

**A PROJECT REPORT  
ON  
FAKE REVIEW DETECTION  
&  
PRODUCT RECOMMENDATION  
DIPLOMA  
IN  
COMPUTER ENGINEERING  
BY**

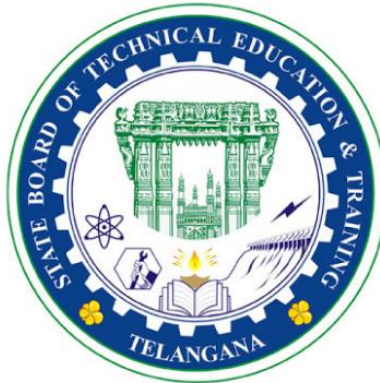
**S. PRAVEEN  
P. SATHWIK  
G. ABHIRAM  
K. HIMAJA  
A. VARSHINI**

**21047-CS-012  
21047-CS-006  
21047-CS-020  
21047-CS-059  
21047-CS-009**

Under the Guidance of

**G. SREE MADHURI, M. Tech.,**

LECTURER IN COMPUTER ENGINEERING



**SANJAY GANDHI MEMORIAL GOVERNMENT POLYTECHNIC**  
Near Ramoji Film City, Abdullapurmet, R. R. Dist.- 501505  
**(Accredited by National Board of Accreditation)**  
**Department of Computer Engineering**



## CERTIFICATE

This is to certify that the project work entitled “**FAKE REVIEW DETECTION & PRODUCT RECOMMENDATION**” carried out by **S. PRAVEEN, G. ABHIRAM, P. SATHWIK, K. HIMAJA, A. VARSHINI** bearing Roll No. **21047-CS-012, 21047-CS-020, 21047-CS-006, 21047-CS-006, 21047-CS-059, 21047-CS-009** in partial fulfilment of the requirements for the award of the degree of Diploma in Computer Engineering is a record of bonafide work carried out by him/her under my guidance.

The results of investigation enclosed in this report have been verified and found satisfactory. The results embodied in this project report have been submitted to any other University or Institute for the award of any other degree of diploma.

Signature of the Guide  
**G. SREE MADHURI, M. Tech.,**

Head of the Department  
**B. VAJRAIAH, M. Tech.,**

External

Principal  
**CH. V. KRISHNA RAO**  
M.E., MBA, DIS, FIE FIE(PH. D)

## DECLARATION BY CANDIDATE

We **S. PRAVEEN, G. ABHIRAM, P. SATHWIK, K. HIMAJA, A. VARSHINI** bearing Roll No. **21047-CS-012, 21047-CS-020, 21047-CS-006, 21047-CS-006, 21047-CS-059, 21047-CS-009** in **“FAKE REVIEW DETECTION & PRODUCT RECOMMENDATION”** is done under the guidance of **Mrs G. SREE MADHURI**, Department of Computer Engineering, S.G.M. Government Polytechnic, is submitted in partial fulfilment of the requirements for the award of the Diploma in Computer Engineering.

This is a record of bonafide work carried out by me in S.G.M. GOVERNMENT POLYTECHNIC and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

**BY**

<b>S. PRAVEEN</b>	<b>21047-CS-012</b>
<b>P. SATHWIK</b>	<b>21047-CS-006</b>
<b>G. ABHIRAM</b>	<b>21047-CS-020</b>
<b>K. HIMAJA</b>	<b>21047-CS-059</b>
<b>A. VARSHINI</b>	<b>21047-CS-009</b>

## TABLE OF CONTENTS

<b>Section</b>	<b>Page no.</b>
<b>ABSTRACT</b>	<b>1</b>
<b>Chapter 1 - INTRODUCTION</b>	<b>2</b>
<b>1.1.</b> Features of the project	<b>3-3</b>
<b>1.2.</b> Applications of the project	<b>4-4</b>
<b>1.3.</b> Hardware Requirements	<b>5-5</b>
<b>1.4.</b> Software Requirements	<b>6-6</b>
<b>1.5.</b> Existing System	<b>7-7</b>
<b>1.6.</b> Proposed System	<b>8-8</b>
<b>Chapter 2 - LITERATURE REVIEW</b>	<b>9</b>
<b>Chapter 3-DESIGN AND METHODOLOGIES</b>	<b>10</b>
<b>3.1.</b> Python	<b>10-10</b>
<b>3.2.</b> Machine Learning	<b>11-11</b>
<b>3.2.1.</b> Types of Machine Learning	<b>11-11</b>
<b>3.3.</b> Installed Libraries	<b>12-13</b>
<b>3.3.1.</b> Requirements.txt	<b>14-14</b>
<b>Chapter 4 - IMPLEMENTATION</b>	<b>15</b>
<b>4.1.</b> TextBlob	<b>15-15</b>

<b>4.1.1.</b>	Parts of Speech Tagging	<b>15-16</b>
<b>4.1.2.</b>	Fake review detection flowchart	<b>17-17</b>
<b>4.2.</b>	Collaborative filtering	<b>18-18</b>
<b>4.2.1.</b>	Memory based CF	<b>19-19</b>
<b>4.2.2.</b>	User based CF	<b>19-20</b>
<b>4.2.3.</b>	Formulas Used in User-Based CF	<b>20-20</b>
<b>4.2.4.</b>	Algorithms Used in User-Based CF	<b>21-21</b>
<b>4.4.2.</b>	User based CF	<b>22-22</b>
<b>4.3.</b>	SVD Algorithm	<b>23-24</b>
<b>4.4.</b>	Database	<b>25-25</b>
<b>4.5.</b>	Source Code	<b>26-29</b>
<b>4.6.</b>	File Explanation	<b>30-31</b>
<b>Chapter 5 - RESULTS</b>		<b>32</b>
<b>5.1.</b>	Output Screens	<b>32-35</b>
<b>Chapter 6 - CONCLUSION</b>		<b>36</b>
<b>REFERENCE</b>		<b>37</b>

## **ABSTRACT**

This project focuses on combating fake reviews and enhancing product recommendations in e-commerce. Using advanced machine learning and sentiment analysis techniques, it develops a robust system to identify deceptive reviews and improve user-generated content authenticity. The fake review detection component employs linguistic pattern analysis, sentiment anomaly detection, and user behaviour analysis, achieving high accuracy in distinguishing genuine from fraudulent reviews. The framework also integrates authentic reviews into a personalized recommendation system, considering user preferences and verified feedback to enhance product suggestions. Through experimental evaluation, the system proves effective in mitigating fake reviews' impact and elevating recommendation accuracy. This research contributes significantly to creating a reliable and user-centric online shopping environment, fostering consumer trust, and promoting fair competition among businesses, ultimately enhancing the overall online shopping experience.

## **1. INTRODUCTION**

In the digital age, online reviews play a pivotal role in shaping consumer decisions. From purchasing products on e-commerce platforms to selecting hotels for a vacation, users heavily rely on the feedback and ratings provided by their peers. However, the proliferation of fake reviews poses a significant challenge to the integrity and reliability of these platforms. Inaccurate or biased reviews not only mislead consumers but also undermine the credibility of businesses and review platforms alike. To address this issue, the implementation of robust fake review detection systems has become imperative. These systems leverage advanced algorithms and machine learning techniques to analyse review patterns, linguistic cues, and user behaviours, thereby identifying suspicious reviews. By detecting and filtering out fake reviews, businesses can uphold transparency and trustworthiness, fostering a more authentic online ecosystem.

