IBM NALAYATHIRAN PROJECT REPORT

Smart Fashion Recommender Application

TEAM ID : PNT2022TMID13339

TEAM MEMBERS

Sathiyananth P	(621319104051)
Karthikeyan T	(621319104022)
Naveen V	(621319104037)
Salman M J M	(621319104047)

TABLE OF CONTENT

CHAPTER 1
INTRODUCTION

The fashion sector is currently being compelled to accelerate its digital transition, bringing conventional offline services online. In the United Kingdom, 77% of retail enterprises use AI in the e-commerce market via chat-bots. AI and fashion studies have looked at the aspects that influence customer acceptability (Rese, Ganster, & Baier, 2020; Sanny, Susastra, Roberts, & Yusramdaleni, 2020), forecasting and production management (Sun and Zhang, 2018), smart textiles (Roh, Chi, and Kang, 2010), trend analysis of colour palettes (Lai and Westland, 2020), accurate image labelling through machine learning (ML) (Park and Choi, 2020), and "conversational agents," also referred to as "chat-bots" (Grewal, Roggeveen, and Nordfält, 2017). Chat-bots can offer individualized shopping experiences across both offline and online channels while fostering customer welfare. However, the complexity of human language and the usefulness of the chat-bot in this setting continue to be the key challenges. Chat-bots might be seen of as Natural User Interfaces that allow people to communicate with computers more naturally using everyday language, making e-commerce more acceptable in general and in the sale of clothing in particular. As a result, chat-bots may be viewed as a crucial, pertinent interface component for many fashion e-commerce operations, such as making suggestions, browsing and searching through extensive catalogue, enhancing the functionality of virtual fitting rooms, and offering (post-sale) customer assistance.

When it comes to designing and implementing chatbots within the customer journey, Pantano and Pizzi (2020) advise that: retailers may want to select already designed chatbot platforms or to design their own; from a technological point of view, conversational agents are designed to mimic natural language; The most popular platforms include social media (such as instant messengers like Facebook Messenger) and voice-based AI (which uses audio recognition) like Siri and Alexa. However, analytical abilities are still needed to learn from consumer data. From a computational standpoint, there is a trend to increase the chatbot's responsiveness, and patents were mainly made to offer fresh Natural Language Processing (NLP), digital communication, and customer management techniques. Just because a few firms are prepared to construct their own chatbot, it is essential that managers and designers are aware of the potential and constraints presented by the most recent technical developments.

1.1 PROJEC OVERVIEW

One way to categories chatbot studies is into computational and non-computational aspects. Computational aspects are those that fall under the umbrella of computer science or information technology, such as the use of NLP, and non-computational aspects are all other aspects, such as researching consumer acceptance. To further develop these prior classifications There are several distinct categories for computational papers. Here, we concentrate on the broad categories. Chatbots as conversational Recommender systems adopt a more complex approach, providing a larger range of interactions that enhance preference elicitation and communicate with users in natural language (NL). Since this is what the so-called context-aware Recommender systems aim to do, chatbots can offer tools to collect contextual data. In the case of chatbots used in fashion e-commerce, additional capabilities pertaining to online sales may be necessary, such as those that engage and encourage users to purchase a product and eventually involve negotiating conversations by simulating a salesperson speaking with a customer (Jusoh, 2018). This type of system addresses the key difficulties of natural language-based interaction, such as information extraction, NL interpretation, and NL production, which may be particular to the fashion domain.

1.2 PURPOSE OF THE PROJECT

In this Smart Fashion Recommender Application, the consumer initially logs in to the website and browses the items. Instead of having to navigate through multiple screens to book things online. The new revolutionary approach is that the user will instantly complete their online shopping based on their preferences without having to search. The user can also directly chat to the Chatbot about the items. Obtain suggestions based on the information supplied by the user. It is possible to accomplish this by utilizing the chatbot. This chatbot functionality will make it simple for users and customers to get what they need. The user's orders and preferences can be managed by chatbots as well. Additionally, using information that is already saved in the database and the chatbot, customers will be offered suggestions based on their preferences. Likewise, it will let the user know when new sales or promotions are available.

1) . PURPOSE

Smart Fashion Recommender Application are becoming a popular topic. The main purpose of an online ordering system is to provide customers for a easy way to place an order a the product with the help of chatbot over the intenet.

2) FEASIBILITY STUDY

The measure to which the creation of an informant system will benefit or be practicable for an organisation. The viability of the issue is assessed. So far, we've focused on the following two primary categories of feasibility research throughout the development of the project Smart Fashion Recommender Application:

- ✓ **Technical Feasibility**

  It assesses the technological solution's effectiveness as well as the availability of technical resources and experience. The term "technical feasibility" refers to what is feasible and reasonable. It mostly covers their principal concerns.

