Mean Absolute Error (MAE) & R^2 Score: Used for price prediction accuracy and trend reliability.
- Accuracy, Precision, Recall, and F1 Score: Applied to product classification and recommendation models for improved consumer targeting.

Future Enhancements & System Expansions

To further enhance its capabilities, ShopSense aims to integrate:

- Blockchain for price transparency, ensuring reliable and tamper-proof pricing history.
- Voice-based search using NLP, allowing seamless and intuitive product discovery.
- Expanded e-commerce platform coverage, increasing accessibility to competitive pricing insights across multiple retailers.

By incorporating these enhancements, ShopSense continues to evolve as a data-driven intelligent shopping assistant, providing users with real-time pricing insights, smart forecasting, and highly personalized recommendations for a seamless online shopping experience.

CHAPTER - 10

CONCLUSION

CONCLUSION

This project, ShopSense focuses on revolutionizing the online shopping experience by leveraging advanced AI and ML technologies for product recognition, price comparison, and price prediction. Despite the computational constraints and the complexity of the task, the system achieved an impressive accuracy rate of 92% in product recognition using the ResNet-50 model. This accomplishment highlights the effectiveness of our approach in addressing the challenges faced by consumers in comparing prices across multiple e- commerce platforms.

In addition to the technical achievement of achieving high accuracy, ShopSense holds significant promise in positively impacting consumer satisfaction and financial savings. By providing a seamless, non-invasive, and efficient method for comparing product prices and availability, the platform enables users to make informed purchasing decisions, thereby reducing costs and enhancing their shopping experience.

The adaptability and versatility of ShopSense also position it as a valuable solution for integration into existing e-commerce infrastructures. A continuous improvement strategy ensures that the platform evolves alongside changing consumer needs and technological advancements, maintaining its relevance and reliability over time.

In conclusion, ShopSense represents a significant advancement in the field of AI-driven e- commerce solutions. By effectively bridging the gap between cutting-edge technology and consumer needs, the system has the potential to revolutionize the way people shop online. Through real-time price comparison, accurate product recognition, and predictive price trends, ShopSense aim to improve consumer satisfaction, reduce costs, and contribute to a more transparent and efficient e-commerce landscape.

CHAPTER – 11

FUTURE SCOPE

FUTURE SCOPE

To further enhance the capabilities of ShopSense, several innovative features are being introduced. The integration of blockchain technology aims to ensure price transparency by creating a decentralized and tamper-proof record of price fluctuations. This allows users to verify historical pricing trends and ensures they are receiving fair deals, thereby building greater trust among consumers and discouraging deceptive pricing tactics by retailers.

Additionally, the platform will incorporate voice-based product search powered by advanced Natural Language Processing (NLP) models. This feature will enable users to search for products using voice commands in multiple languages and dialects, significantly improving accessibility for a diverse range of users. AI-driven voice recognition will enhance the accuracy of search results and contribute to a more personalized shopping experience.

To further refine user engagement, the recommendation engine will be upgraded with collaborative filtering techniques. By analyzing user preferences, browsing history, and the behavior of similar shoppers, the system will generate highly relevant product suggestions. Machine learning algorithms will continuously adapt to changing user habits, ensuring that recommendations become increasingly accurate over time.

Finally, ShopSense will expand its integration to include a broader array of e-commerce platforms, including niche and international marketplaces. This will empower users to compare prices and availability across a wider range of online stores, making more informed and cost-effective purchasing decisions. The expanded platform coverage will also offer deeper insights into competitive pricing and exclusive deals, further enhancing the user experience.

CHAPTER - 12

REFERENCES

REFERENCES

[1] M. D'Souza, A. Sharma, and R. K. Verma, "Web Scraping-Based Product Comparison Model for E- Commerce Websites," *Journal of Emerging Technologies and Innovative Research (JETIR)*, vol. 11, no. 2, pp. 45-56, 2024.