## 1.1. Features of the Project

- **Understanding Fake Review Detection:** This section provides an overview of fake review detection methodologies, including linguistic analysis, anomaly detection, and machine learning algorithms.
- **Exploring Product Recommendation Systems:** Here, we explore the principles and techniques behind product recommendation systems, including collaborative filtering, content-based filtering, and hybrid approaches.
- **Integration and Implementation Strategies:** This section offers insights into integrating fake review detection and product recommendation systems into existing platforms.
- **Future Trends and Opportunities:** Finally, we explore emerging trends and future opportunities in the realm of fake review detection and product recommendation.



## **1.2. Application of the Project to the Real World**

### **E-commerce Platforms:**

- Detect fake reviews to maintain trust.
- Recommend products based on user preferences.

### **Restaurant and Hotel Reviews:**

- Identify fake reviews for accurate information.
- Recommend similar places based on user preferences.

### **Movie and TV Show Ratings:**

- Filter out biased ratings for reliable information.
- Recommend movies/shows based on user interests.

### **Travel Booking Platforms:**

- Ensure genuine reviews for travel decisions.
- Recommend destinations based on user preferences.

### **Mobile App Stores:**

- Remove fraudulent app reviews for trustworthiness.
- Recommend apps based on user history and preferences.

### **1.3. Hardware Requirements**

- Computer/server specifications
- Storage capacity
- GPU specifications
- Intel i5 or equivalent processor
- Minimum 8GB RAM
- 500GB SSD storage
- Ethernet or Wi-Fi connectivity

## **1.4. Software Requirements**

- Programming language environment (e.g., Python)
- CSV file handling libraries
- Programming Languages
- Machine Learning Libraries
- Natural Language Processing (NLP) Libraries
- Data Storage
- Web Development Framework
- Version Control
- Development Environment

## **1.5. Existing System**

- **Fake Review Detection:**

The current system relies on manual review analysis by human moderators and basic keyword-based algorithms to identify potential fake reviews. Automation is limited, and the system heavily depends on user reports.

- **Product Recommendation:**

In terms of product recommendation, the existing system employs a basic collaborative filtering or rule-based approach with minimal personalization based on user preferences. Advanced machine learning models are not extensively utilized.

## **1.6. Proposed System**

- **Fake Review Detection:**

The proposed system aims to enhance fakes review detection through advanced Natural Language Processing (NLP) techniques. Machine learning models will be employed to analyse review sentiment, linguistic patterns, and detect anomalies. User behaviour analysis and the integration of deep learning models will contribute to more accurate fake review identification.

- **Product Recommendation:**

For product recommendation, the proposed system will introduce a more personalized approach. Recommendations will be based on user preferences, purchase history, and behaviour. Real-time updates will ensure dynamic evolution based on user interactions and evolving preferences.

## **2. LITERATURE REVIEW**

The project on fake review detection and product recommendation is crucial in today's digital landscape where online reviews greatly influence consumer decisions. With the rise of fake reviews, maintaining trust and authenticity is vital. The project employs advanced machine learning and sentiment analysis techniques to identify deceptive reviews accurately. By analyzing linguistic patterns, sentiment anomalies, and user behavior, the system effectively distinguishes genuine reviews from fraudulent ones, ensuring a more reliable online shopping experience. Additionally, the integration of authentic reviews into personalized product recommendations enhances user satisfaction and promotes fair competition among businesses. This research significantly contributes to creating a trustworthy online shopping environment, ultimately benefiting consumers and businesses alike.

## 3. DESIGN AND METHODOLOGIES

### 3.1. Python

**Python**, a versatile and high-level programming language, has become a cornerstone in the realm of software development, data science, and automation. Created by **Guido van Rossum** in the late 1980s, Python is renowned for its readability and clean syntax, making it an ideal choice for both beginners and seasoned developers.



#### **GUIDO VAN ROSSUM**

One of Python's key strengths lies in its extensive standard library, offering a wide array of modules and packages that simplify complex tasks. Its object-oriented programming paradigm promotes code reusability and modularity, fostering efficient and maintainable software development practices.

Python's popularity in data science stems from libraries like NumPy, Pandas, and Matplotlib, empowering scientists and analysts to handle large datasets and visualize results seamlessly. Additionally, frameworks such as Django and Flask facilitate web development, enabling the creation of robust and scalable applications.

## **3.2. Machine Learning**

Machine learning (ML) is a subset of artificial intelligence (AI) that focuses on the development of algorithms and models that enable computers to learn from and make predictions or decisions based on data. The primary goal of machine learning is to enable computers to automatically learn and improve from experience without being explicitly programmed.

### **3.2.1. Types of Machine Learning**

#### **1. Supervised Learning:**

- The algorithm is trained on a labelled dataset, where the input data and corresponding are provided.
- It learns to map the input to the output, making predictions on new, unseen data.

#### **2. Unsupervised Learning:**

- The algorithm is given unlabelled data and must find patterns or structures on its own.
- Common tasks include clustering and dimensionality reduction.

#### **3. Reinforcement Learning:**

- The algorithm learns by interacting with an environment and receiving feedback in the form of rewards or penalties.
- It aims to learn a sequence of actions that maximizes the cumulative reward.



### 3.3. Installed Libraries

- **Pandas:** Used for data manipulation and analysis. Common tasks include reading/writing data from/to various file formats (CSV, Excel, SQL databases), data cleaning, filtering, grouping, and visualization.
- **NumPy:** Fundamental package for numerical computing with Python. Commonly used for mathematical operations on arrays, linear algebra, random number generation, Fourier transform, etc.
- **Math:** Provides mathematical functions defined by the C standard. Useful for basic mathematical operations like trigonometry, logarithms, exponentiation, etc.
- **JSON:** Used for parsing and encoding JSON data. Commonly used for data interchange between a server and web application.
- **Time:** Provides various time-related functions. Useful for measuring execution time, dealing with time stamps, conversions between different time formats, etc.
- **Matplotlib:** Comprehensive library for creating static, animated, and interactive visualizations in Python. Commonly used for creating plots, histograms, scatter plots, etc.
- **Seaborn:** Built on top of Matplotlib, Seaborn provides a high-level interface for drawing attractive and informative statistical graphics. Commonly used for statistical data visualization, such as heatmaps, violin plots, pair plots, etc.

- **Scikit-learn (sklearn):** Simple and efficient tools for predictive data analysis. Commonly used for building and training machine learning models, including classification, regression, clustering, dimensionality reduction, etc.
- **Joblib:** Library for lightweight pipelining in Python. Commonly used for saving and loading Python objects (such as machine learning models) to/from disk.
- **Scipy:** Scientific library for mathematics, science, and engineering. Commonly used for numerical integration, optimization, interpolation, Fourier transforms, signal processing, etc.
- **Flask:** Lightweight WSGI web application framework. Commonly used for building web applications and RESTful APIs in Python.
- **Hashlib:** Provides cryptographic hash functions. Commonly used for secure password storage, digital signatures, message digest, etc.
- **Cassandra-driver:** Python driver for Apache Cassandra. Commonly used for interacting with Cassandra databases, executing queries, and managing cluster connections.
- **Re (Regular Expressions):** Module for working with regular expressions. Commonly used for pattern matching, searching, and replacing text strings based on patterns.
- **TextBlob:** Simple API for common natural language processing (NLP) tasks such as part-of- speech tagging, noun phrase extraction, sentiment analysis, classification, translation, etc.