- ✓ **Operational Feasibility**

The operational feasibility of a project is determined by how effectively it will assist the client and service provider during the operational phase.

CHAPTER 2

LITERATURE SURVEY

[1] TITLE: Personalized fashion Recommender system with image based neural networks. In IOP Conference Series: Materials Science and Engineering (Vol. 981, No. 2, p. 022073). IOP Publishing.

AUTHOR: Sridevi, M., ManikyaArun, N., Sheshikala, M., & Sudarshan, E. (2020, December)**.**

Processes the DeepFashion dataset's photos using neural networks, and then creates final suggestions using a closest neighbor-backed Recommender. It processes the DeepFashion dataset's photos using neural networks, and then creates final suggestions using a closest neighbor- backed recommender. With an increase in the standard of living, peoples' attention gradually moved towards fashion that is concerned to be a popular aesthetic expression. Humans are inevitably drawn towards something that is visually more attractive. This tendency of humans has led to development of fashion industry over the course of time. However, given too many options of garments on the e-commerce websites, has presented new challenges to the customers in identifying their correct outfit. Thus, in this paper, we proposed a personalized Fashion Recommender system that generates recommendations for the user based on an input given. Unlike the conventional systems that rely on user's previous purchases and history, this project aims at using an image of a product given as input by the user to generate recommendations since many-a-time people see something that they are interested in and tend to look for products that are similar to that. We use neural networks to process the images from DeepFashion dataset and a nearest neighbour backed recommender to generate the final recommendations.

[2] TITLE: Modeling Instant User Intent and Content-level Transition for Sequential Fashion Recommendation. IEEE Transactions on Multimedia.

AUTHOR: Yujuan Ding, Yunshan Ma, Wai Keung Wong, Tat-Seng Chua (2021).

Fashion recommendation, aiming to explore specific user preference in fashion, has become an important research topic for its practical significance to the fashion business sector. However, little work has been done on an important sub-task called sequential fashion recommendation, which aims to capture additional short-term fashion interest of users by modeling the item-to-item transitions. In this paper, we propose a novel Attentional Content-level Translation-based Recommender (ACTR) framework, which simultaneously models the instant user intent of each transition and the intent-specific transition probability. Specifically, we define instant intent with the relationships between adjacent items that the users interacted, which are the three fundamental domain-specific relationships of: match , substitute and others . To further exploit the characteristics of fashion domain and alleviate the item transition sparsity problem, we augment the item-level transition modeling with multiple sub-transitions using various content-level attributes. An attention mechanism is further devised to effectively aggregate multiple content-level transitions. To the best of our knowledge, this is the first work that specifies the implicit user actions in online fashion shopping with explicit instant intent, which enhances the connectivity of fashion items and boosts the recommendation performance. Extensive experiments on two real-world fashion E-commerce datasets demonstrate the effectiveness of the proposed method in sequential fashion recommendation.

[3] TITLE:Fashion Recommender Systems in Cold Start

AUTHOR: Mehdi Elahi and Lianyong Qi(2020).

Fashion is defined as “The cultural construction of the embodied identity”. It is commonly described as the prevailing style of dress or behavior and it can be characterized by the notion of change. Fashion encompasses various forms of self fashioning ranging form street styles to high fashion made by designers. A major challenge in the fashion domain is the increasing Variety, Volume, and Velocity of fashion production which makes it difficult for the consumers to choose which product to purchase 2. Shakespeare has -somehow- addressed this issue a long time ago by noting “that the fashion wears out more apparel than the man” [109]. This is not necessarily all negative as the more choices available the better the opportunity for consumers to choose appealing products. However, this phenomenon shall result in the problem of choice overload, i.e., the problem of having unlimited number of choices, especially when they do not differ significantly from each other. Recommender systems can mitigate this problem by suggesting a personalized selection of items (i.e., fashion products) that are predicted to be the most appealing for a target user (i.e., fashion consumer). This is done by filtering irrelevant items and recommending a shortlist of the most relevant ones for the users. An effective filtering requires the system to (thoroughly) analyze the user preferences and (deeply) learn the particular taste and affinity of every individual user.

[4]TITLE:IBM Watson Application as FAQ Assistant about Moodle.

AUTHOR: Jeferson da Silva Oliveira, Danubia Bueno Espíndola, Regina Barwaldt,Luciano Maciel Ribeiro, Marcelo Pias(2019).

