Program of Data Science

Faculty of Computing and Data Science

Alexandria University

Graduation Project I

RECSEN

Submitted By:

1. Mohamed Adel Abdel Raouf Gomaa 20201494864
2. Sohayb Emad Mohamed Fawzy 20201495517
3. Mohamed ElAraby Abdel Aziz 20201496716
4. Abdel Rahman Sami Abdel Raziq 20201495525
5. Abdel Rahman Sayed Abdel Nabi Heiba 20201470085
6. Mohamed Murad Samir 20201210256
7. Mohammed Adel Ali 20201368842
8. Youssef Mohamed Ali Ahmed 20201446532

Supervised By

Prof. Muhammed Walid

2023-2024

Problem statements:

- The main focus of this project is to use data science techniques on products data, namely recommendation system and sentiment analysis.

- The recommendation system aims to provide personalized suggestions of products to customers based on their preferences, ratings, and purchase history. The system will use machine learning to generate relevant recommendations.

- The sentiment analysis aims to extract the sentiment or emotion expressed in the customer reviews of the products. The system will use natural language processing (NLP) techniques to transform the text into numerical features and then apply classification algorithms to predict the sentiment polarity.

- The project will also include a simple web app that allows the user to see the data, the results, and the models in an interactive way.

ACKNOWLEDGEMENTS:

We have been fortunate to have help, support, and encouragement from many people. We would like to acknowledge them for their Cooperation. First, of all, thanks are due to ALLAH for getting this work done. Second, we would like to express our appreciation to our Supervisor: Dr. Muhammed Walid. For all you have done, which we will never forget. We truly appreciate you and the time you spent helping us on many occasions. Thank you very much for the course. We enjoyed every minute of your lecture as well as your marvelous sense of humor.

We are also grateful to our classmates for their encouraging, late-night feedback and moral support.

قال تعالي:

وَلَا تَحْسَبَنَّ الَّذِينَ قُتِلُوا فِي سَبِيلِ اللَّهِ أَمْوَاتًا بَلْ أَحْيَاءٌ عِنْدَ رَبِّهِمْ يُرْزَقُونَ

اهداء الي اخواننا في فلسطين

نصرهم الله و تقبل موتاهم شهداء

Motivation

Imagine you are looking for a new laptop online. You type in your query and hit enter. You are greeted by millions of results, each with different features, prices, ratings, and reviews. How do you decide which one to buy? How do you know if you are getting the best deal? How do you avoid wasting hours of your precious time scrolling through endless pages of products?

This is the problem that my colleagues and I faced as online shoppers. We are a team of data science students who love to explore and analyze data. We wanted to create a project that would make online shopping easier and more enjoyable for ourselves and other users.

We decided to build a recommendation system that would suggest products based on the user's preferences and behavior. We will use various data sources, such as product descriptions, user ratings, purchase history, and browsing patterns, to train our machine learning models. We also will add a sentiment analysis component that would show the user the reviews of the recommended products and their positive or negative tone. We will use natural language processing techniques to extract the opinions and emotions of the reviewers. We also will provide visualizations to summarize the feedback from the users who bought or used the products.

Our project will be designed to facilitate the user's experience by providing personalized, relevant, and trustworthy recommendations. It helps the user to find the best product for their needs, budget, and taste. It also saves the user time and effort by showing them the reviews and sentiment analysis results of the recommended products.

Contents

Problem statements: 2
ACKNOWLEDGEMENTS: 3
Motivation 4
Table 1 Nomenclature 7
List of Figures 8
Chapter 1: Introduction 9
1.1 Introduction 9
1.2 Related Work 9
1.3 Our goal 10
Chapter 2: Recommendation System 11
2.1 What is recommendation system? 11
2.2 recommendation system phases 11
2.3 Types of recommendation system 11
2.3.1Types of personalized recommender systems 13
Chapter 3: Sentimental Analysis 23
3.1 What is sentiment Analysis? 23
3.1 Types of Sentiment Analysis 24
3.2 Fine-grained or Aspect-level 27
3.3 Why Aspect-level sentiment analysis? 28
3.4 Sentiment Analysis Scope: 28
3.5 Text Processes 29
Chapter 4: System Analysis and System Design 30
4.1 System Development Life Cycle: 30
4.2 System Analysis Stages: 32
4.3 Software Failure (Risks): 37
4.4 UML Diagrams: 38
4.4.1 Use Case Diagram 39
4.4.2 Data Flow Diagram 44
4.4.3 Sequence Diagram 48

4.4.4 ERD Model 53
Chapter 5: Dataset 59
Chapter 6: Implementation 61
6.1Web tools that will be used in our project 61
6.2 Code review: 64
Chapter 7: References: 68

NOMENCLATURE

Table 1 Nomenclature

UI: User Interface
OS: Operating System
AI: Artificial Intelligence
ML: Machine Learning
DL: Deep Learning
NLP: Natural Language Process
SDLC: System Development Life Cycle
DDS: Design Document Specification
SRS: Software Requirement Specification
UML: Unified Modeling Language
DFD: Data Flow Diagram
SQL: Structured Query Language
ERD: Entity Relationship Diagram
RDBMS: Relational Database Management System

List of Figures

Chapter two:

Figure 1/Types of recommendation system 12
Figure 2/User-based collaborative 15
Figure 3/Item-based collaborative 16
Figure 4/user-based matrix 16
Figure 5/Person Correlation formula 17
Figure 6/Matrix Factorization 19
Figure 7/User latent factors 19
Figure 8/Hybrid Recommender 21
Figure 9/Sentiment Analysis 23
Figure 10/Amazon reviews 24
Figure 11/Data processing 26
Figure 12/Agile methodology 31
Figure 13/Black Box Testing 35
Figure 14/ usecase actor 39
Figure 15/use case extend 40
Figure 16/Use case include 40
Figure 17/use case Association 41
Figure 18/UML System 41
Figure 19/ Use case Dependency 41
Figure 20/Use cse Generalization 42
Figure 21/Use case Realization 42
Figure 22/Use Case Diagram 43
Figure 23/DFD Process 44
Figure 24/DFD Data Flow 45
Figure 25/DFD External Entities 45
Figure 26/DFD Data Store 45
Figure 27/DFD Level 0 46
Figure 28/DFD Level 1 47
Figure 29/sequence object 49
Figure 30/Sequnce Lifeline 49
Figure 31/sequence Actor 50
Figure 32/sequence Entity 50
Figure 33/sequence Activation bar 51
Figure 34/sequence Synchronous message 51
Figure 35/sequence Alternative 51
Figure 36/sequence Diagram 52
Figure 37/ERD Entity 54
Figure 38/ERD Entity Attributes 54
Figure 39/ERD Model 56

Chapter 1: Introduction

1.1 Introduction

We introduce an application that helps people to save effort and time by using machine learning (Recommendation System) and deep learning (Sentimental Analysis).

We all know after covid-19 period all of us go to the online shopping. The online shopping sometimes is massy because of the many of products are shown and we don't the pros and cons of the products we want to buy. This problem is common right now so we decided to build a website page to avoid this massy include two algorithms recommendation system and sentimental analysis. Recommendation system that will suggest products based on the user's preference and behavior. The sentimental analysis that will allow user to show the feedback from other users and the rate of the products to make sure that his decision whatever he will buy this product or not is the prefect decision for him.

1.2 Related Work

During the search process, we found some applications that like our idea and now, we will illustrate some of the applications that were introduced recently within the scope of our idea.

AliExpress*:*

Is a global online marketplace that offers a wide range of products from various categories, such as food, home appliances, computer & office, home improvement, sports & entertainment, and more. You can find deals, discounts, coupons, and free shipping on millions of items from different sellers and brands

Gazelle*:*

Is a recommerce company that offers a platform for buying and selling pre-owned consumer electronics, such as smartphones, tablets, and laptops. Gazelle inspects every product before it is sold and guarantees its quality as certified pre-owned. Gazelle also helps reduce e-waste by giving new life to used devices. You can visit Gazelle's website to see their current offers and prices.

Feature Matrix:

App Name\ feature	Context of data	Rating System	Ability to trade for user	Sentimental Analysis
recsen	Only about electronics	Exist	No Ability	Use Sentimental Analysis
gazelle	Only about electronics	Exist	Ability To buy and sell	Doesn't use sentimental analysis
AliExpress	Multifaceted	Exist	Ability to buy only	Doesn't use sentimental analysis

1.3 Our goal

Recommendations systems analyze user behavior data to suggest personalized content that aligns with their interests, with the goal of improving the user experience, engagement and sales conversion rates for businesses. Sentiment analysis provides insights into customer opinions and reviews to better understand satisfaction levels and identify areas for experience improvement. Together they aim to enhance the user experience through relevant suggestions and address issues, while helping businesses make data-driven decisions to increase engagement, sales and customer retention. The challenges include handling lack of data for new users and gaining meaningful sentiment insights at scale.

Chapter 2: Recommendation System

In this chapter we will write about the recommendation system, its importance, types and how the recommendation system work.

2.1 What is recommendation system?

Recommender systems are sophisticated algorithms designed to provide product-relevant suggestions to users.

Recommender systems play a paramount role in enhancing user experiences on various online platforms, including e-commerce websites, streaming services, and social media.

Essentially, recommender systems aim to analyze user data and behavior to make tailored recommendations.

2.2 recommendation system phases

Data collection:

Recommender systems start by gathering data on user interactions, preferences, and behaviors. This data can include browsing history, ratings.

Data processing:

Once collected, they process the data to extract meaningful patterns and insights. This involves techniques like data cleaning, transformation, and feature engineering.

Algorithm generation:

Depending on the data, a specific recommender algorithm is applied to generate recommendations. Common types include collaborative filtering, content-based filtering, and hybrid methods.

Presentation:

Finally, the top-ranked items are then presented to the user.

2.3 Types of recommendation system

Now that we've learned how recommender systems work, let's explore the basic types of recommenders – non-personalized and personalized.

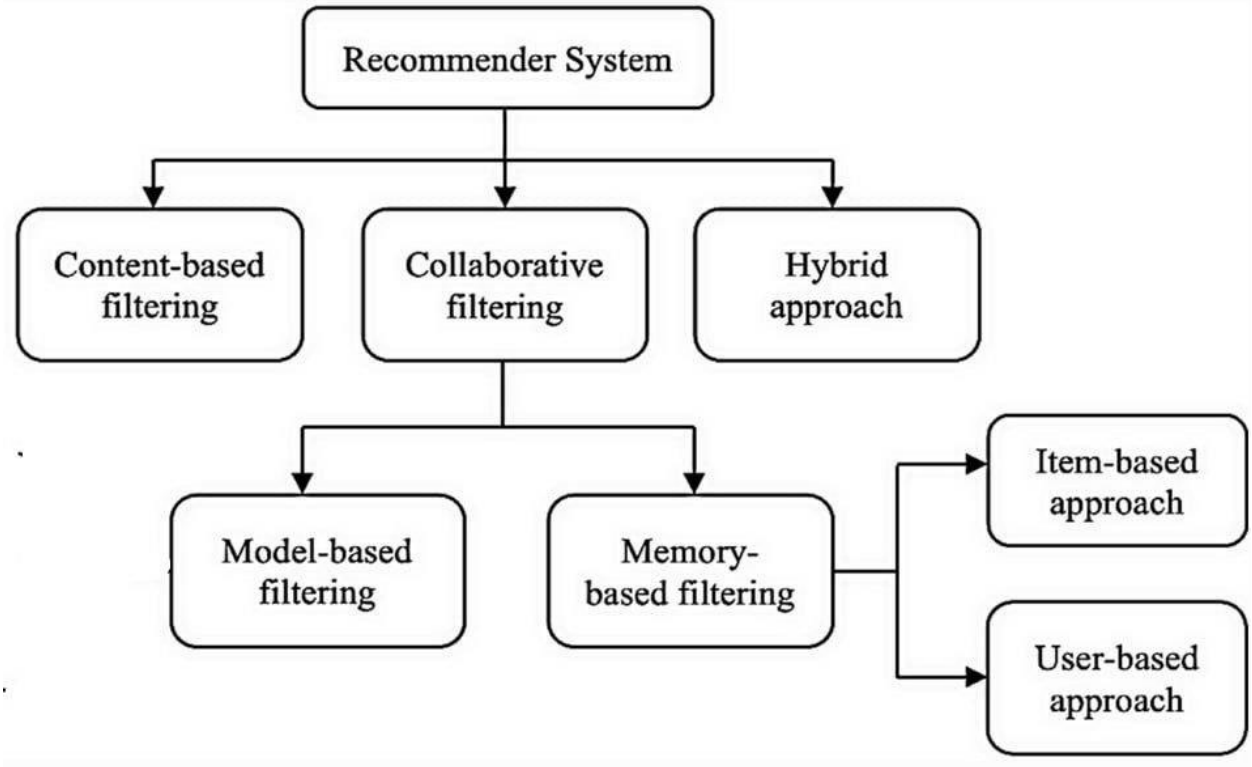


Figure 1/Types of recommendation system

- Non-personalized recommender systems:

Non-personalized recommendation systems provide recommendations to users without taking into account their individual preferences or behavior.

These systems make recommendations based on the characteristics of items or content themselves rather than relying on user-specific data.

A popular non-personalized recommender is the popularity-based recommender which recommends the most popular items to the users, for instance:

Top-10 movies, Top 5 trending products, New products.

However, non-personalized recommendation systems have their limitations, including the inability to provide highly tailored recommendations. They may be a good option for a first step in the process of personalization, but you shouldn't stop there.

Once you gather enough data about the user in question, personalized offers and recommendations are the logical next step.

- Personalized recommender systems

Personalized recommendation systems are designed to provide tailored recommendations to individual users based on their past behavior, preferences, and demographic information.