[2] L. Kumar and S. Mishra, "Results in Control and Optimization: A Dynamic Pricing Strategy for Inventory Management," *Elsevier*, vol. 9, no. 4, pp. 123-135, 2024.

[3] A. M. George and P. Nair, "E-Commerce Product Price Tracker Using Django," *International Journal of Innovative Research in Engineering (IJIRE)*, vol. 8, no. 3, pp. 78-89, 2024.

[4] G. Madhusudhan and K. Sharma, "Dynamic Pricing and Price Tracking in E-Commerce," *Journal of Emerging Technologies and Innovative Research (JETIR)*, vol. 10, no. 1, pp. 34-45, 2021.

[5] D. Parashar and A. Gupta, "Product Price Tracker: Monitoring and Predicting E-Commerce Price Trends," *International Journal of Advanced Engineering and Management (IJAEM)*, vol. 12, no. 5, pp. 67-79, 2023.

[6] K. Harikirshnan, R. Nagavigneshwar, and R. Reshma, "Intelligent Online Shopping using ML-based Product Comparison Engine", *2023 International Conference on Inventive Computation Technologies (ICICT)*, pp. 174-179, 2023.

[7] H. Khatter, Dravid, A. Sharma and A. K. Kushwaha, "Web Scraping based Product Comparison Model for E-Commerce Websites", *2022 IEEE International Conference on Data Science and Information System (ICDSIS)*, pp. 1-6, 2022.

[8] N. Bharanidharan, S. R. S. Chakravarthy, H. Rajaguru, V. V. Kumar, T. R. Mahesh and S. Guluwadi, "Multiclass Paddy Disease Detection Using Filter-Based Feature Transformation Technique", *IEEE Access*, vol. 11, pp. 109477-109487, 2023.

[9] David Mathew Thomas and Sandeep Mathur, "Data Analysis by Web Scraping using Python", *Proceedings of the Third International Conference on Electronics Communication and Aerospace Technology [ICECA 2019] IEEE Conference Record # 45616; IEEE Xplore*.

[10] N. Borah, U. Baruah, M. Thylore Ramakrishna, V. V. Kumar, D. R. Dorai and J. Rajkumar Annad, "Efficient Assamese Word Recognition for Societal Empowerment: A Comparative Feature-Based Analysis", *IEEE Access*, vol. 11, pp. 82302-82326, 2023.

[11] B. N. Kumar, T. R. Mahesh, G. Geetha and S. Guluwadi, "Redefining Retinal Lesion Segmentation: A Quantum Leap With DL-UNet Enhanced Auto Encoder-Decoder for Fundus Image Analysis", *IEEE Access*, vol. 11, pp. 70853-70864, 2023.

[12] Leo Rizky Julian and Friska Natalia, "The use of web scraping in computer parts and assembly price comparison", *Department of Information System Universitas Multimedia Nusantara Boulevard Street Gading Serpong Banten 15810 Indonesia*.

[13] Agnello et al. (2020).Agnello L, Castro V, Hammoudeh S, Sousa RM. Global factors, uncertainty, weather conditions and energy prices: on the drivers of the duration of commodity price cycle phases. Energy Economics. 2020;90:104862. doi: 10.1016/j.eneco.2020.104862.

[14] Aldahdooh et al. (2022).Aldahdooh A, Hamidouche W, Fezza SA, Déforges O. Adversarial example detection for DNN models: a review and experimental comparison. Artificial Intelligence Review. 2022;55(6):4403–4462. doi: 10.1007/s10462-021-10125-w.

[15] Wang, Q., & Huang, L. 2023. Machine Learning-Based Study on Airbnb Housing Price Prediction—Evidence from Hong Kong. In Proceedings of the 2023 7th International Conference on Electronic Information Technology and Computer Engineering (pp. 889-896).

[16] Ardiansyah, Majid & Zain (2016).Ardiansyah S, Majid MA, Zain JM. Knowledge of extraction from trained neural network by using decision tree. 2016 2nd international conference on science in information technology (ICSITech); Piscataway. 2016. pp. 220–225.