This complete research work presents the use of chatbot in the educational context for automation of care to the student of higher education. Due to the evolution of technologies in educational and professional contexts, new teaching/learning methods have been an emerging demand for today's society. The expansion of learning modalities such as face-to-face, combined and distance learning as well as demand anywhere, at any time, has been a premise in these education systems. With the advancement of virtual learning environments (VLEs), successful interaction depends on solutions that enable efficient communication between the student and the educational context. One way to achieve this goal has been to use chatbots to respond to students' questions in the educational field. This has become possible through artificial intelligence and natural language processing embodied in simple service platforms. The purpose of this article is to develop a chatbot that interacts with students through text messages, on subjects in a closed context, to students' doubts about the Higher Education Institution Course. To validate chatbot performance, it will be submitted to a set of questions of the same context that were not correctly answered in the previous interactions. The results of these interactions will be analyzed in order to verify the efficiency and contribution of a chatbot as a support tool for the student. It is hoped that this study contributes to solutions for assistance automation in the learning context.

[5]TITLE:Data-Driven Web APIs Recommendation for Building Web Applications

AUTHOR: Lianyong Qi, Qiang He, Feifei Chen, Xuyun Zhang, Wanchun Dou,Qiang Ni.(2020)

The ever-increasing popularity of web APIs allows app developers to leverage a set of existing web APIs to achieve their sophisticated objectives. The heavily fragmented distribution of web APIs makes it challenging for an app developer to find appropriate and compatible web APIs. Currently, app developers usually have to manually discover candidate web APIs, verify their compatibility and select appropriate and compatible ones. This process is cumbersome and requires detailed knowledge of web APIs which is often too demanding. It has become a major obstacle to further and broader applications of web APIs. To address this issue, we first propose a web API correlation graph built on extensive data about the compatibility between web APIs. Then, we propose WAR (Web A PIs R ecommendation), the first data-driven approach for web APIs recommendation that integrates web API discovery, verification and selection operations based on keywords search over the web API correlation graph. WAR assists app developers without detailed knowledge of web APIs in searching for appropriate and compatible web APIs by typing a few keywords that represent the tasks required to achieve app developers' objectives. WAR can significantly save app developers' time and effort in searching for web APIs. We conducted large-scale experiments on 18,478 real-world web APIs and 6,146 real-world apps to demonstrate the usefulness and efficiency of WAR.

2.1 EXISTING SYSTEM

The age of recommendation systems began in the 1990s, as a result of considerable research development in Collective Intelligence. During this time, consumers were often given suggestions based on their rating structure. However, there are other ways to gather information about a consumer's preference for a product via the Internet. These data may be accessed in the form of voting, labeling, reviewing, and the number of likes or dislikes provided by the user. It may also contain blog reviews, YouTube videos, or messaging regarding a product.

Existing web apps only provide search choices to find the required items for the customer/user; nevertheless, by using this strategy, the user will browse to other pages that the user does not need to view. If a chatbot is available, it will mainly be utilized for customer service purposes only. This user will be misled as to which product he desires.

2.3 REFERENCES:

[1]. Oishik Chatterjee, Jaidam Ram Tej, Narendra Varma Dasaraju (2022). Incorporating Customer Reviews in Size and Fit Recommendation systems for Fashion E-Commerce.

[2]. Clemencia Siro, Mohammad Aliannejadi, Maarten de Rijke (2022). Understanding User Satisfaction with Task-oriented Dialogue Systems.

[3]. Yujuan Ding, Yunshan Ma, Wai Keung Wong, Tat-Seng Chua(2021). Modeling Instant User Intent and Content-Level Transition for Sequential Fashion Recommendation.

[4]. Guang-Lu Sun, Jun-Yan He, Xiao Wu, Bo Zhao, Qiang Peng (2021). Learning fashion compatibility across categories with deep multimodal neural networks.

[5]. Gkelios, S., Sophokleous, A., Plakias, S., Boutalis, Y., & Chatzichristofis, S (2021). Deep convolutional features for image retrieval.

[6]. Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, Meng Wang (2021), A Survey on Accuracy-oriented Neural Recommendation: From Collaborative Filtering to Information-rich Recommendation.

[7]. Hongrui Zhao, Jin Yu, Yanan Li, Donghui Wang, Jie Liu, Hongxia Yang, Fei Wu (2020). Dress like an Internet Celebrity: Fashion Retrieval in Videos.

[8]. Lianyong Qi, Qiang He, Feifei Chen, Xuyun Zhang, Wanchun Dou,Qiang Ni(2020), Data-Driven Web APIs Recommendation for Building Web Applications.

[9]. Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, Jingjing Liu (2020). Recommendation systems for Fashion E-Commerce.

[10]. **Jeferson da Silva Oliveira, Danubia Bueno Espíndola, Regina Barwaldt,Luciano Maciel Ribeiro, Marcelo Pias(2019),** IBM Watson Application as FAQ Assistant about Moodle.