Based on the user's data such as ratings, personalized recommenders try to understand and predict what items or content a specific user is likely to be interested in. In that way, every user will get customized recommendations.

At this point, you might ask yourself – what makes a good recommendation?

Well, a good recommendation:

Is personalized (relevant to that user),

Is diverse (includes different user interests),

Doesn't recommend the same items to users for the second time,

There are a few types of personalized recommendation systems, including content-based filtering, collaborative filtering, and hybrid recommenders.

Let's explore them in greater detail.

2.3.1Types of personalized recommender systems

Personalized recommender systems can be categorized into several types, each with its own methods and techniques for providing tailored recommendations.

These include:

- Content-based filtering,
- Collaborative filtering
- Hybrid recommenders.

Content-based filtering:

Suggest items similar to those the user has shown interest in, based on the item's description or features

For example, if a user has already view a product from a certain brand, you assume that they have a preference for that brand. Also, there is a probability that they will buy a similar product in the future.

Pros of the content-based approach:

The content-based approach is one of the common techniques used in personalized recommendation systems. It has its advantages and disadvantages, which are important to consider when deciding to implement this approach.

Let's take a look at some of its most obvious advantages first:

Less cold-start problem:

Content-based recommendations can effectively address the "cold-start" problem, allowing new users or items with limited interaction history to still receive relevant recommendations.

Reduced data privacy concerns:

Since content-based systems primarily use item attributes, they may not require as much user data, which can mitigate privacy concerns associated with collecting and storing user data.

Cons of the content-based approach:

On the other hand, the content-based approach can come with a few disadvantages, too. These can include:

The "Filter bubble":

Content filtering can recommend only content similar to the user's past preferences.

Limited serendipity:

Content-based systems may have limited capability to recommend items that are outside a user's known preferences.

This can lead to a lack of diversity and novelty in the recommendations, and prevent the user from discovering new and unexpected items

Collaborative filtering

Is a popular technique used to provide personalized recommendations to users based on the behavior and preferences of similar users

The fundamental idea behind collaborative filtering is that users who have interacted with items in similar ways or have had similar preferences in the past are likely to have similar preferences in the future, too.

Collaborative filtering relies on user community to generate recommendations.

There are two main types of collaborative filtering: memory-based and model-based.

Memory-based recommenders

Memory-based recommenders rely on the direct similarity between users or items to make recommendations.

Usually, these systems use raw, historical user interaction data, such as user-item ratings or purchase histories, to identify similarities between users or items and generate personalized recommendations.

Memory-based recommenders can be categorized into two main types user-based and item-based collaborative filtering.

A user-based collaborative filtering recommender system:

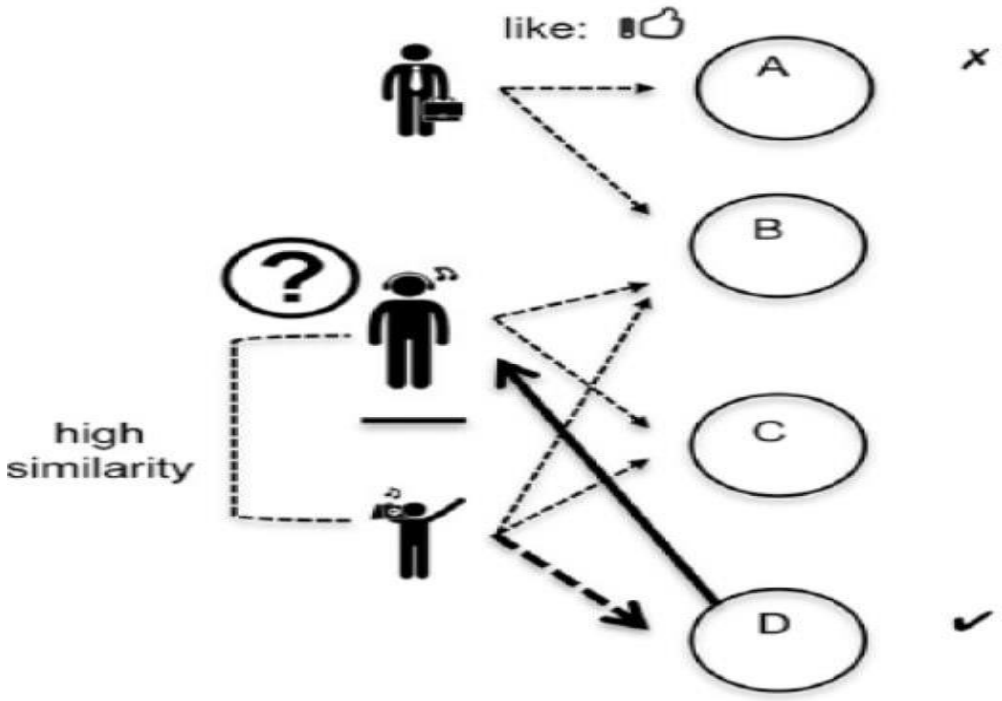


Figure 2/User-based collaborative

With the user-based approach, recommendations to the target user are made by identifying other users who have shown similar behavior or preferences. This translates to finding users who are most similar to the target user based on their historical interactions with items. This could be “users who are similar to you also liked…” type of recommendations.

An item-based collaborative filtering recommender system:

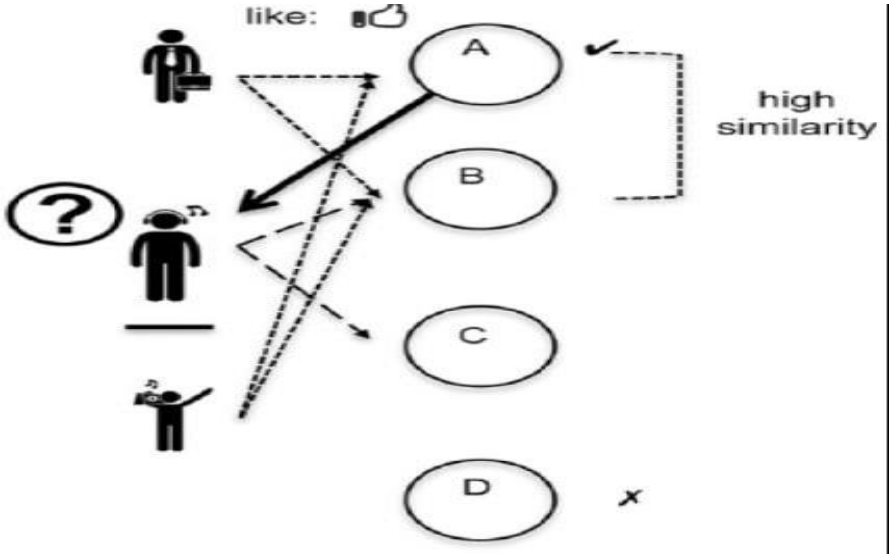


Figure 3/Item-based collaborative

In item-based collaborative filtering, recommendations are made by identifying items that are similar to the ones the target user has already interacted with.

The idea is to find items that share similar user interactions and recommend those items to the target user. This can include “users who liked this item also liked…” type of recommendations.

Main steps for user-based collaborative filtering:

Given a user Alice and an item i not yet rated by Alice

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Figure 4/user-based matrix

The goal is to estimate Alice's rating for this item i, by

..Find a set of users who liked the same items as Alice in the past and who have rated item I

Use, the average of their ratings to predict, if Alice will like item i

1. Calculate the similarity using person correlation formula

Pearson Correlation

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r_b})^2}}$$

- a, b: users
- $r_{a,p}$: rating of user a for item p
- P: set of items, rated both by a and b
- $\overline{r_a}$, $\overline{r_b}$: user's average ratings
- Possible similarity values are between -1 to 1

Figure 5/Person Correlation formula

After calculating the similarity between alice and other users we choose the most similar users to Alice,

Then we predict the rate of alice using the rates of these similar users using the next formula

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

Specifying a threshold theta, if the predicted rate > theta then we recommend this item to Alice

Model-based recommenders:

Model-based recommenders make use of machine learning models to generate recommendations

There are different types of model-based recommenders, such as matrix factorization, Singular Value Decomposition (SVD), or neural networks

Matrix factorization

Matrix factorization is a mathematical technique used to decompose a large matrix into the product of multiple smaller matrices.

matrix factorization is commonly employed to uncover latent patterns or features in user-item interaction data, allowing for personalized recommendations. Latent information can be reported by analyzing user behavior.

If there is feedback from the user, for example – they have view a particular product and have been given a rating, that can be represented in the form of a matrix. In this case,

- Rows represent users,
- Columns represent items, and
- The values in the matrix represent user-item interactions (e.g., ratings, purchase history, clicks, or binary preferences).

Since it's almost impossible for the user to rate every item, this matrix will have many unfilled values. This is called sparsity.

The matrix factorization process

Matrix factorization aims to approximate this interaction matrix by factorizing it into two or more lower-dimensional matrices:

- User latent factor matrix (U), which contains information about **users** and their relationships with latent factors.
- Item latent factor matrix (V), which contains information about **items** and their relationships with latent factors.

The matrix factorization process includes the following steps:

- Initialization of random user and item matrix,
- The ratings matrix is obtained by multiplying the user and the transposed item matrix,
- The goal of matrix factorization is to minimize the loss function (the difference in the ratings of the predicted and actual matrices must be minimal). Each rating can be described as a dot product of a row in the user matrix and a column in the item matrix.

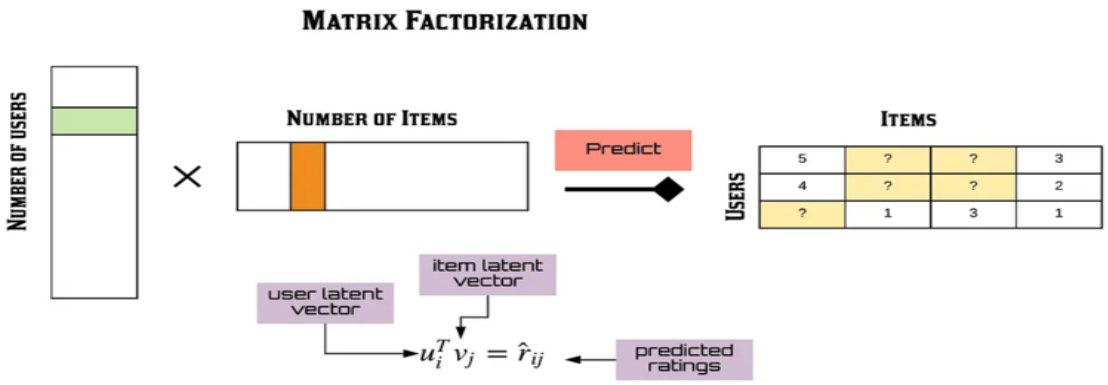


Figure 6/Matrix Factorization

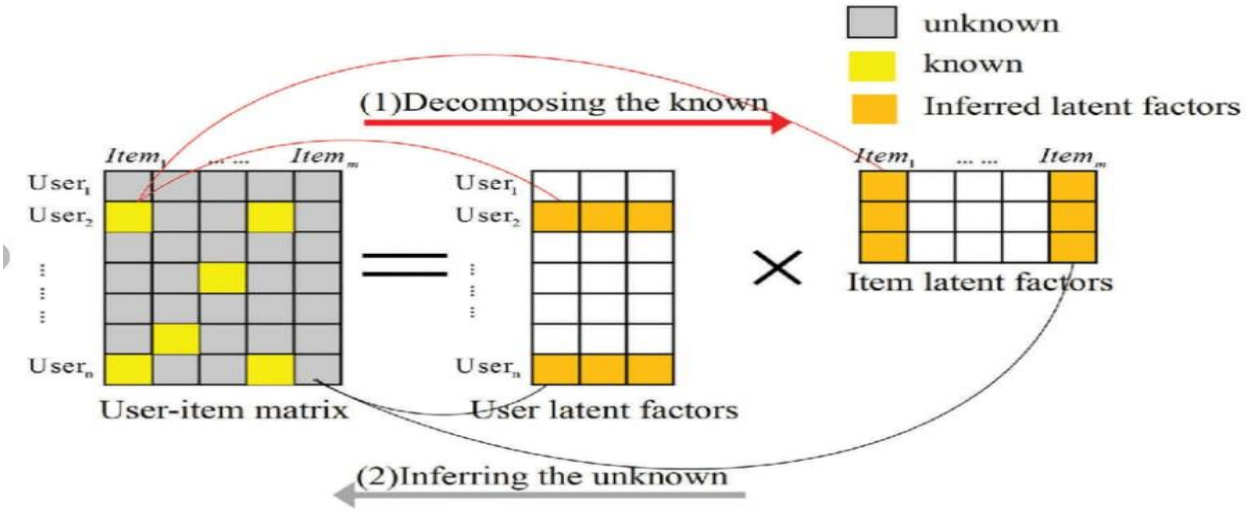


Figure 7/User latent factors

Pros of collaborative filtering

Effective personalization: Collaborative filtering is highly effective in providing personalized recommendations to users. It takes into account the behavior and preferences of similar users to suggest items that a particular user is likely to enjoy.

No need for item attributes: Collaborative filtering works solely based on user-item interactions, making it applicable to a wide range of recommendation scenarios where item features may be sparse or unavailable. This is especially useful in content-rich platforms

Serendipitous discoveries: Collaborative filtering can introduce users to items they might not have discovered otherwise. By analyzing user behaviors and identifying patterns across the user community, collaborative filtering can recommend items that align with a user's tastes but may not be immediately obvious to them.

Cons of collaborative filtering

The "cold-start" problem:

User cold start occurs when a new user joins the system without any prior interaction history. Collaborative filtering relies on historical interactions to make recommendations, so it can't provide personalized suggestions to new users who start with no data.

Item cold start happens when a new item is added, and there's no user interaction data for it. Collaborative filtering has difficulty recommending new items since it lacks information about how users have engaged with these items in the past.