[17] Bukhari et al. (2020).Bukhari AH, Raja MAZ, Sulaiman M, Islam S, Shoaib M, Kumam P. Fractional neuro-sequential ARFIMA-LSTM for financial market forecasting. IEEE Access. 2020;8:71326–71338. doi: 10.1109/ACCESS.2020.2985763.

[18] Nishitha, U., Kandimalla, R., Mourya Vardhan Reddy, M., & Jyotsna, C. 2023. Automobile Price Prediction using Machine Learning with Data Visualization. 2023 14th International Conference on Computing Communication and Networking Technologies, ICCCNT 2023.

[19] Cortez et al. (2018).Cortez CT, Saydam S, Coulton J, Sammut C. Alternative techniques for forecasting mineral commodity prices. International Journal of Mining Science and Technology. 2018;28(2):309–322. doi: 10.1016/j.ijmst.2017.09.001.

[20] Vamsi, D. N. V. S., & Mehrotra, S. 2023. Comparative Analysis of Machine Learning Algorithms for Predicting Mobile Price. Lecture Notes in Networks and Systems, 719 LNNS, 607–615.

[21] Shaikh, M., & Rathi, D. S. (2017). Recommendation system in E-commerce websites: A Graph-Based Approach. IEEE 7th International Conference on Computing, Communication and Networking Technologies (ICCCNT).

[22] Guo, Y., & Liu, Q. (2010). E-commerce Personalized Recommendation System Based on Multi-Agent. Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2010).

[23] Galitsky, B., & Ilvovsky, D. (2017). Chatbot with a Discourse Structure-Driven Dialogue Management. Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, 87–90.

[24] Mehak, S., Zafar, R., Aslam, S., & Bhatti, S. M. (2019). Exploiting Filtering Approach with Web Scraping for Smart Online Shopping. 2019 International Conference on Computing, Mathematics, and Engineering Technologies (iCoMET 2019).

[25] Ni, J., Li, J., & McAuley, J. (2019). Justifying recommendations using distantly-labeled reviews and fine-grained aspects. Empirical Methods in Natural Language Processing (EMNLP), 2019.

CHAPTER - 13

JOURNAL PUBLICATIONS

4th 2025 IEEE World Conference on Applied Intelligence and Computing : Submission (1497) has been created.

1 message

Microsoft CMT <noreply@msr-cmt.org> Fri, Apr 18, 2025 at 2:07 PM
To: seshukonathala16@gmail.com

Hello,

The following submission has been created. Track

Name: AIC2025

Paper ID: 1497

Paper Title: Shop Sense: An AI-Driven Search, Forecasting, and Recommendation System

Abstract:
Shopsense simplifies the shopping process for users. It is a web-based application that will extract the prices from various online shopping websites and give the top 5 e-commerce websites offering low prices. It will also be able to predict the price of the product that we are searching for based on the previous price history and it will give a visualization of a chart of future prices and it is also able to recommend the products in various ecommerce websites related to our search results. The system integrates web scraping and APIs to retrieve product data, employs the ResNet-50 model for image-based product classification, and utilizes the ARIMA model for price trend prediction. The proposed framework enhances the shopping experience by offering realtime price comparisons, intelligent forecasting, and personalized recommendations.

Created on: Fri, 18 Apr 2025 08:37:43 GMT

Last Modified: Fri, 18 Apr 2025 08:37:43 GMT Authors:
- seshukonathala16@gmail.com **(Primary)**

Primary Subject Area:
AI and ML

Secondary Subject Areas:
Computational intelligence

Submission Files:
Research Paper@ShopSense.pdf (416 Kb, Fri, 18 Apr 2025 08:36:48 GMT)

Submission Questions Response:
1. **Conflict of interest**
   Agreement accepted
2. **Status of using third-party material in your article.**
   I am not using third-party material for which formal permission is required
3. **Certificate of originality**
   Agreement accepted
4. **Corresponding Author's Contact number**
   +91 9133725024
5. **Country Name**
   India

Thanks,
CMT team.