### 3.3.1. Requirements.txt

The requirements.txt file is a commonly used convention in the python programming ecosystem to specify the dependencies of a python project. This file typically contains a list of python packages along with their versions that are required for the project to run successfully.

The use of a requirements.txt file in a python project offers several benefits:

- **Dependency Management**
- **Reproducibility**
- **Collaboration**

#### **Installations of libraries**

To download and install the libraries specified in the requirements.txt file, you can use the pip package manager. Use this command to download all libraries.

**“pip install -r requirements.txt”**

To know which libraries are downloaded uses this command **“pip freeze”**

## 4. IMPLEMENTATION

### 4.1. TextBlob

TextBlob module is a Python library and offers a simple API to access its methods and perform basic NLP tasks. It is built on the top of NLTK module. Install TextBlob using the following commands in terminal.

- **pip install -U textblob**
- **python -m textblob.download\_corpora**

This will install TextBlob and download the necessary NLTK corpora. The above installation will take quite some time due to the massive amount of tokenizers, chunkers, other algorithms, and all of the corpora to be downloaded. Some terms that will be frequently used are:

- **Corpus** – Body of text, singular. Corpora is the plural of this.
- **Lexicon** – Words and their meanings
- **Token** – Each “entity” that is a part of whatever was split up based on rules. For examples, each word is a token when a sentence is “tokenized” into words. Each sentence can also be a token, if you tokenized the sentences out of a paragraph.

#### 4.1.1. Part of Speech Tagging using TextBlob

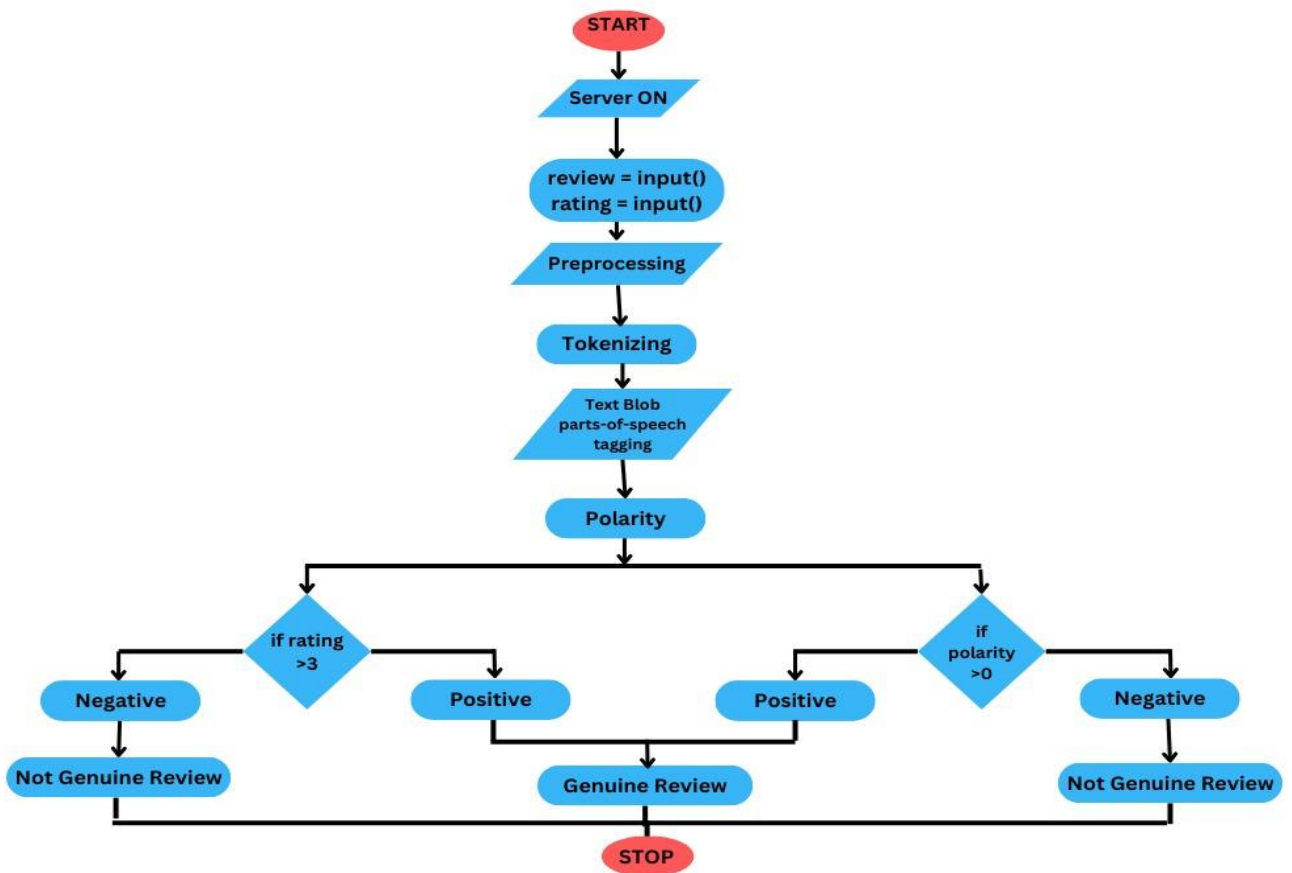
TextBlob module is used for building programs for text analysis. One of the more powerful aspects of the TextBlob module is the Part of Speech tagging.

In corpus linguistics, part-of-speech tagging (POS tagging or PoS tagging or POST), also called grammatical tagging or Word-category disambiguation.

**For Example:****Input:** Everything is all about money.**Output:** [('Everything', 'NN'), ('is', 'VBZ'), ('all', 'DT'), ('about', 'IN'), ('money', 'NN')]

Tags	Description
NN	noun, singular 'desk'
NNS	noun, plural 'desk'
CC	coordinating conjunction
CD	cardinal digit
EX	existential there (like: "there is" ... think of it like "there exists")
DT	Determiner
FW	foreign word
IN	preposition/subordinating conjunction
JJ	adjective 'big'
JJR	adjective, comparative 'bigger'
JJS	adjective, superlative 'biggest'
LS	list marker 1)
MD	modal could, will
NNP	proper noun, singular 'Harrison'
NNPS	proper noun, plural 'Americans'
PDT	predeterminer 'all the kids'
POS	possessive ending parent's
PRP	personal pronoun I, he, she
PRP\$	possessive pronoun my, his, hers
RB	adverb very, silently,
RBR	adverb, comparative better
RBS	adverb, superlative best
RP	particle give up
TO	To go 'to' the store.
UH	interjection errrrrrrm
VB	verb, base form take
VBD	verb, past tense took
VBN	verb, past participle taken
VBP	Verb, sing. present, non-3d take
VBZ	Verb, 3rd person sing. present takes
WDT	wh-determiner which
WP	wh-pronoun who, what
WP\$	possessive wh-pronoun whose

#### 4.1.2. Fake Review Detection flowchart



## 4.2. Collaborative Filtering in Machine Learning

### Recommendation system

There are a lot of applications where websites collect data from their users and use that data to predict the likes and dislikes of their users. This allows them to recommend the content that they like. Recommender systems are a way of suggesting similar items and ideas to a user's specific way of thinking.