2) **Sensitivity to sparse data:**

Collaborative filtering depends on having enough user-item interaction data to provide meaningful recommendations. In situations where data is sparse and users interact with only a small number of items, collaborative filtering may struggle to find useful patterns or similarities between users and items.

Potential for popularity bias:

Collaborative filtering tends to recommend popular items more frequently. This can lead to a "rich get richer" phenomenon, where already popular items receive even more attention, while niche or less-known items are overlooked.

HYBRID RECOMMENDERS:

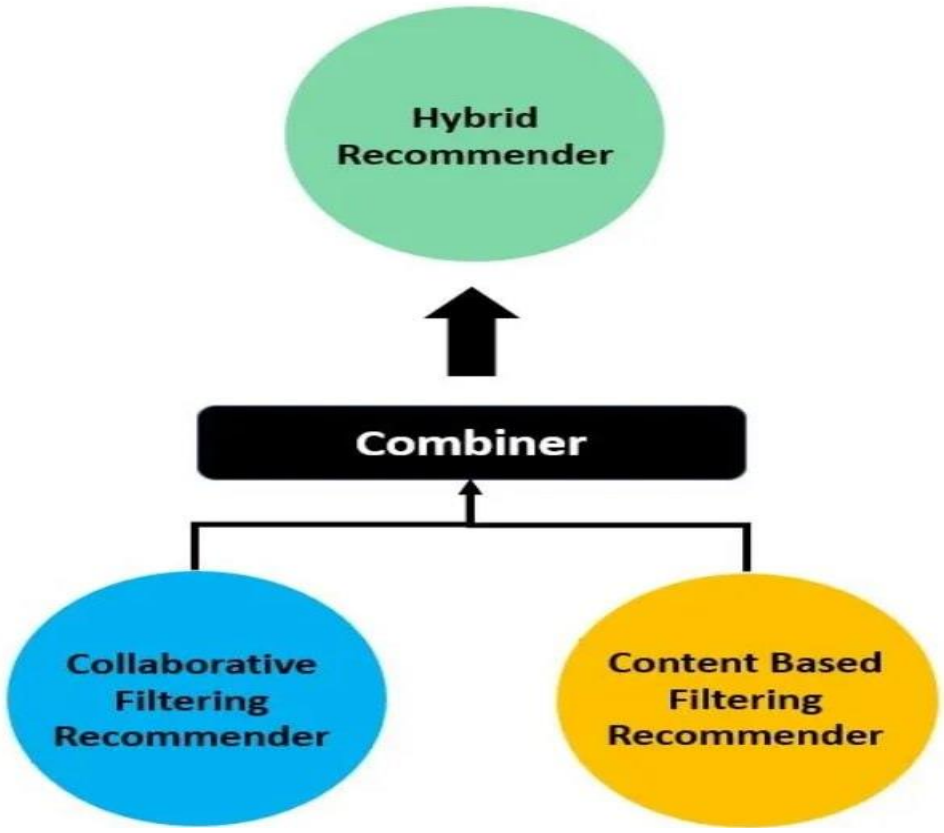


Figure 8/Hybrid Recommender

Hybrid recommendation systems combine multiple recommendation techniques or approaches to provide more accurate, diverse, and effective personalized recommendations.

They are particularly valuable in real-world recommendation scenarios because they can provide more robust, accurate, and adaptable recommendations.

The choice of which hybrid approach to use depends on the specific requirements and constraints of the recommendation system and the nature of the available data.

Pros of hybrid recommenders:

Some of the most common advantages of hybrid recommenders include:

- **Improved recommendation quality:** Hybrid recommenders leverage multiple recommendation techniques, combining their strengths to provide more accurate and diverse recommendations. This often results in higher recommendation quality compared to individual methods, benefiting users by offering more relevant suggestions.
- **Enhanced robustness and flexibility:** Hybrid models are often more robust in handling various recommendation scenarios. They can adapt to different data characteristics, user behaviors, and recommendation challenges. This flexibility is valuable in real-world recommendation systems.
- **Addressing common recommendation limitations:** Hybrid recommenders can mitigate the limitations of individual recommendation techniques. For example, they can overcome the "cold-start" problem for new users and items by incorporating content-based recommendations, providing serendipitous suggestions, and reducing popularity bias.

Cons of hybrid recommenders

Just like all other recommenders systems, hybrid recommenders have their downsides, too. Some include:

- **Increased complexity and development effort:** Implementing and maintaining hybrid recommendation systems can be more complex and resource-intensive. It requires expertise in multiple recommendation techniques and careful integration of these methods.
- **Data and computational demands:** Hybrid models often require more data and computational resources because they use multiple recommendation algorithms. This can be challenging, especially in large-scale systems with vast user-item interactions and a diverse catalog of items.
- **Tuning and parameter sensitivity**: Hybrid recommenders may involve a greater number of parameters and hyperparameters that need to be fine-tuned. Yet, ensuring optimal parameter settings for each recommendation component can be challenging and time-consuming.

While hybrid recommenders offer significant advantages in terms of recommendation quality and versatility, you should carefully consider the trade-offs and resource requirements when deciding which system to implement

After describing the pros and cons of each type of the recommendation system we decided to use the hybrid recommendation system.

Chapter 3: Sentimental Analysis

3.1 What is sentiment Analysis?

Sentiment analysis is a form of text research that uses a mix of statistics, natural language processing (NLP), and machine learning to identify and extract subjective information — for instance, a reviewer's feelings, thoughts, judgments, or assessments about a particular topic, event, or a company and its activities.

This analysis type is also known as opinion mining (with a focus on extraction) or affective rating. Some specialists prefer the terms sentiment classification and extraction. Regardless of the name, the goal is the same: to know a user or audience opinion on a target object by analyzing a vast amount of text from various sources.

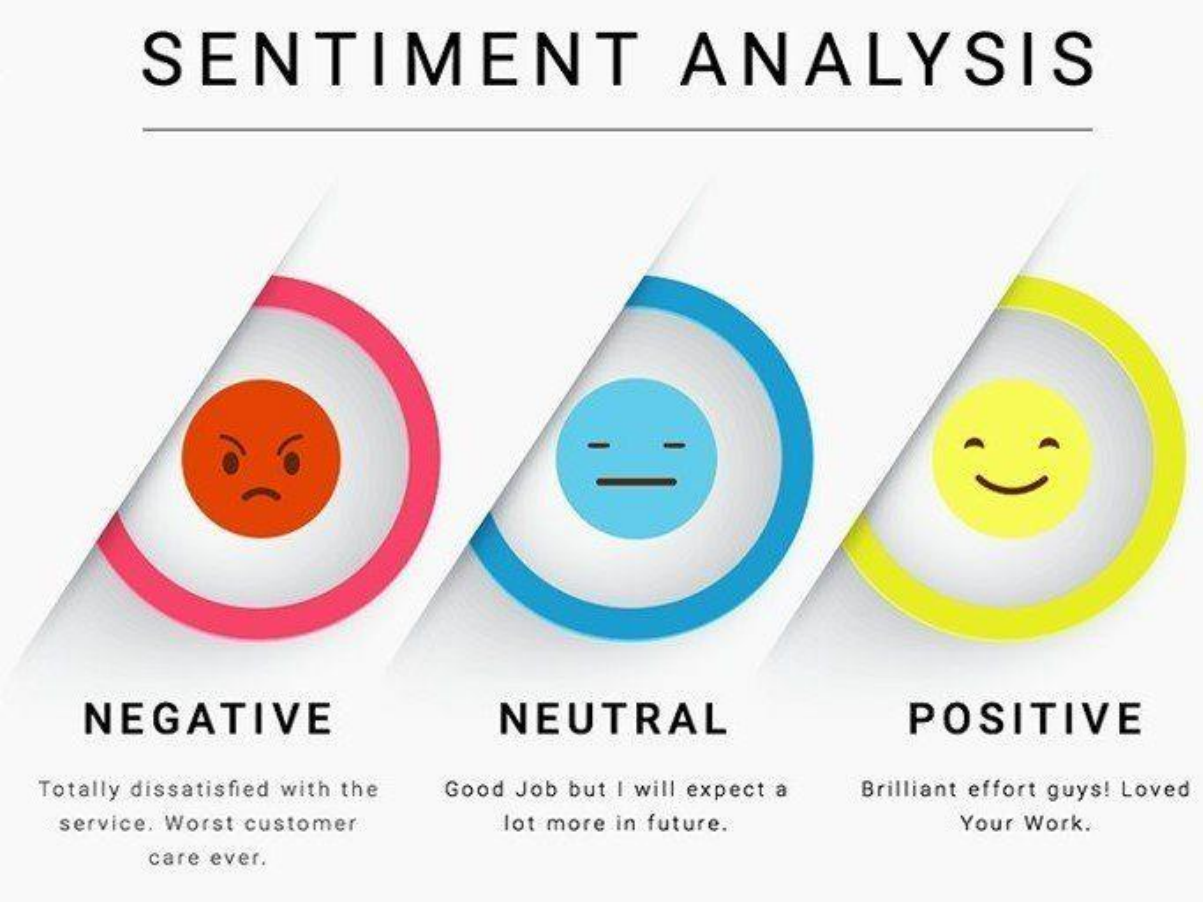


Figure 9/Sentiment Analysis

Basically, sentiment analysis distinguishes three types of emotions — negative, neutral, and positive. It can be applied to a separate sentence or its part as well as being used for document classification, where the term document covers a broad range of textual items like emails, reviews, comments, articles, and more.

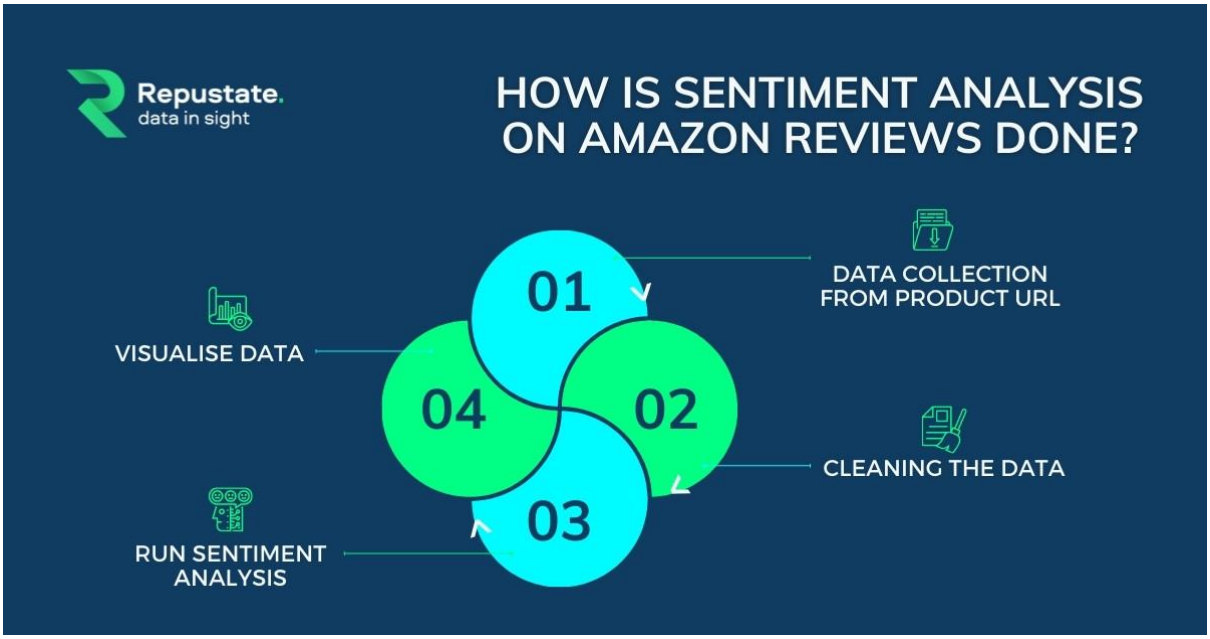


Figure 10/Amazon reviews

3.1 Types of Sentiment Analysis

- Fine-grained sentiment analysis
- Aspect-level sentiment analysis

What is the Fine-grained Sentiment Analysis?

Fine-grained sentiment analysis (FGSA) is a type of sentiment analysis that goes beyond simply classifying text as positive, negative, or neutral. It aims to provide a more detailed understanding of the sentiment expressed in a text by analyzing the sentiment towards specific aspects or entities within the text.

Here are some key characteristics of FGSA:

Objectives:

- Identify the sentiment towards specific aspects or entities (e.g., product features, service quality, customer service interactions)
- Capture the degree or intensity of sentiment (e.g., not just positive, but very positive)
- Analyze mixed sentiment within a single piece of text

Benefits:

- Provides more actionable insights for businesses and organizations
- Helps identify customer pain points and areas for improvement
- Enables personalized customer service interactions
- Improves product development and marketing campaigns

Techniques:

- Deep learning: Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) are commonly used for FGSA due to their ability to learn complex relationships between words and context.
- Rule-based approaches: Define rules based on linguistic features like n-grams and negation cues to identify sentiment towards specific aspects.
- Lexicon-based approaches: Use sentiment lexicons containing words associated with positive, negative, and neutral sentiment to analyze aspect-specific sentiment.

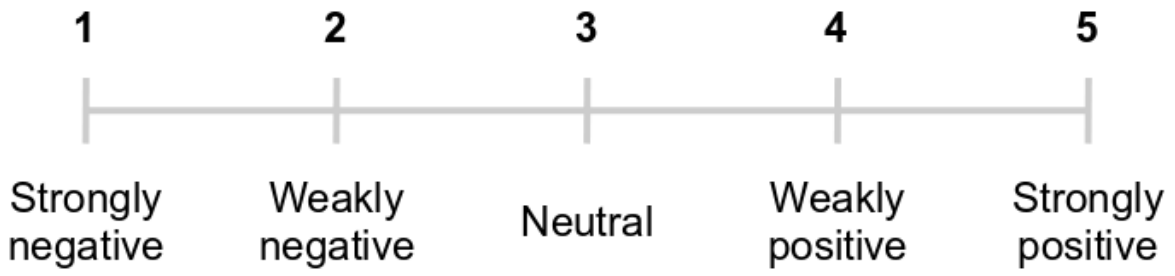
Applications:

- Analyzing customer reviews and feedback: Identify areas for improvement and understand customer satisfaction.
- Monitoring social media sentiment: Track public opinion and brand reputation.
- Developing chatbots and virtual assistants: Respond in a way that aligns with the user's sentiment and intent.
- Personalizing content and recommendations: Tailor content to individual users based on their sentiment towards specific aspects.

Challenges:

- Data annotation: Requires labeled data with sentiment annotations for specific aspects, which can be expensive and time-consuming.
- Context dependency: Sentiment towards an aspect can be influenced by the overall context of the text.
- Ambiguity and sarcasm: Identifying the true sentiment can be challenging when dealing with ambiguous statements or sarcasm.

Overall, FGSA is a powerful tool for extracting valuable insights from text data and gaining a deeper understanding of the sentiment expressed towards specific aspects. As technology advances, we can expect FGSA to become increasingly accurate and sophisticated, providing even richer insights for various applications.



Data preprocessing

During data preprocessing, raw data is converted to the required format. The dataset is cleaned up by removing all useless data, followed by the undesired qualities.

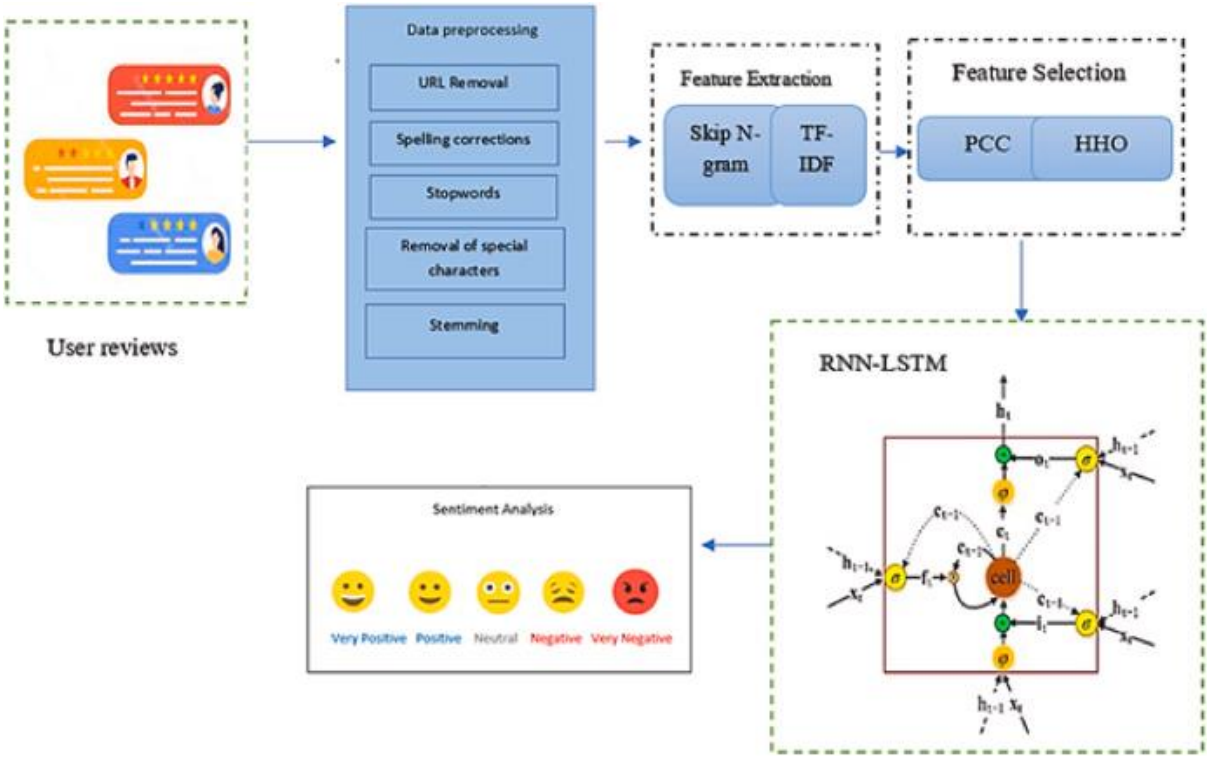


Figure 11/Data processing

3.2 Fine-grained or Aspect-level

Type of sentiment	Fine-grained	Aspect-Level
Strengths	○ Provides a more nuanced understanding of sentiment within a text, going beyond overall positivity/negativity. ○ Can identify emotional sub-categories (e.g., frustration, joy, sarcasm) in addition to simple sentiment. ○ Useful for tasks like understanding user experience or analyzing product reviews.	○ Analyzes sentiment towards specific aspects or features of a product, service, or topic. ○ Provides deeper insights for business or product improvement. ○ Helps understand user opinions on different facets of something.
Weaknesses	○ May be more computationally expensive and challenging to implement. ○ Requires more annotated training data. ○ Doesn't always provide explicitly targeted sentiment towards specific aspects	○ Requires identifying and extracting targeted aspects, which can be complex. ○ May not comprehensively capture general sentiment beyond specific aspects. ○ Often requires specific datasets focused on aspect-based opinions.

3.3 Why Aspect-level sentiment analysis?

Traditional sentiment analysis tells you if a customer review is positive or negative, but it leaves you in the dark about why. Diving deeper with aspect-level sentiment analysis (ALSA) unlocks a treasure trove of insights that traditional analysis simply can't provide. Here's how ALSA can revolutionize your understanding of customer reviews:

Pinpoint strengths and weaknesses: ALSA breaks down reviews into specific aspects like battery life, camera quality, or customer service. This lets you see exactly what customers love and where they're frustrated. Imagine knowing "great battery life, but confusing menus" is a recurring theme – a gold mine for product improvement!

You make customers happy: You can address their concerns, leading to better customer service and loyal fans.

Overall, aspect-level analysis helps you understand your customers better, which makes your business stronger. It's like having a superpower to see what motivates them!

3.4 Sentiment Analysis Scope:

SA, as a classification problem, has three different levels:

1. Document-Level
2. Sentence-Level
3. Aspect-Level

- Document-Level:

  Document-Level is about classifying the whole opinion document into sentiments or positive or negative

- Sentence-Level

  Sentence-Level is about classifying sentiment expressed in each sentence

- Aspect-Level

  Aspect-Level plans to characterize the opinion regarding the particular aspects.

3.5 Text Processes

- Tokenization:

  breaks up text into small chunks called tokens.

  • Text: Analyzing text is not that hard

  • Tokens = ["Analyzing", "text", • POS: Definite article, noun,

  "is", "not", "that", "hard", "."]

- Stopword Removal

  Takes out common words which are not useful for sentiment analysis. They provide no meaning.

  Example: a, and, or, the, etc.

- Lemmatization

To transform words back to their root form.

List of words: going, gone, went

Lemma: go

Chapter 4: System Analysis and System Design

In this chapter the reader will get advanced knowledge about the Analysis of the project with lots of details our system support both functional and nonfunctional requirements:

4.1 System Development Life Cycle:

An effective system development life Cycle (SDLC) need to result in an excessive nice system that meets consumer expectations, reaches of completion within time and value reviews, and works successfully and efficiently in the current and planned information technology infrastructure.

System Development Life Cycle (SDLC) is a conceptual version which includes rules and methods for developing or changing systems throughout their life cycles.

SDLC includes the following activities:

1.Plan: This phase involves defining the scope, objectives, and expectations of the project with the stakeholders and customers.

2.Design: In this phase, a high-level design of the software solution is created based on the requirements .

3.Development: This phase involves developing and implementing the software solution in short cycles called sprints or iterations.

4.Test: This phase involves verifying and validating that the software solution meets the quality standards and specifications defined in the design phase.

5.Deploy: This phase involves releasing the software solution to the end-users or customers for their use.

6.Review: This is the final phase of the Agile methodology, often referred to as the “Retrospective” phase.

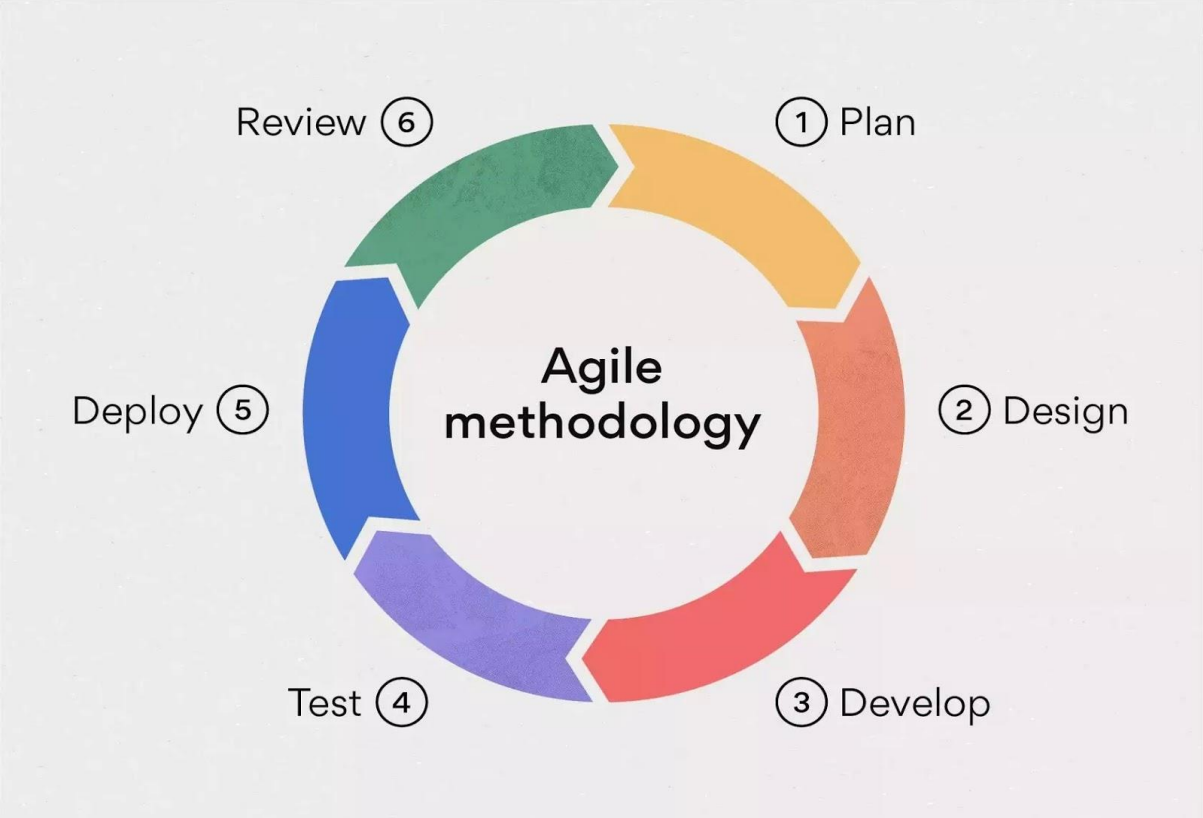


Figure 12/Agile methodology

Why we will use Agile Methodology?

- **Agile software methodologies deliver superior quality products**: By testing the software continuously and incorporating feedback from customers and stakeholders, agile teams ensure that the software meets the quality standards and specifications.
- **Agile software methodologies increase customer satisfaction**: By involving customers in the decision-making process and delivering value early and often, agile teams create products that meet or exceed customer expectations.
- **Agile software methodologies improve team collaboration**: By breaking down the project into smaller and manageable units, agile teams foster a culture of teamwork, communication, and innovation among team members.
- **Agile software methodologies enhance risk management**: By delivering working software frequently and validating it with customers, agile teams reduce the risk of delivering a product that does not meet customer requirements or expectations.

4.2 System Analysis Stages:

Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with input from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved by the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

Functional requirements

- System must be sufficient to the cause for which it is designed.

- System must be able in assessment process.

- The system should be able to check whether the performer acts the practice at the correct manner or not.

Functionality	Description
User Registration and Authentication	○ Users should be able to create accounts and log in securely. ○ User roles and permissions should be defined to control access to different features.
Product Catalog	○ The system should provide a catalog of products with relevant information such as name, description, price, and ratings. ○ Products should be categorized or tagged to facilitate searching and recommendation.
Recommendation System	○ The system should analyze user preferences and behavior to provide personalized product recommendations. ○ Recommendations should be based on factors such as previous purchases, ratings, and browsing history.

Sentiment Analysis	○ The system should analyze user reviews and feedback to determine sentiment (positive, negative, neutral) towards products. ○ Sentiment analysis should help in generating insights about product performance and customer satisfaction.
Search Functionality	○ Users should be able to search for products based on keywords, categories, or specific criteria. ○ The search results should be relevant and displayed in a user-friendly manner.
User Feedback	○ Users should have the ability to provide ratings, reviews, and feedback for products. ○ The system should allow users to view and interact with other user feedback.

Non-functional requirements:

Non-Functionality	Description
Performance	○ The system should respond promptly to user actions, providing quick recommendations and analysis results. ○ The response time should be within acceptable limits even during peak usage.
Scalability	○ The system should be able to handle a growing number of users and products without significant performance degradation. ○ It should be scalable to accommodate increased data and user load.
Security	○ User data, including personal information and preferences, should be securely stored and protected. ○ The system should implement appropriate authentication and authorization mechanisms to prevent unauthorized access.