There are basically two types of recommender Systems:

**Collaborative Filtering:** Collaborative Filtering recommends items based on similarity measures between users and/or items. The basic assumption behind the algorithm is that users with similar interests have common preferences.

**Content-Based Recommendation:** It is supervised machine learning used to induce a classifier to discriminate between interesting and uninteresting items for the user.

### Collaborative Filtering

In Collaborative Filtering, we tend to find similar users and recommend what similar users like. In this type of recommendation system, we don't use the features of the item to recommend it, rather we classify the users into clusters of similar types and recommend each user according to the preference of its cluster.

There are basically four types of algorithms or say techniques to build Collaborative filtering-based recommender systems:

- Memory-Based
- Model-Based
- Hybrid
- Deep Learning

### 4.2.1. Memory-based Collaborative Filtering

Memory-based CF is one method that calculates the similarity between users or items using the user's previous data based on ranking. The main objective of this method is to describe the degree of resemblance between users or objects and discover homogenous ratings to suggest the obscured items.

Memory-based CF consists of the following two methods:

- a) **User-Based Collaborative Filtering**
- b) **Item-based Collaborative Filtering**

### 4.2.2. User-Based Collaborative Filtering

User-Based Collaborative Filtering is a technique used to predict the items that a user might like based on the ratings given to those items by other users with similar tastes. Many websites employ collaborative filtering to build their recommendation systems.

In this method

1. **Identify the Target User:** Determine the user for whom recommendations are being made. In the given example, Jack is the target user.
2. **Find Similar Users:** Locate users who have rated items similarly to the target user. This step involves calculating the similarity between users using various similarity measures such as:
  - **Cosine Similarity**
  - **Pearson Correlation**
  - **Euclidean Distance**
3. **Explore Interacted Items:** Once similar users are identified, examine the items they have interacted with and rated.



4. **Forecast Rankings:** Predict the ranking of unseen items for the target user based on the ratings of similar users.
5. **Recommendation:** Recommend items to the target user based on the forecasted rankings. If the predicted rankings are above a certain threshold, suggest these items to the target user.

### 4.2.3. Formulas Used in User-Based CF:

#### 1. Cosine Similarity:

- $$\text{Cosine Similarity}(u, v) = \frac{\sum_{i=1}^n u_i \times v_i}{\sqrt{\sum_{i=1}^n u_i^2} \times \sqrt{\sum_{i=1}^n v_i^2}}$$

- Where  $u$  and  $v$  are vectors representing ratings given by two users, and  $n$  is the number of items.

#### 2. Pearson Correlation:

- $$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x}) \times (y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \times \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

- Where  $x$  and  $y$  are vectors representing ratings,  $\bar{x}$  and  $\bar{y}$  are the means of  $x$  and  $y$  respectively, and  $n$  is the number of items.

#### 3. Euclidean Distance:

- $$d(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

- Where  $p$  and  $q$  are vectors representing ratings.

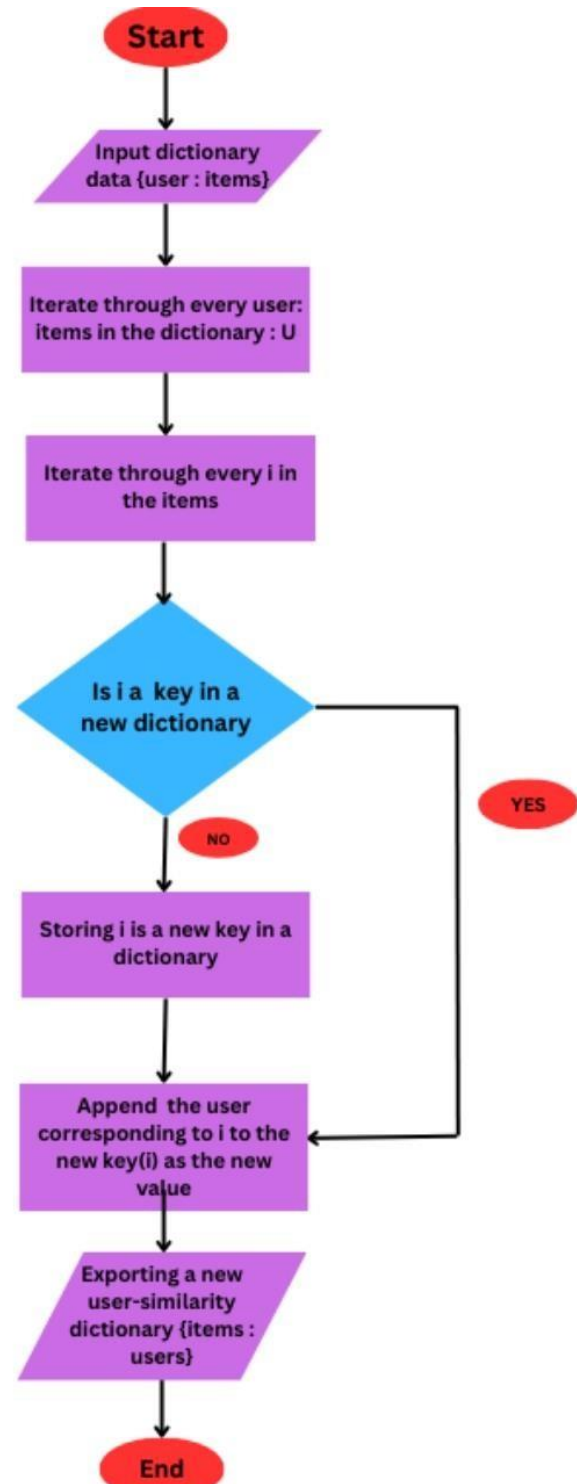
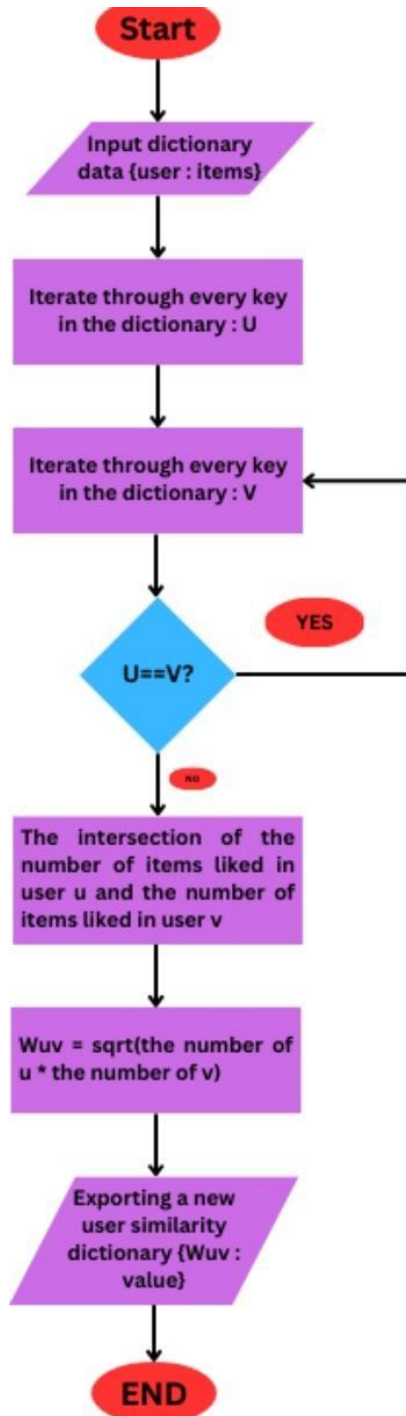
#### **4.2.4. Algorithms Used in User-Based CF:**

Common algorithms for finding similar users in user-based collaborative filtering include:

- **K-Nearest Neighbors (KNN)**
- **Singular Value Decomposition (SVD)**
- **Alternating Least Squares (ALS)**

These algorithms are used to calculate similarity between users and provide recommendations based on their preferences.