Usability	○ The web app should have an intuitive user interface that is easy to navigate and understand. ○ The design should consider accessibility guidelines to ensure inclusivity for users with disabilities.
Reliability	○ The system should be reliable and available for users at all times, minimizing downtime and disruptions. ○ It should have appropriate backup and recovery mechanisms to handle data loss or system failures.
Integration	○ The web app should be able to integrate with external systems or APIs to retrieve product information, reviews, or other relevant data. ○ Integration with social media platforms for sharing recommendations or feedback can be considered.

Stage 3: Designing the Architecture

SRS is the reference for architects to come up with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters such as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third-party modules. The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Developing

In this stage of SDLC the actual development starts, and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

We must follow the coding programming tools like compilers, interpreters, debuggers, etc. we will generate the code in different high level programming languages such Python, React Native (JS), and Dart (Flutter).

Stage 5: Testing

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the application where application defects are reported, tracked, fixed, and retested, until the application reaches the quality standards defined in the SRS.

Black Box Testing

Is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration system and acceptance.

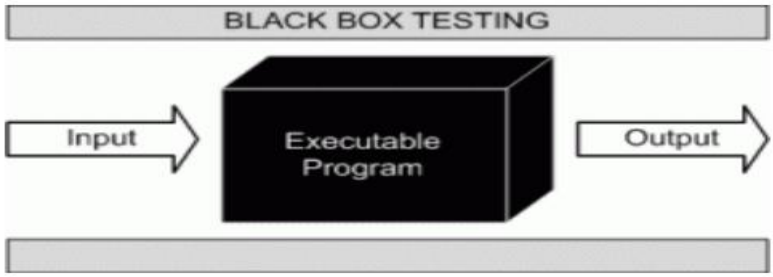


Figure 13/Black Box Testing

Example:

A simple login screen of software or a web application. The login screen has two fields, username and password as an input and the output will be to enable access to the system.

And it will test the valid username and password to login to the right account.

• A user logs in when inputs a present username and correct password.

• A user receives an error message when entering username and incorrect password.

For example, a user might enter the password in the wrong format, and a user might not receive an error message on entering an incorrect password.

There are many Types of Black Box Testing, but the following are the prominent ones.

• Functional testing

– This black box testing type is related to the functional requirements of a system.

• Non-functional testing

– Test non-functional requirements such as performance, scalability, usability.

• Regression testing

– Is done after code fixes, upgrades, or any other system maintenance to check the new code has not affected the existing code. Tools used for Black Box Testing:

• For Functional/ Regression Tests you can use

– QTP, Selenium • For Non-Functional Tests, you can use – LoadRunner, JMeter Why black Box?

• The tester doesn't need any technical knowledge to test the system. It is essential to understand the user's perspective.

• Code access is not required.

• Well suited and efficient for large code segment

• Testing is performed after development, and both the activities are independent of each other.

• Black box testing methodology is close to agile.

4.3 Software Failure (Risks):

Software failure risk is concerned only with faults that can produce failure. Software can fail because of incomplete or incorrect requirements analysis, poor design, and inadequate testing or quality assurance. Faults can be introduced at any time prior to the use of the software.

- Changing requirements.
- Inaccurate estimates of needed resources.
- Badly defined system requirements.
- Poor reporting of the project's status.
- Lack of resources.
- Schedule Risk
- Performance quality
- Timing

4.4 UML Diagrams:

There are several types of UML diagrams and each one of them serves a different purpose regardless of whether it is being designed before the implementation or after.

The two broadest categories that encompass all other types are Behavioral UML diagram and

Structural UML diagram. As the name suggests, some UML diagrams try to analyze and depict the structure of a system or process, whereas others describe the behavior of the system, its actors, and its building components. The different types are broken down as **follows:**

Behavioral UML Diagram:

- Activity Diagram
- Use Case Diagram
- Interaction Overview Diagram
- Timing Diagram
- State Machine Diagram
- Communication Diagram
- Sequence Diagram

Structural UML Diagram

- Class Diagram
- Object Diagram
- Component Diagram
- Composite Structure Diagram
- Deployment Diagram
- Package Diagram
- Profile Diagram

4.4.1 Use Case Diagram

A use case describes how a user uses a system to accomplish a particular goal. A use case diagram consists of the system, the related use cases and actors and relates these to each other to visualize: what is being described? (system), who is using the system? (actors) and what do the actors want to achieve? (Use cases), thus, use cases help ensure that the correct system is developed by capturing the requirements from the user's point of view.

What is a Use Case Diagram in UML?

A use case is a list of actions or event steps typically defining the interactions between a role of an actor and a system to achieve a goal. A use case is a useful technique for identifying, clarifying, and organizing system requirements. A use case is made up of a set of possible sequences of interactions between systems and users that define the features to be implemented and the resolution of any errors that may be encountered.

While a use case itself might drill into a lot of detail (such as, flow of events and scenarios) about every possibility, a use-case diagram can help provide a higher-level view of the system, providing the simplified and graphical representation of what the system must do.

A use case (or set of use cases) has these characteristics:

1. Organizes functional requirements

2. Models the goals of system/actor (user) interactions

3. Describes one main flow of events (main scenarios) and possibly other exceptional flows (alternatives), also called paths or user scenarios

➢ Use Case Diagram Notations

Actor

Actors are usually individuals involved with the system defined according to their roles. The actor can be a human or other external system.

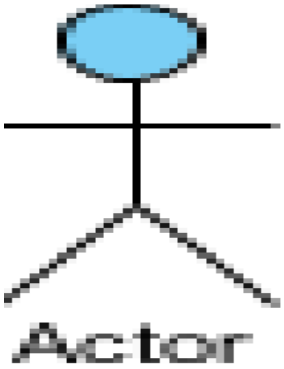


Figure 14/ usecase actor

Use Case:

A use case describes how actors use a system to accomplish a particular goal. Use cases are typically initiated by a user to fulfill goals describing the activities and variants involved in attaining the goal.

Relationship

The relationships between and among the actors and the use cases.

<<Extend>> Use Case

An extending use case is, effectively, an alternate course of the base use case. The <> use case accomplishes this by conceptually inserting additional action sequences into the base use-case sequence.

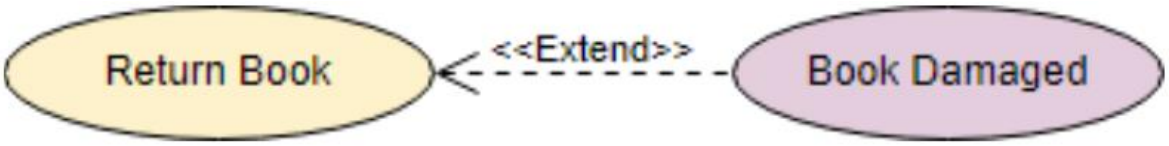


Figure 15/use case extend

<<Include>> Use Case

The time to use the <<include>> relationship is after you have completed the first cut description of all your main Use Cases. You can now look at the Use Cases and identify common sequences of user-system interaction.

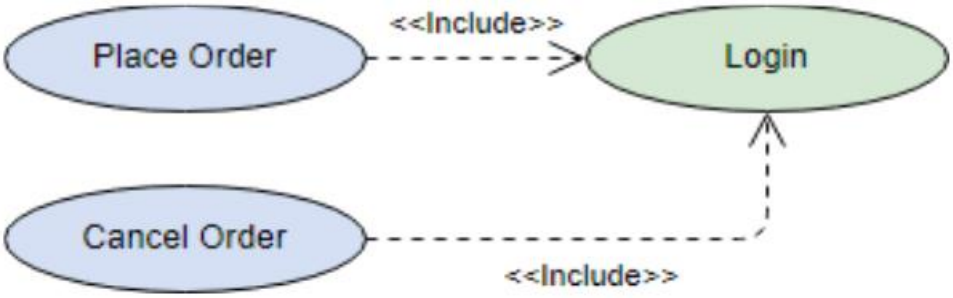


Figure 16/Use case include

Association

Actor and use case can be associated to indicate that the actor participates in that use case. Therefore, an association corresponds to a sequence of actions between the actor and use case in achieving the use case.

Figure 17/use case Association

UML System

The scope of a system can be represented by a system (shape), or sometimes known as a system boundary. The use cases of the system are placed inside the system shape, while the actors who interact with the system are put outside the system. The use cases in the system make up the total requirements of the system

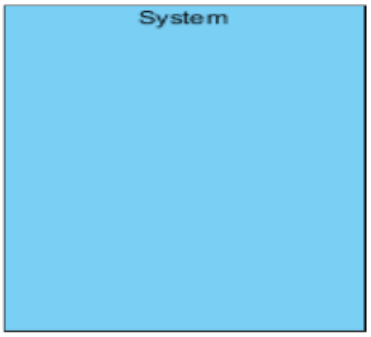


Figure 18/UML System

Dependency

A dependency relationship represents that a model element relies on another model element for specification and/or implementation.

Figure 19/ Use case Dependency

Generalization

A generalization relationship is used to represent inheritance relationship between model elements of same type. The more specific model element shares the same specification with the more general the model element but carries more details in extra.

Figure 20/Use cse Generalization

Realization

A realization is a relationship between a specification and its implementation

Figure 21/Use case Realization

Use Case Diagram

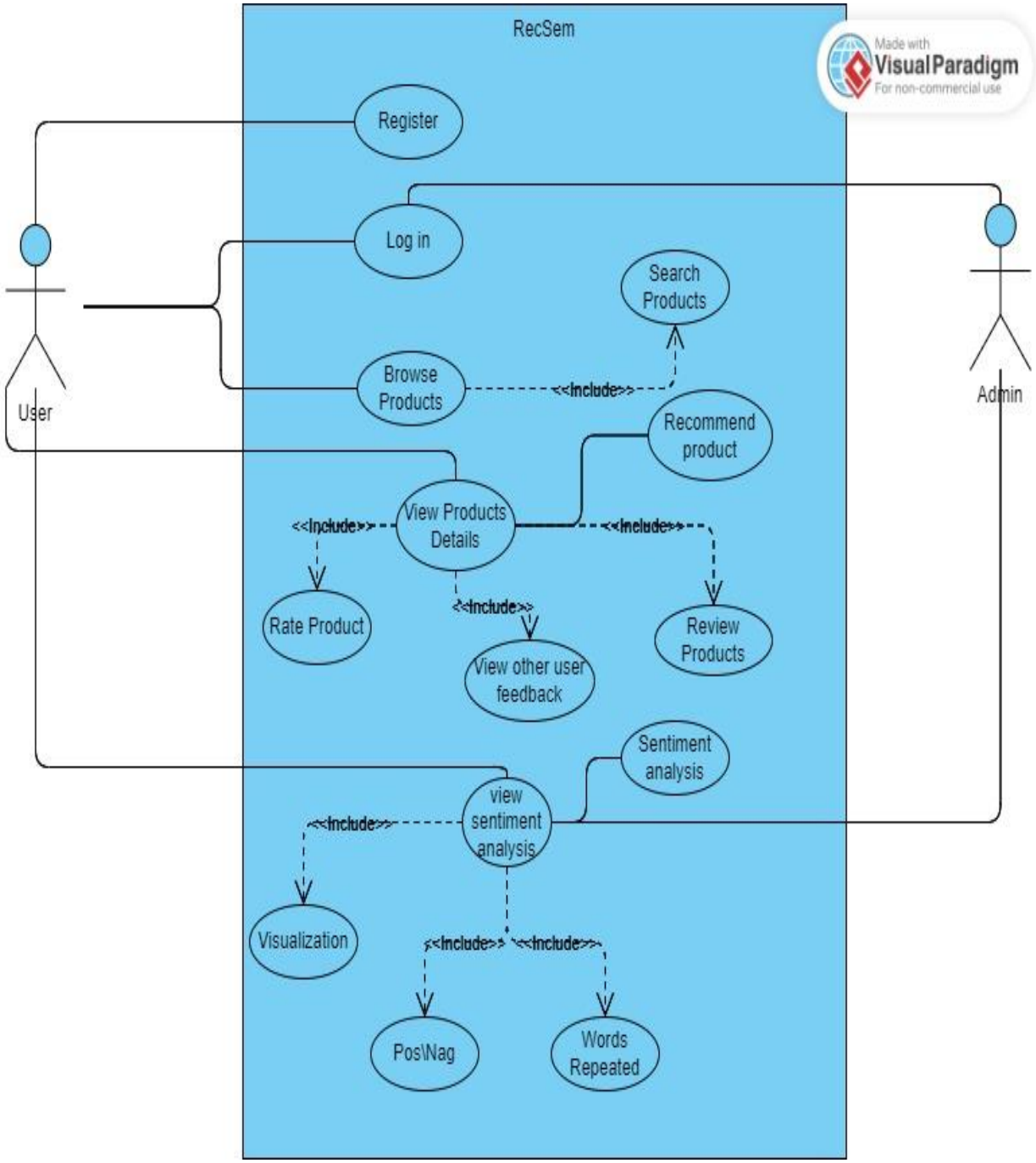


Figure 22/Use Case Diagram

4.4.2 Data Flow Diagram

A Data Flow Diagram (DFD):

It is a graphical representation that depicts the flow of data within a system. It illustrates how data is input, processed, and output in a system, showing the interactions between different components or processes.

Its Types:

1. Level 0
2. Level 1

Level 0:

The Level 0 DFD typically consists of a single process symbol representing the entire system, external entities (sources or destinations of data), and the data flows between them. It focuses on illustrating the overall scope and boundaries of the system, without delving into the internal processes or data stores.

Level 1:

The Level 1 DFD shows the main processes as separate symbols, along with the data flows between them. It may also include data stores that hold data temporarily or persistently.

Overall, the Level 0 DFD provides a high-level overview of the system's scope and external interactions, while the Level 1 DFD delves into the internal processes and data flows within the system. The Level 1 DFD is a more detailed breakdown of the Level 0 DFD

Processes:

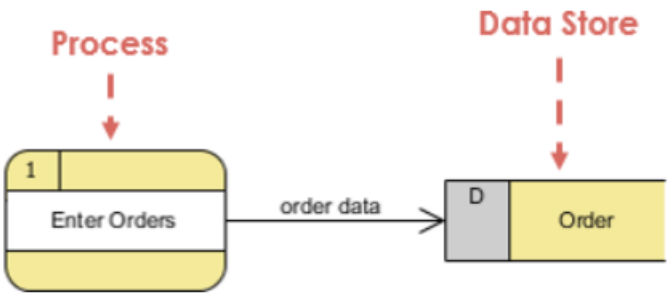
Processes represent the activities or functions that transform input data into output data. Each process is depicted as a circle or rectangle with a label describing the function it performs.

Figure 23/DFD Process

Data Flows:

Data flows represent the movement of data between processes, external entities, and data stores. They are depicted as arrows and show the direction of data flow.

Figure 24/DFD Data Flow



External Entities:

External entities are sources or destinations of data that interact with the system but are not part of it. They can be users, external systems, or organizations.

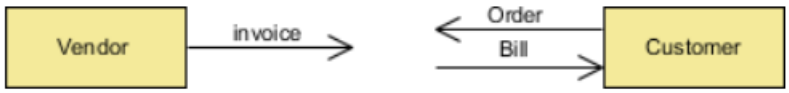


Figure 25/DFD External Entities

Data Stores:

Data stores represent the repositories where data is stored within the system. They are depicted as rectangles with labels.

Figure 26/DFD Data Store

Overall, DFDs are used to analyze and model systems at a high level, providing a clear and concise representation of the data flow and interactions between different components. They are commonly used during the requirements gathering and system design phases of software development to understand the system's data processing and identify potential issues or improvements.

DFD Level 0

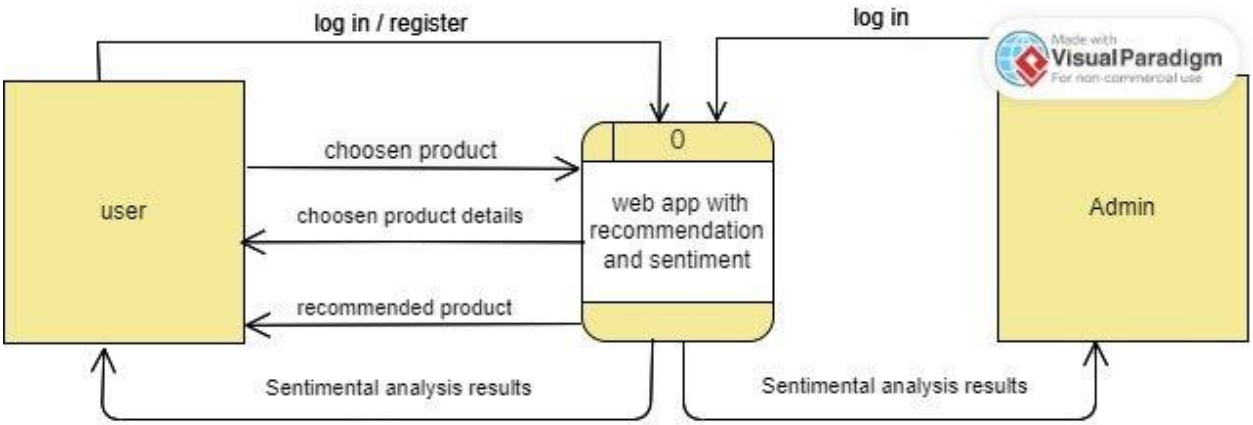


Figure 27/DFD Level 0

DFD Level 1

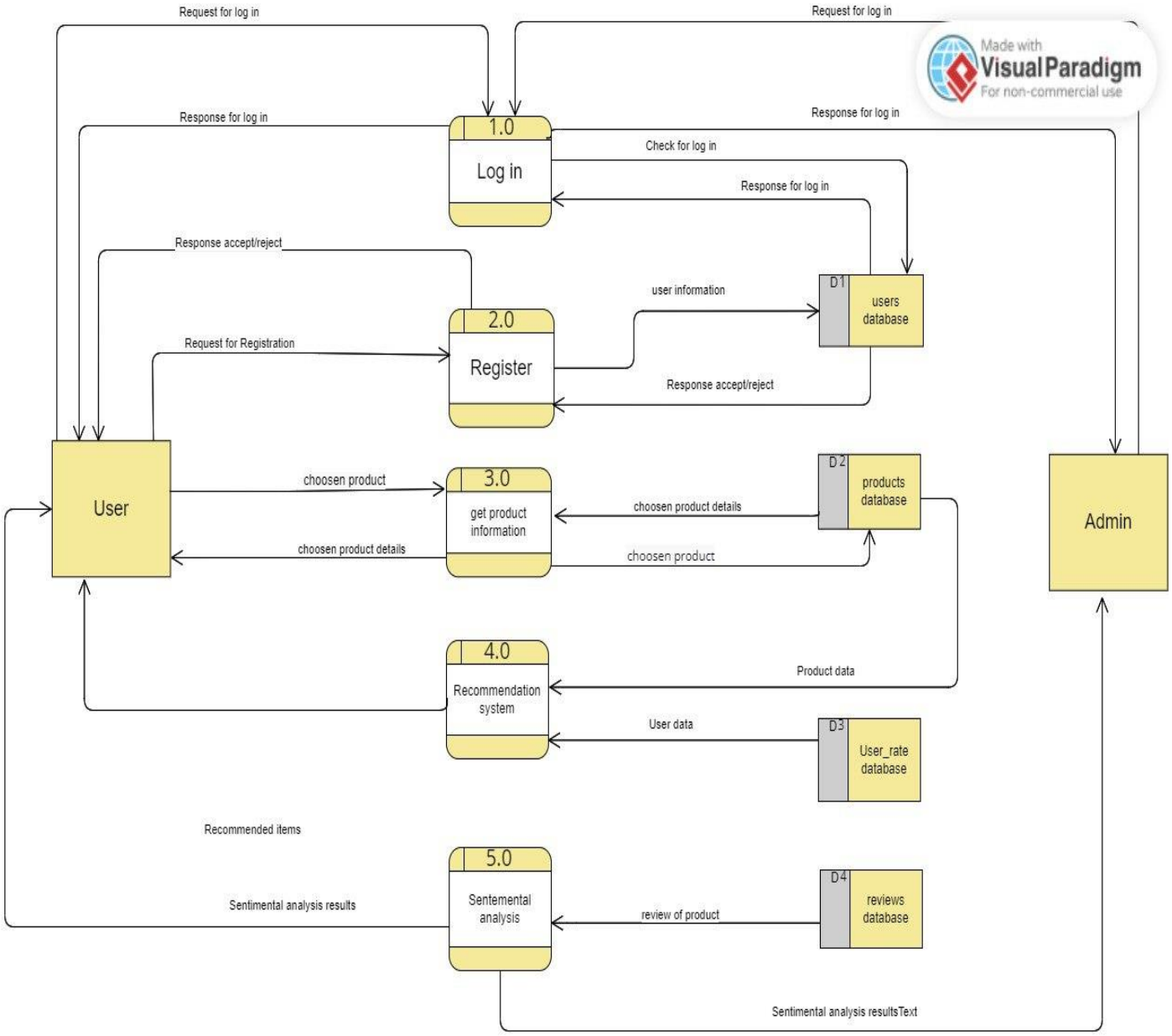


Figure 28/DFD Level 1

4.4.3 Sequence Diagram

What is a sequence diagram?

A sequence diagram is a type of interaction diagram that details how operations are carried out between objects in a system.

It shows the order of messages, time, and actors involved in the collaboration.

The Level of Used Sequence Diagram:

System Sequence Diagrams (SSDs): These are high-level diagrams that provide an overview of the interactions between the system and external entities (like a user or another system). They are less detailed and are often used in the early stages of development.

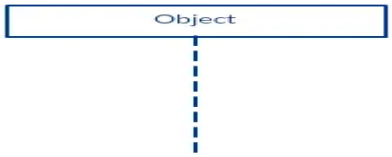
Why is sequence Diagram Important?

- **Message Flow**: They represent the flow of messages from one object to another, making it easier to understand the communication between different components of a system.
- **Adaptability**: They can be easily updated according to changes within a system.
- **Visualizing Interactions**: They capture and visualize the nature and order of collaboration between objects over time. This visualization can help in understanding complex systems.
- **Design and Documentation**: They are commonly used during the design and documentation phases of software development.

Elements of Sequence Diagram:

Object:

A sequence diagram is structured in such a way that it represents a timeline that begins at the top and descends gradually to mark the sequence of interactions. Each object has a column and the messages exchanged between them are represented by arrows.



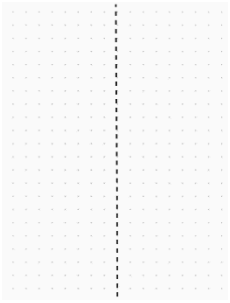
t

Figure 29/sequence object

Lifelines:

A sequence diagram is made up of several of these lifeline notations that should be arranged horizontally across the top of the diagram. No two lifeline notations should overlap each other. They represent the different objects or parts that interact with each other in the system during the sequence.

Figure 30/Sequnce Lifeline



Actors:

A lifeline notation with an actor element symbol is used when the particular sequence diagram is owned by a use case.

Figure 31/sequence Actor

Entity:

A lifeline with an entity element represents system

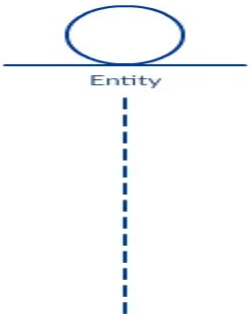


Figure 32/sequence Entity

The activation bar:

It is the box placed on the lifeline. It is used to indicate that an object is active (or instantiated) during an interaction between two objects. The length of the rectangle indicates the duration of the objects staying active

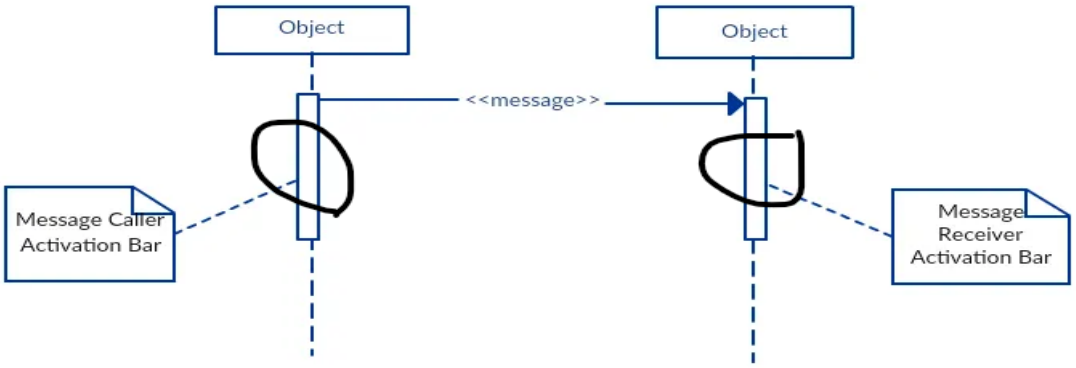


Figure 33/sequence Activation bar

Synchronous message:

As shown in the activation bars example, a synchronous message is used when the sender waits for the receiver to process the message and return before carrying on with another message. The arrowhead used to indicate this type of message is a solid one, like the one below.

Figure 34/sequence Synchronous message

Alternatives:

The alternative combination fragment is used when a choice needs to be made between two or more message sequences. It models the “if then else” logic.

The alternative fragment is represented by a large rectangle or a frame; it is specified by mentioning ‘alt’ inside the frame’s name box.