#### 4.4.2. User-Bases Collaborative Filtering



### 4.3. Singular Value Decomposition (SVD) Algorithm for Product Recommendation

Singular Value Decomposition (SVD) is a dimensionality reduction technique that is widely used in recommendation systems, particularly in collaborative filtering approaches. It decomposes a matrix into three other matrices in such a way that the original matrix can be approximated well by the product of these matrices. In the context of recommendation systems, SVD is used to reduce the dimensionality of the user-item rating matrix and discover latent factors that represent user preferences and item characteristics.

Here's how the SVD algorithm works for product recommendation:

**1. User-Item Rating Matrix:** Start with a large matrix where rows represent users, columns represent items, and each cell represents the rating given by a user to an item (if available).

**2. SVD Decomposition:** Apply SVD to decompose the user-item rating matrix into three matrices:

- **U (User Matrix):** This matrix represents users and their associations with latent factors. Each row corresponds to a user, and each column represents the strength of the association between the user and a latent factor.
- **$\Sigma$  (Singular Values):** This is a diagonal matrix containing the singular values arranged in descending order. Singular values represent the importance of each latent factor.
- **$V^T$  (Transpose of Item Matrix):** This matrix represents items and their associations with latent factors. Each row corresponds to an item, and each column represents the strength of the association between the item and a latent factor.

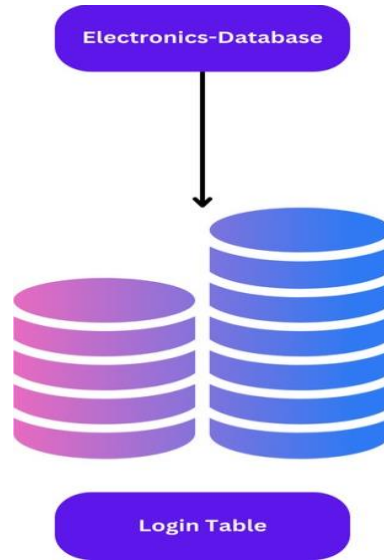
**3. Dimensionality Reduction:** Typically, only the top  $k$  singular values and their corresponding columns in  $U$  and  $V^T$  are retained to reduce the dimensionality of the matrices. This helps in capturing the most significant latent factors while discarding noise and less relevant information.

**4. Recommendation Generation:**

- To generate recommendations for a particular user, first, calculate the predicted ratings for all items that the user has not rated.
- This is done by taking the dot product of the user's row vector in  $U$  (representing the user's preferences across latent factors) and the column vectors in  $V^T$  (representing the latent factors associated with each item).
- The predicted ratings can be adjusted to fall within a desired range (e.g., 1 to 5 for rating scales) and can be used to rank the items.
- The top-ranked items are recommended to the user as potential choices.

**5. Model Tuning:** The number of latent factors ( $k$ ) retained during dimensionality reduction is a hyperparameter that can be tuned using techniques like cross-validation to optimize recommendation performance.

## 4.4. Database Schema




























Cassandra is a distributed NoSQL database known for its high availability, scalability, and performance. It's suitable for handling large volumes of data across multiple nodes in a cluster, making it ideal for applications requiring low-latency access, high throughput, and flexibility in data modeling.

**Login Table:** [id, name, email, password] these are the columns present in the login table and the use case if user wants to login into the website then the user must be there in the login table otherwise they have to register first then the details will be appended to this table and id is generated automatically and password will be saved in the form of hexadecimal format so that even admin cannot be able to see the password.


## 4.5. Source Code


### File Structure

-  app.py
-  connect\_database.py
-  data.py
-  dummy.py
-  Electronics\_ecommerce-token.json
-  fake-product-review-detection.ipynb
-  fake\_review\_detection.py
-  pivot\_df.joblib
-  preds\_df.joblib
-  product\_recommendation.py
-  recommender.ipynb
-  secure-connect-electronics-ecommerce.zip
-  static/
  -  bot.css
  -  bot.js
  -  css/
    -  animate.css
    -  bootstrap.min.css
    -  font-awesome.min.css
    -  fontawesome-stars.css
    -  helper.css
  -  images/
    -  ui-bg\_glass\_75\_d0e5f5\_1x400.png
    -  .....


 jquery-ui.min.css


 ...


 fonts/


 fontawesome-webfont3e6e.eot

 ....


 images/


 about-us/

 icon/


 1.png


 ....


 banner/


 1\_1.jpg


 ...

 bg-banner/

 1.jpg


 2.jpg

 blog-banner/

 1.jpg

 ...


 large-size/

 1.jpg


 ...

 favicon.png

 featured-product/

 1.jpg

 ...

 menu/



- 📁 flag-icon/
  - 📄 1.jpg
  - 📄 2.jpg
  - 📄 li-bg-menu.jpg
- 📁 logo/
  - 📄 1.jpg
  - 📄 2.jpg
- 📁 payment/
  - 📄 1.png
- 📁 product/
  - 📁 large-size/
    - 📄 1.jpg
    - 📄 ....
  - 📁 small-size/
    - 📄 1.jpg
    - 📄 ....
- 📁 product-details/
  - 📄 1.jpg
  - 📄 admin.png
  - 📄 user.png
- 📁 shipping-icon/
  - 📄 1.png
  - 📄 ....
- 📁 slider/
  - 📄 1.jpg
  - 📄 ...
- 📁 static-top/

-  1.jpg
-  team/
  -  1.png
  -  ...
-  wishlist-thumb/
  -  1.jpg
  -  ...
-  js/
  -  ajax-mail.js
  -  ....
  -  vendor/
    -  jquery-1.12.4.min.js
    -  ...
-  svd\_model.joblib
-  templates/
  -  about-us.html
  -  chatbot.html
  -  contact.html
  -  index.html
  -  login-register.html
  -  shop-3-column.html
  -  shopping-cart.html
  -  single-product.html
  -  wishlist.html

## 4.6. File Explanation

### Files:

**1.app.py:** This is likely the main Python application file, where the main logic of your application resides.

**2. connect\_database.py:** It probably contains code to connect to a database.

**3. data.py:** Likely contains data processing or manipulation functions.

**4. dummy.py:** Possibly a file used for testing or placeholder purposes.

**5. Electronics\_ecommerce-token.json:** Likely a JSON file containing some authentication token related to an electronics e-commerce application.

**6. fake\_product\_review\_detection.ipynb:** A Jupyter Notebook file (.ipynb) used for detecting fake product reviews.

**7. fake\_review\_detection.py:** Python script for detecting fake reviews.

**8. pivot\_df.joblib:** This is likely a serialized DataFrame object saved using joblib, possibly for later loading and analysis.