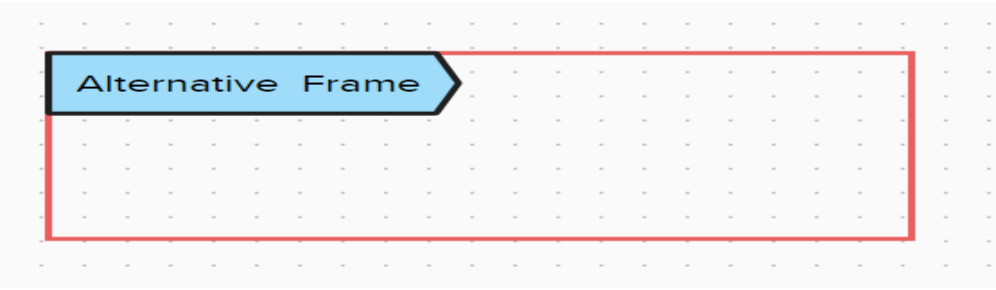


Figure 35/sequence Alternative

Sequences Diagram

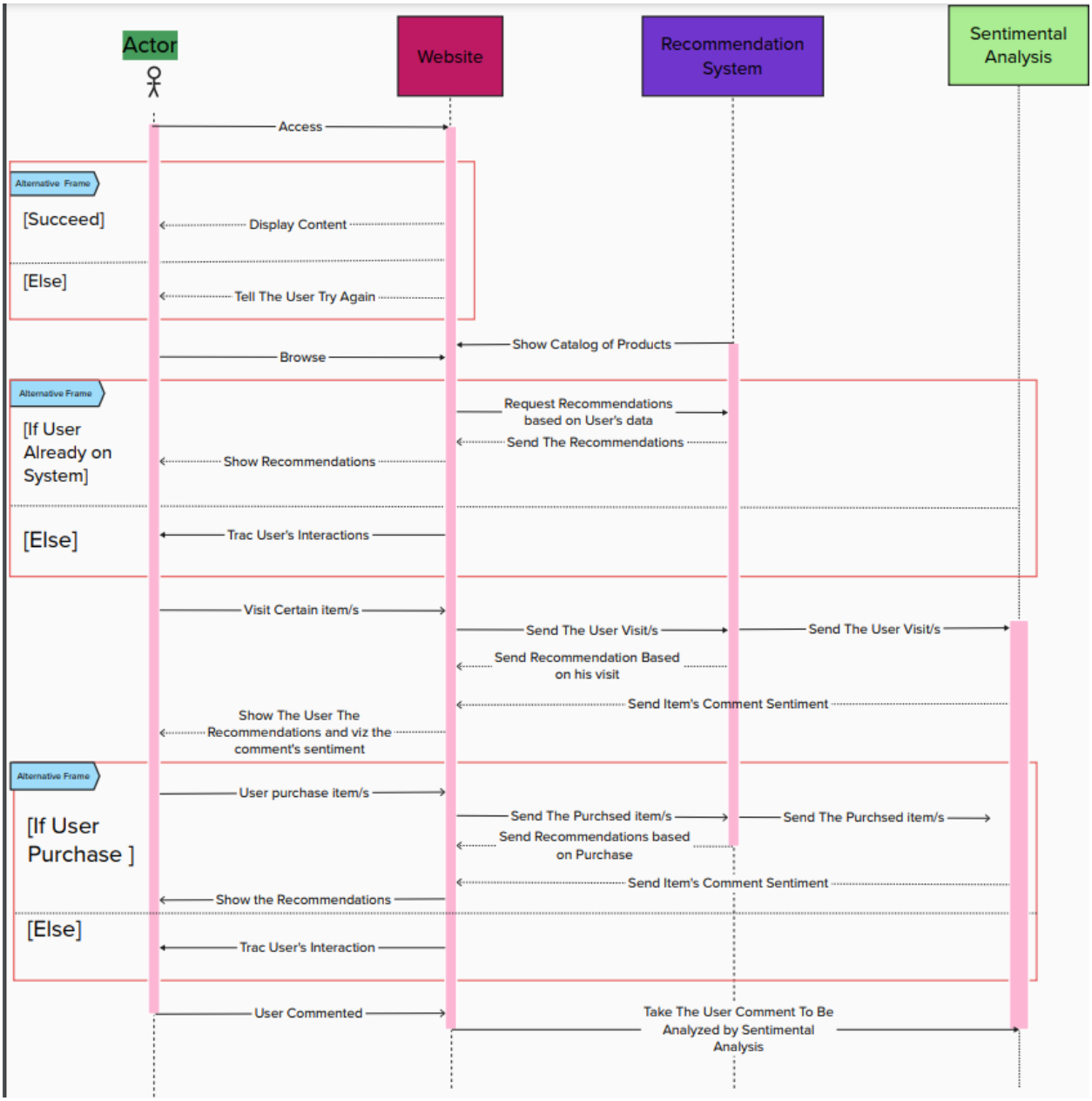


Figure 36/sequence Diagram

4.4.4 ERD Model

ERD ER-modeling is a data modeling technique used in software Engineering to produce a conceptual data model of an information system. Diagrams created using this ER-modeling technique are called Entity-Relationship Diagrams, or ER diagrams or ERDs. Relations are defined between tables for cross referencing. The way it stores data makes users easy to understand the structure and content of the data. Developers may use Structured Query Language (SQL) to query data and add indexes to database for faster querying; making relational database performs well even when the amount of data increases over time. Therefore, despite being challenged by object databases for years, relational database remains to be the most prevalent way of storing enterprise data to this date. Oracle, Microsoft SQL Server, MySQL, and PostgreSQL are some of the popular relational database management systems.

ER diagram (ERD):

First, what is an Entity Relationship Diagram? Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. An ERD contains different symbols and connectors that visualize two important information: The major entities within the system scope, and the inter-relationships among these entities. And that's why it's called "Entity" "Relationship" diagram (ERD)! When we talk about entities in ERD, very often we are referring to business objects such as people/roles (e.g., Student), tangible business objects (e.g., Product), intangible business objects (e.g., Log), etc. "Relationship" is about how these entities relate to each other within the system.

In a typical ER design, you can find symbols such as rounded rectangles and connectors (with different styles of their ends) that depict the entities, their attributes, and inter-relationships.

When to draw ER Diagrams?

So, when do we draw ERDs? While ER models are mostly developed for designing relational databases in terms of concept visualization and in terms of physical database design, there are still other situations when ER diagrams can help. Here are some typical use cases.

Database design

- Depending on the scale of change, it can be risky to alter a database structure directly in a DBMS. To avoid ruining the data in a production database, it is important to plan out the changes carefully. ERD is a tool that helps. By drawing ER diagrams to visualize database design ideas, you have a chance to identify the mistakes and design flaws, and to make corrections before executing the changes in the database.

Database debugging - Debugging database issues can be challenging, especially when the database contains many tables, which require writing complex SQL in getting

the information you need. By visualizing a database schema with an ERD, you have a full picture of the entire database schema. You can easily locate entities, view their attributes, and identify the relationships they have with others. All these allow you to analyze an existing database and to reveal database problems more easily.

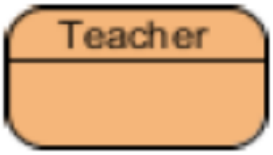
Database creation and patching - Visual Paradigm, an ERD tool, supports a database generation tool that can automate the database creation and patching process by means of ER diagrams. So, with this ER Diagram tool, your ER design is no longer just a static diagram but a mirror that truly reflects the physical database structure.

An ER Diagram contains entities, attributes, and relationships. In this section, we will go through the ERD symbols in detail.

1. Entity:

   An ERD entity is a definable thing or concept within a system, such as a person/role (e.g., Student), object (e.g., Invoice), concept (e.g., Profile) or event (e.g., Transaction) (note: In ERD, the term "entity" is often used instead of "table", but they are the same). When determining entities, think of them as nouns. In ER models, an entity is shown as a rounded rectangle, with its name on top and its attributes listed in the body of the entity shape. The ERD example below shows an example of an ER entity.

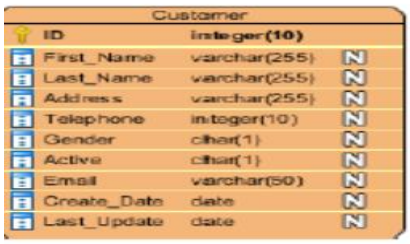
Figure 37/ERD Entity



2. Entity Attributes:

Also known as a column, an attribute is a property or characteristic of the entity that holds it. An attribute has a name that describes the property and a type that describes the kind of attribute it is, such as varchar for a string, and int for integer. When an ERD is drawn for physical database development, it is important to ensure the use of types that are supported by the target RDBMS.

Figure 38/ERD Entity Attributes



3.Primary Key

Also known as PK, a primary key is a special kind of entity attribute that uniquely defines a record in a database table. In other words, there must not be two (or more) records that share the same value for the primary key attribute. The ERD example below shows an entity 'Product' with a primary key attribute 'ID', and a preview of table records in the database. The third record is invalid because the value of ID 'PDT-0002' is already used by another record.

4.Foreign Key

Also known as FK, a foreign key is a reference to a primary key in a table. It is used to identify the relationships between entities. Note that foreign keys need not be unique. Multiple records can share the same values. The ER Diagram example below shows an entity with some columns, among which a foreign key is used in referencing another entity.

1. Relationship

A relationship between two entities signifies that the two entities are associated with each other somehow. For example, a student might enroll in a course. The entity Student is therefore related to Course, and a relationship is presented as a connector connecting between them.

2. Cardinality

Cardinality defines the possible number of occurrences in one entity which is associated with the number of occurrences in another. For example, ONE team has MANY players. When present in an ERD, the entity Team and Player are inter-connected with a one-to-many relationship. In an ER diagram, cardinality is represented as a crow's foot at the connector's ends. The three common cardinal relationships are one-to-one, one-to-many, and many-to-many.

- One-to-One cardinality

A one-to-one relationship is mostly used to split an entity in two to provide information concisely and make it more understandable. The figure below shows an example of a one-to-one relationship.

- One-to-Many cardinality

A one-to-many relationship refers to the relationship between two entities X and Y in which an instance of X may be linked to many instances of Y, but an instance of Y is linked to only one instance of X. The figure below shows an example of a one-to-many relationship.

• Many-to-Many cardinality

A many-to-many relationship refers to the relationship between two entities X and Y in which X may be linked to many instances of Y and vice versa. The figure below shows an example of a many-to-many relationship. Note that a many-to-many relationship is split into a pair of one-tomany relationships in a physical ERD. You will know what a physical ERD is in the next section.

ERD Model

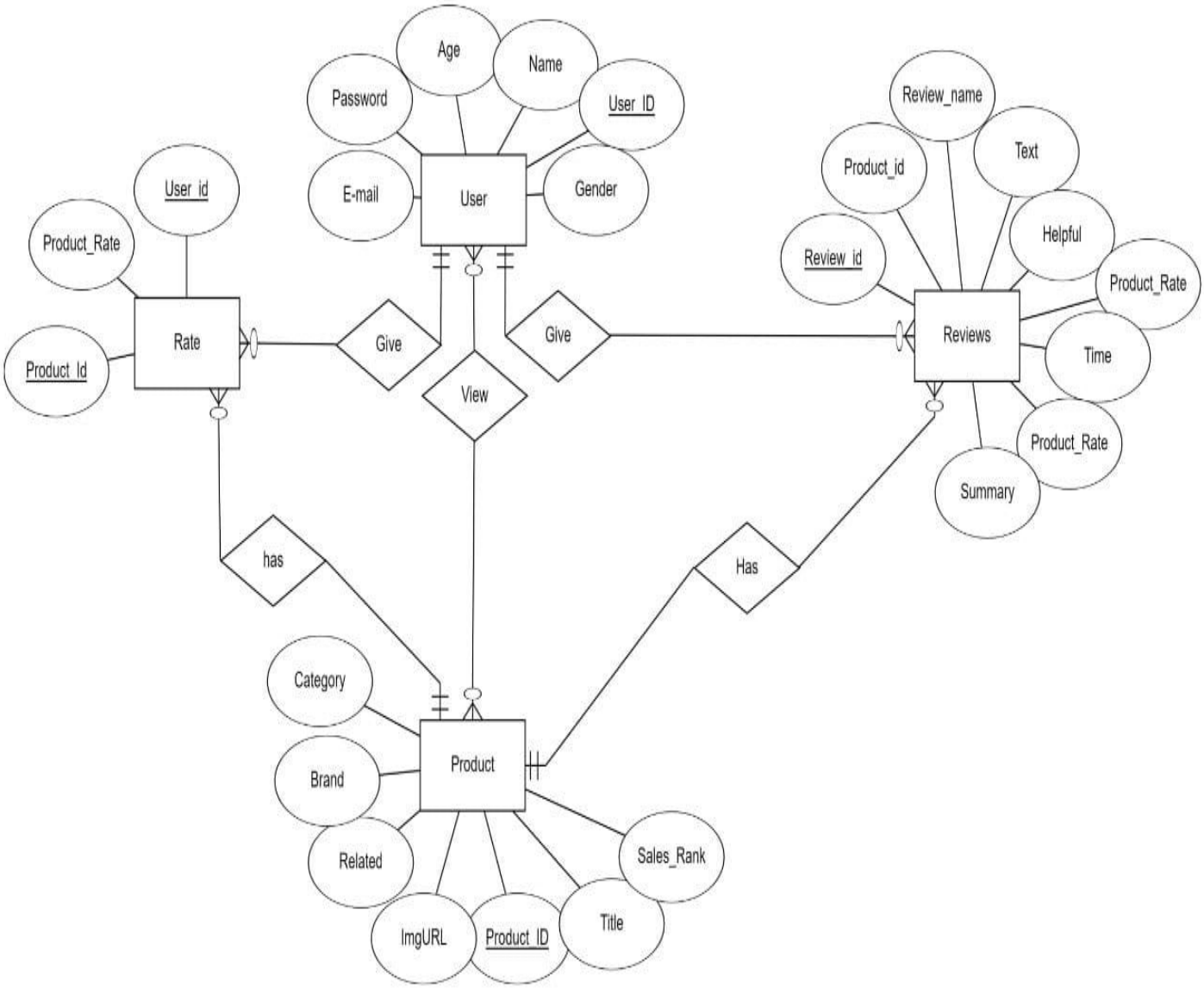


Figure 39/ERD Model

• User: This entity represents a user of the system who can view products, write reviews and Gives Ratings. The attributes of a user include:

-User ID: A unique identifier for each

-Name: The name of the use

-Email: The email address of the user.

-Password: The password of the user.

-Gender: The gender of the user.

-Age: The age of the user.

• Product: This entity represents a product that can be viewed and reviewed. The attributes of a product include:

-Product ID: A unique identifier for each product.

-Title: The title of the product.

-ImgURL: The URL of an image of the product.

-Brand: The brand of the product.

-Category: The category of the product.

-Related: A list of related products.

-Sales Rank: The sales rank of the product.

• Review:

This entity represents a review of a product. The attributes of a review include:

Review ID: A unique identifier for each review.

User ID: The ID of the user who wrote the review.

Product ID: The ID of the product that the review is for.

Text: The text of the review.

Time: The time the review was written.

Helpful: Whether the review was helpful or not.

● Rating:

This entity represents the ratings of each product given by each user. The attributes of Rating include:

- Product id: The ID of the product that has been rated

- User id: The ID of the User that has rated

- Product Rating: The number of Rating

This Entity has composite key which means more than a single column is used to uniquely identify each row in the table.

Business Rule

1-User and Review:

• Only registered users can write reviews.

• One-to-Many: This is the core relationship. One user can write many reviews, but each review must be written by one user. This is represented by a single User ID referencing multiple Review IDs.