**9. preds\_df.joblib:** Similar to pivot\_df.joblib, it's another serialized DataFrame object.

**10. product\_recommendation.py:** Python script for generating product recommendations.

**11. recommender.ipynb:** A Jupyter Notebook file for product recommendation.

**12.secure-connect-electronics-ecommerce.zip:** Possibly contains secure connection configurations or certificates for the electronics e-commerce application.

## **Directories:**

- 1. `static/`:** Directory containing static files such as CSS, JavaScript, and images for the frontend of the application.
- 2. `static/css/`:** Contains CSS files for styling the frontend.
- 3. `static/fonts/`:** Contains font files used in the application.
- 4. `static/images/`:** Contains various images used in the application, organized into subdirectories based on their usage.
- 5. `static/js/`:** Contains JavaScript files for frontend functionality.
- 6. `templates/`:** Contains HTML templates for different pages of the application, such as the homepage, about us page, contact page, etc.

## **Other Files:**

- **`svd_model.joblib`:** Likely contains a serialized machine learning model, possibly for recommendation purposes.

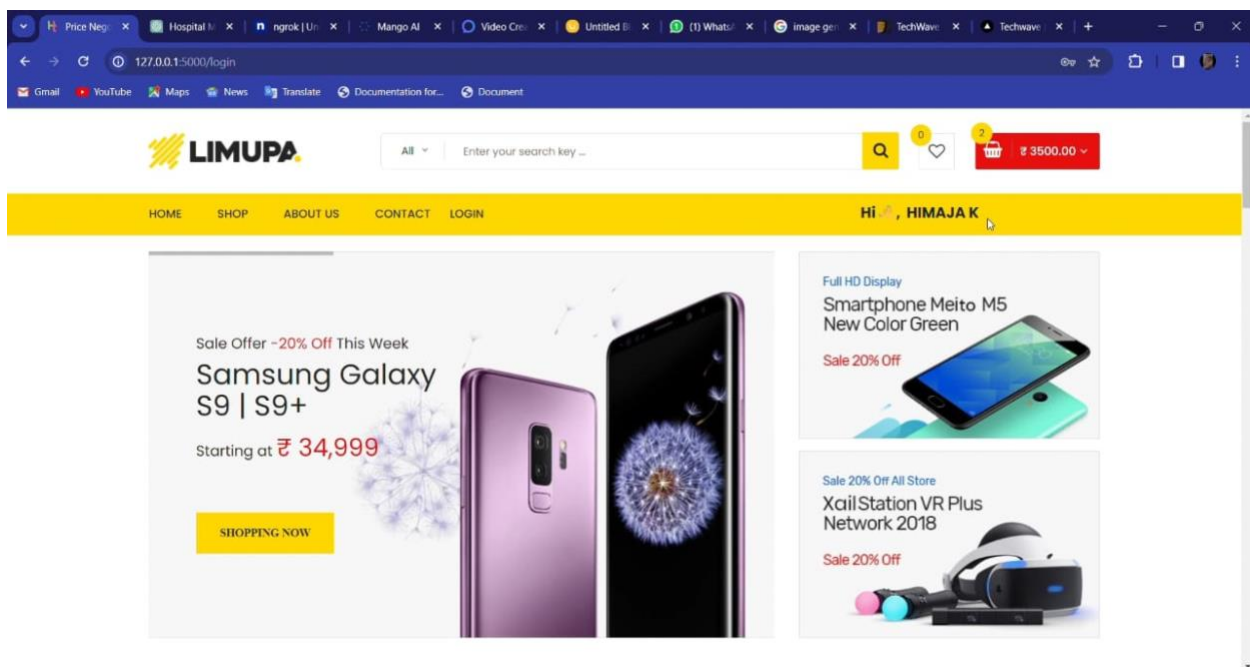
## 5. RESULTS

### 5.1. Output screens

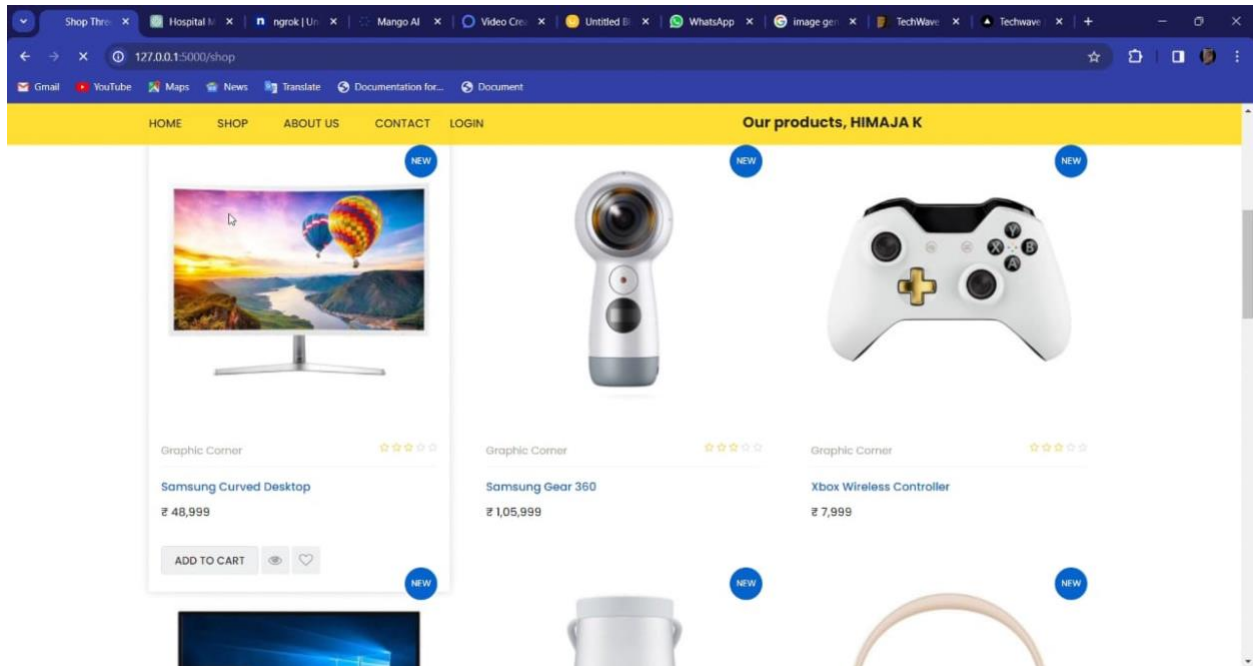
#### Login page

The screenshot shows the LIMUPA website interface. At the top, there's a navigation bar with the LIMUPA logo, a search bar, and a shopping cart icon showing ₹ 3500.00. Below this is a yellow navigation bar with links: HOME, SHOP, ABOUT US, CONTACT, and LOGIN. The main content area is divided into two columns. The left column is titled 'Login' and contains fields for 'Email Address\*' (with the value 'himojakadarti23@gmail.com') and 'Password'. There's a 'Remember Me' checkbox and a 'Forgot password?' link. A 'LOGIN' button is at the bottom. The right column is titled 'Register' and contains fields for 'First Name', 'Last Name', 'Email Address\*', 'Password', and 'Confirm Password'. A 'REGISTER' button is at the bottom.