2-Product and Review:

• One-to-Many: Similar to the previous relationship, one product can have many reviews, but each review must be for one product. This is also represented by a single Product ID referencing multiple Review IDs.

3- User and Rate:

• One-to-Many: one user can give many ratings, but each Rating Must be for one user.

4- Product and Rate

• One-to-Many: one product can have many ratings, but each Rating must be for one product.

5-User and Product:

• Many-to-Many: one user can view many products, and each product can be Viewed by many user.

Chapter 5: Dataset

Title:

Amazon Customer Reviews Dataset from 2014: A Comprehensive Resource for Recommendation Systems and Sentiment Analysis

Introduction:

The Amazon Customer Dataset from 2014 serves as a valuable resource for developing recommendation systems and conducting sentiment analysis. This dataset contains a wealth of information gathered from customer reviews on the Amazon platform, providing insights into consumer preferences, sentiments, and product feedback. By leveraging this dataset, businesses can extract meaningful patterns, improve customer experience, and make data-driven decisions.

Dataset Overview:

The dataset encompasses a vast collection of customer reviews spanning various product categories on Amazon. With a temporal focus on the year 2014, it captures a snapshot of customer sentiments and opinions during that period. The dataset comprises structured information, including review texts, ratings, product metadata, and additional attributes that deliver valuable context for analysis.

Size and Scope:

The Amazon Customer Dataset from 2014 boasts a substantial volume of data (142.8 million reviews), encompassing a diverse range of products across multiple domains. The dataset contains millions of customer reviews, offering ample opportunities for comprehensive analysis and modeling. The broad scope of products covered ensures the availability of rich and varied insights, allowing for a comprehensive understanding of customer preferences and sentiments.

Benefits and Insights:

By utilizing the Amazon Customer Reviews Dataset from 2014, businesses can gain the following benefits and insights:

1. Customer Behavior Analysis: Analyze customer preferences, identify emerging trends, and understand changes in consumer behavior over time.
2. Product Performance Assessment: Evaluate product performance based on customer feedback, ratings, and sentiments to identify areas of improvement or competitive advantage.
3. Personalized Recommendations: Develop personalized recommendation systems to enhance the customer experience, drive sales, and foster customer loyalty.
4. Market Intelligence: Gain valuable market intelligence by understanding customer sentiments towards specific product categories, brands, or features.

Chapter 6: Implementation

6.1Web tools that will be used in our project

Frontend:

Vue.js

Vue.js is a JavaScript framework that allows US to create declarative and component-based user interfaces with HTML, CSS, and JavaScript. It's known for its simplicity, ease of use, and component-based architecture.

Pros:

1. Easy to Learn: Vue.js is loved by developers because it is versatile and lightweight. Anyone who has some experience in front-end development can master it within a couple of days.

2. Flexibility: With Vue.js, you can easily modify your structure, add new blocks, or remove existing ones, just like you would do with a Lego set.

3. Component-Based Architecture: Vue.js follows a component-based architecture, which means all the frontend application code can be divided into independent components. These components, consisting of template, logic, and styles, are bound together to form the web app.

Backend:

Flask

Flask is a micro web framework written in Python.

Pros:

1. Lightweight: Flask is like a small car. It's easy to handle and doesn't require a lot of resources, making it ideal for simpler or smaller projects.

2. Flexibility: Flask is like a car with customizable features. we can modify it to suit our needs, adding or removing parts as necessary.

3. Easy to Learn: we can easily learn how to drive Flask. It's designed to be straightforward and intuitive.

4. Community Support: Imagine having a large group of car enthusiasts who are always ready to help you, share their unique car modifications, or solve any issues you might have. That's what the Flask community is like.

5. Testing: Flask allows for testing, like taking your car for a test drive before buying it. This ensures that everything is working as expected.

High-level flow of how we can structure The project with Flask and Vue.js:

1. Set up Environment: Install the necessary software for your project, including Python, Flask, Node.js, and Vue.js.

2. Create Flask Backend:

- Define routes for application. For example, we might have one route to get recommendations and another to perform sentiment analysis.

- In each route, use our Python models to process the data and return the results.

3. Create Vue.js Frontend:

- Define components for our application. For example, you might have one component for user input, another for displaying recommendations, and another for showing sentiment analysis results.

- In each component, use Vue.js to define how the component should look (using HTML and CSS) and how it should behave (using JavaScript).

4. Connect Frontend and Backend:

- In our Vue.js components, use the `fetch` API or a library like axios to send HTTP requests to your Flask backend.

- In your Flask routes, return the results in a format that Vue.js components can understand (like JSON).

5.Test Application: Make sure all parts of application are working correctly together.

6.Deploy Application: Once everything is working locally, deploy application to a server so others can use it.

Database System we will use Mysql:

Absolutely, storing previous data about users' recommendations and sentiment can be very useful for improving the user experience. Here's how you can do it:

- Database Setup: Choose a database that suits our needs (like PostgreSQL, MySQL, MongoDB, etc.). Set up tables or collections to store user data, recommendation data, and sentiment data.
- Storing Recommendations: After generating a recommendation for a user, we can store it in database along with a timestamp and user identifier. This allows us to keep track of what recommendations were made to each user and when.
- Storing Sentiment Data: Similarly, after performing sentiment analysis, we can store the results in our database along with a timestamp and user identifier. This allows us to track how a user's sentiment changes over time.
- Retrieving Data: When a user returns to application, retrieve their past recommendations and sentiment data from the database. This can be used to provide a more personalized experience for the user.

6.2 Code review:

1-Import library:

Import basic libraries like pandas and Numpy and Nltk libraries and specific modules or part of the library that deals with sentiment analysis using VADER

2- Read dataset:

Read CSV file containing product_id ,reviwername , reviwerid,reviewText and summary

```
In [1]: #Pandas is used to read data files (e.g. xlsx, csv) and creates dataframes for them
import pandas as pd
#NLTK stands fro Natural Language Toolkit, it offers tools for tokenization, lemmatization, and other text preprocessing tasks
import nltk
#stopwords is a collection of common words in a language that are often filtered out during text processing tasks
from nltk.corpus import stopwords
#word_tokenize is a function used for tokenization
#types of tokenization: Word tokenization, Sentence tokenization, and Character tokenization
from nltk.tokenize import word_tokenize
#WordNetLemmatizer is a class used for lemmatization (e.g. foot ---> feet, went ---> go, going ---> go)
from nltk.stem import WordNetLemmatizer
#TextBlob is used to calculate sentiment scores
from textblob import TextBlob
```

```
In [2]: df = pd.read_csv("C:\\Users\\magdy\\Downloads\\Musical_instruments_reviews.csv") #read data by pandas library
#astype(str) is used to convert the data type of a Pandas DataFrame to string
df = df.astype(str)
```

```
In [15]: df.head() #print the first 5 rows of the dataframe
```

Out[15]:

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime	concat	tokens	filtered_te
0	A2IBPI20UZIR0U	1384719342	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	Not much to write about here, but it does exac...	5.0	good	1393545600	02 28, 2014	not much to write about here, but it does exac...	[not, much, to, write, about, here, ,, but, it...	[much, wri exac suppose filte pop
1	A14VAT5EAX3D9S	1384719342	Jake	[13, 14]	The product does exactly as it should and is q...	5.0	Jake	1363392000	03 16, 2013	the product does exactly as it should and is q...	[the, product, does, exactly, as, it, should, ...	[produ exac qui affordable reali;
2	A195EZSQDW3E21	1384719342	Rick Bennette "Rick Bennette"	[1, 1]	The primary job of this device is to block the...	5.0	It Does The Job Well	1377648000	08 28, 2013	the primary job of this device is to block the...	[the, primary, job, of, this, device, is, to, ...	[prima job, devi blo brea would,

3-Data analysis:

Data info and description, count the distinct products by product_id

4- Data cleaning:

Clean the dataset by removing any rows containing missing values

```
In [26]: df.info() #used to get the info of the dataframe data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10261 entries, 0 to 10260
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   reviewerID       10261 non-null  object
 1   asin             10261 non-null  object
 2   reviewerName     10261 non-null  object
 3   helpful          10261 non-null  object
 4   reviewText       10261 non-null  object
 5   overall          10261 non-null  object
 6   summary          10261 non-null  object
 7   unixReviewTime   10261 non-null  object
 8   reviewTime       10261 non-null  object
 9   concat           10261 non-null  object
 10  tokens           10261 non-null  object
 11  filtered_text    10261 non-null  object
 12  lemma            10261 non-null  object
 13  Sentiment_Score  10261 non-null  float64
dtypes: float64(1), object(13)
memory usage: 1.1+ MB
```

5- Concatenate:

Concatenated the two columns reviewText and summary combine them in a single column

```
In [29]: df["reviewText"].isnull().sum() #used to calculate the number of null values
```

```
Out[29]: 0
```

```
In [21]: number_of_unique_products = df["asin"].nunique() #used to count the number of unique products in the whole data
         number_of_unique_products
```

```
Out[21]: 900
```

```
In [4]: #concatenating the two columns reviewText and summary by adding space between them, and adding them to a new column
        df["concat"] = df['reviewText'] + " " + df['summary']
```

```
In [14]: list = ["?", "!", "@", "/", "(", ")", '#', '%', '^', "&", "*", "-",
                 "_", ",", ":", ";", "'", "=", "+", ".", "{", "}", "<", ">", "...", "'s",
                 "'m", "\\", "]", "["] #list containing some of unwanted characters that we need to filter them from the whole reviews
```

```
In [6]: df["concat"] = df["concat"].str.lower() #used to change the data of concat column to lower case
```

```
In [7]: stop_words = stopwords.words("english") + list #adding the list to the stopwords
```

6- Preprocessing data

- Tokenization
- Remove stopwords
- Lemmatization

7-Tokenization:

Split text word by word "I like playing football"

After Tokenization

["I","like","playing","football"]

```
In [8]: df["tokens"] = df["concat"].apply(word_tokenize) #apply the function on the data of concat column

In [30]: df["tokens"]

Out[30]: 0        [not, much, to, write, about, here, ,, but, it...
         1        [the, product, does, exactly, as, it, should, ...
         2        [the, primary, job, of, this, device, is, to, ...
         3        [nice, windscreen, protects, my, mxl, mic, and...
         4        [this, pop, filter, is, great, ., it, looks, a...
                                    ...
         10256    [great, ,, just, as, expected, ., thank, to, a...
         10257    [i, 've, been, thinking, about, trying, the, n...
         10258    [i, have, tried, coated, strings, in, the, pas...
         10259    [well, ,, made, by, elixir, and, developed, wi...
         10260    [these, strings, are, really, quite, good, ,, ...
         Name: tokens, Length: 10261, dtype: object
```

8-Remove stopwords:

Make a list of unwanted characters

And added it to stopwords (collection of a common words in a language) by contacting it and choose English language.

```
In [9]: #filter the text from the stopwords
        df["filtered_text"] = df["tokens"].apply(lambda x: [word for word in x if word not in stop_words])

In [31]: df["filtered_text"]

Out[31]: 0        [much, write, exactly, supposed, filters, pop,...
         1        [product, exactly, quite, affordable.i, realiz...
         2        [primary, job, device, block, breath, would, o...
         3        [nice, windscreen, protects, mxl, mic, prevent...
         4        [pop, filter, great, looks, performs, like, st...
                                    ...
         10256                [great, expected, thank, five, stars]
         10257    ['ve, thinking, trying, nanoweb, strings, bit,...
         10258    [tried, coated, strings, past, including, elix...
         10259    [well, made, elixir, developed, taylor, guitar...
         10260    [strings, really, quite, good, would, n't, cal...
         Name: filtered_text, Length: 10261, dtype: object
```

8-Lemmatization:

Return the words to its root (pos = v) for lemmatize it from words to verbs

Feet --> foot

Went, going, gone --> go

```
In [37]: #pos='v' used to lemmatize(change) words to verbs
         #n ---> noun, a ---> adjective, r ---> adverb
         def lemmatize_past_to_present(text):
             lemmatizer = WordNetLemmatizer() #create a variable contains the class WordNetLemmatizer()
             #return the words lemmatized and seperate them by a space
             lemmatized_words = [lemmatizer.lemmatize(word, pos='v') for word in text]
             return ' '.join(lemmatized_words)

         # Apply the function to the DataFrame column
         df['lemma'] = df['filtered_text'].apply(lemmatize_past_to_present)

In [38]: df['lemma'][0]

Out[38]: 'much write exactly suppose filter pop sound record much crisp one lowest price pop filter amazon might well buy honestly work
         despite price good'
```

9-Sentiment score:

Calculate sentiment score by textblob specific library

```
In [39]: def calculate_sentiment_score(text):
             return TextBlob(text).sentiment.polarity

         #apply the function to the DataFrame column
         df['Sentiment_Score'] = df['lemma'].apply(calculate_sentiment_score)

In [40]: df['Sentiment_Score']

Out[40]: 0        0.400000
         1        0.012500
         2        0.167500
         3        0.425000
         4        0.800000
                    ...
         10256    0.800000
         10257    0.211722
         10258    0.335714
         10259    0.180573
         10260    0.384375
         Name: Sentiment_Score, Length: 10261, dtype: float64
```

Chapter 7: References:

- Recommendation system

  https://thingsolver.com/blog/introduction-to-recommender-systems/

  https://cse.iitkgp.ac.in/~pawang/courses/SC16/recSys.pdf

  https://www.coursera.org/learn/basic-recommender-systems/home/week/1

- Sentiment Analysis

  https://www.sciencedirect.com/science/article/pii/S2665917423001265

  https://github.com/prrao87/fine-grained-sentiment

- System analysis

  System Analysis - an overview | ScienceDirect Topics

- UML diagrams

  https://tallyfy.com/uml-diagram/

- Agile

  https://www.smartsheet.com/understanding-agile-software-development-lifecycle-and-process-workflow