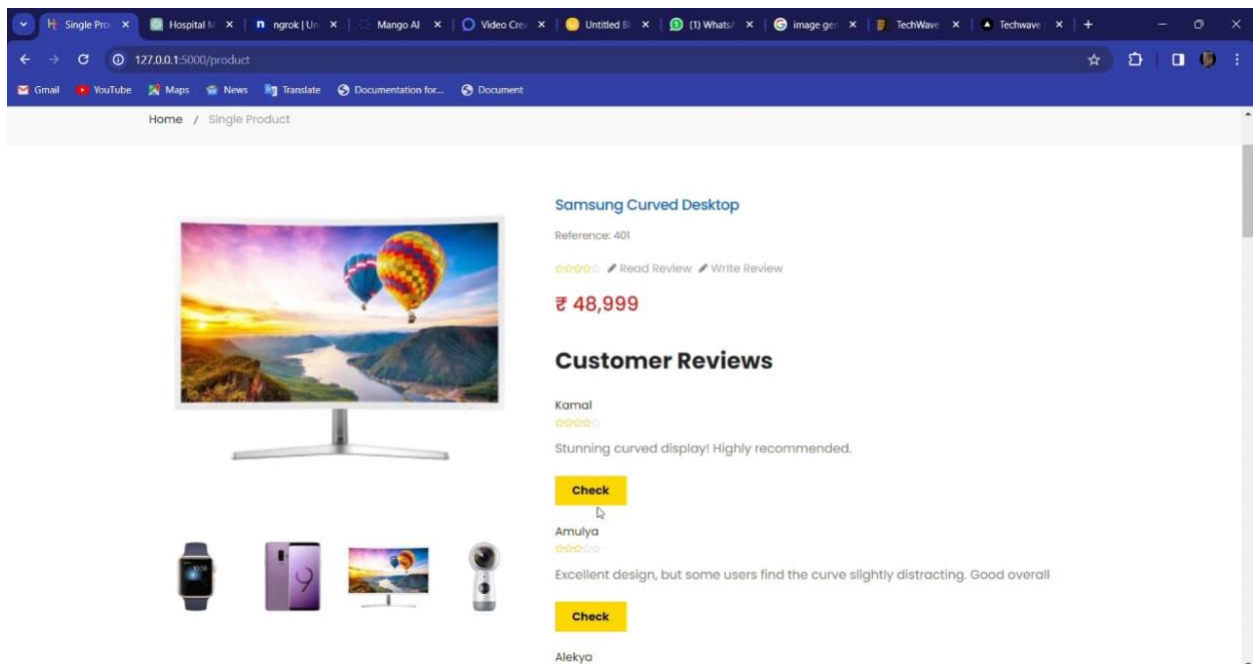
#### Home page



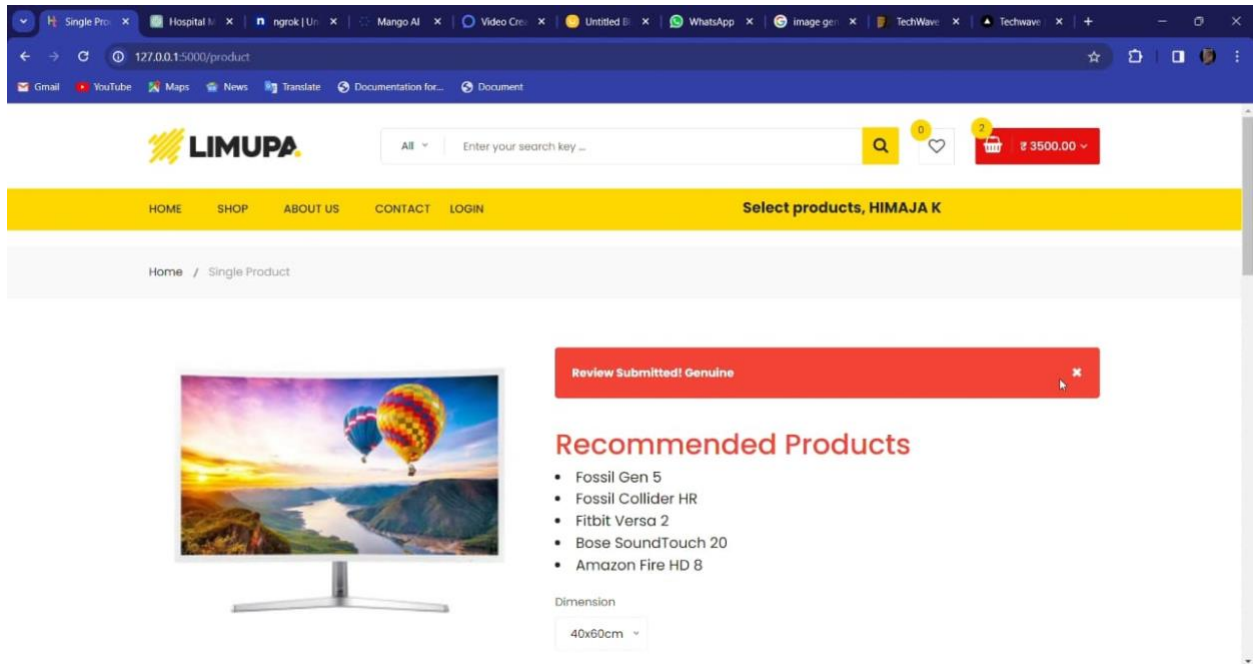
## Shopping page



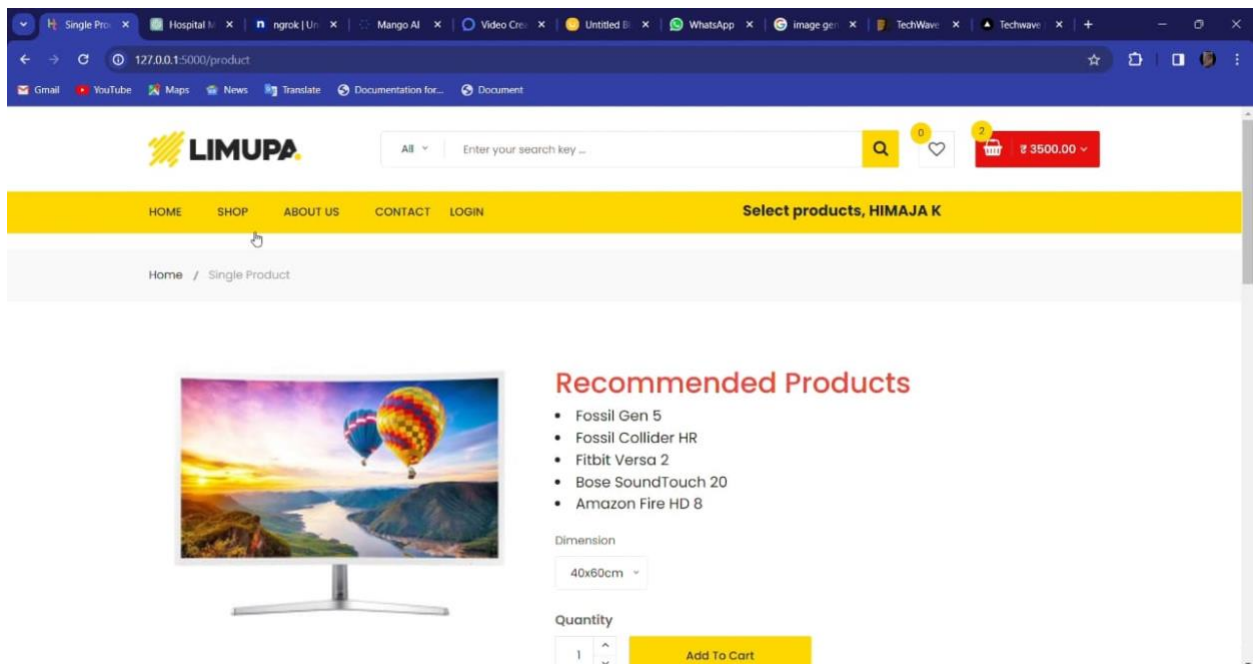
## Product information page



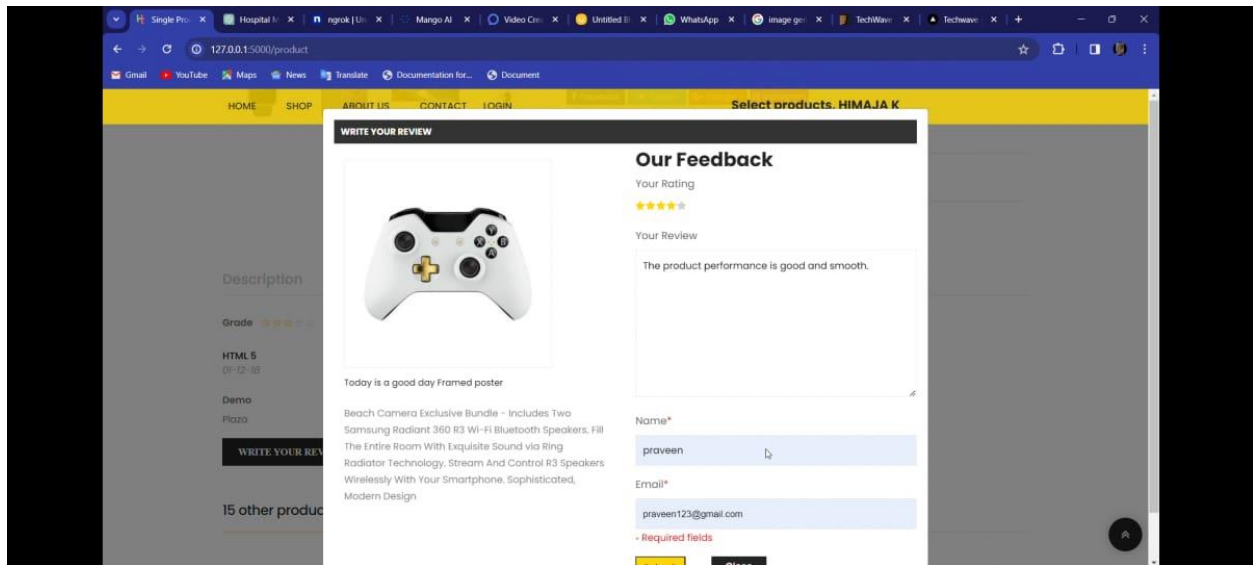
## Fake review validation page



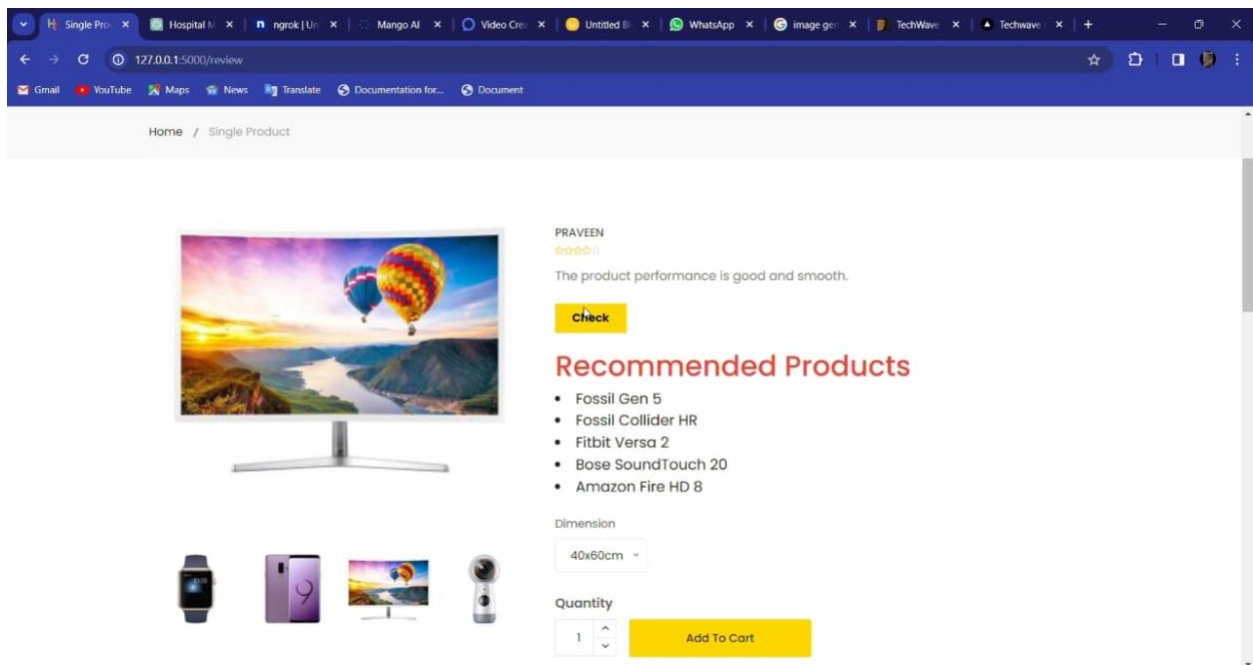
## Product recommendation page



## Review writing page



## Review updated page





## **6. CONCLUSION**

In conclusion, our research offers a comprehensive framework to address the challenges of fake reviews and elevate product recommendations in e-commerce. Our next steps involve deploying the system on AWS to ensure scalability and reliability. Additionally, we aim to bolster our database capacity to accommodate extensive product details, enriching the user experience. Integrating a customer service chatbot will provide timely assistance and enhance user engagement, further enhancing trust in online shopping platforms. By continuously refining our approach and embracing technological advancements, we aspire to contribute significantly to fostering transparency and reliability in the digital marketplace. Ultimately, our efforts seek to empower consumers with trustworthy information and facilitate seamless transactions, thereby promoting confidence and satisfaction in the online shopping experience.

## REFERENCE

Pandas: <https://pandas.pydata.org/docs/>

NumPy: <https://numpy.org/doc/>

Math (Python Standard Library): <https://docs.python.org/3/library/math.html>

JSON (Python Standard Library): <https://docs.python.org/3/library/json.html>

Time (Python Standard Library): <https://docs.python.org/3/library/time.html>

Matplotlib: <https://matplotlib.org/stable/contents.html>

Seaborn: <https://seaborn.pydata.org/tutorial.html>

Scikit-learn (sklearn): <https://scikit-learn.org/stable/documentation.html>

Joblib: <https://joblib.readthedocs.io/en/latest/>

SciPy: <https://docs.scipy.org/doc/>

Flask: <https://flask.palletsprojects.com/en/2.1.x/>

Hashlib : <https://docs.python.org/3/library/hashlib.html>

Cassandra-driver: <https://cassandra.apache.org/doc/latest/>

Re (Python Standard Library): <https://docs.python.org/3/library/re.html>

TextBlob: <https://textblob.readthedocs.io/en/dev/>