2.3 PROBLEM STATEMENT DEFINITION

Chatbots are used in these online tools to recommend deals and stylish clothing. The chatbot should assist the programme in learning about the user's preferences and orders. The chatbot will also make suggestions to customers based on their most recent interests. It must also advise customers of the finest current offerings and prices available. The chatbot should be able to get user input on the application.

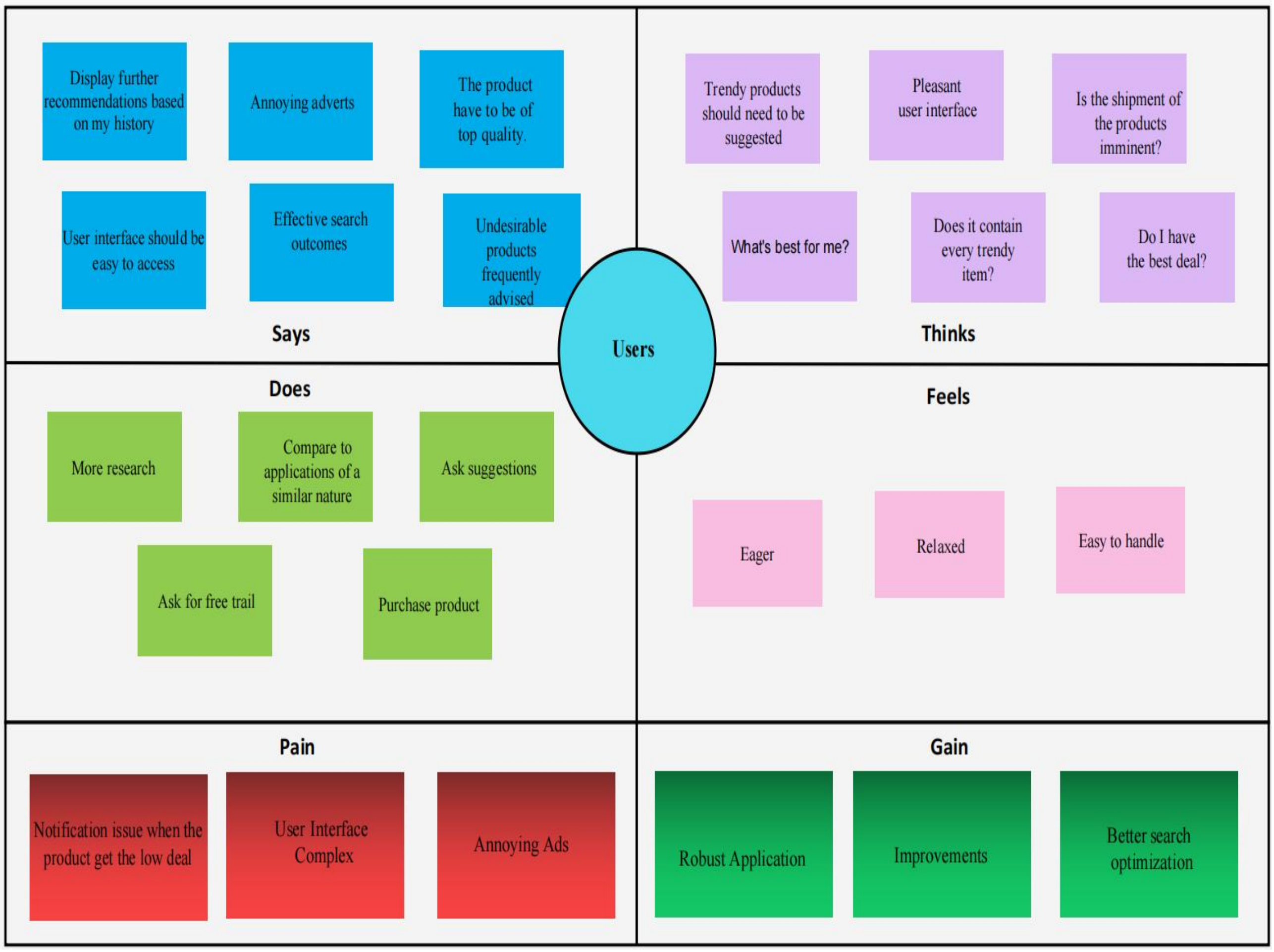
CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

It is intended to explain user demands quickly and clearly, particularly to stakeholders who are not involved in the research and design process, such as executives or clients.

It's also a terrific resource to use when you create new products and services. This is due to the fact that it keeps you focused on your ultimate aim of providing solutions that satisfy a need or solve an issue for your clients.

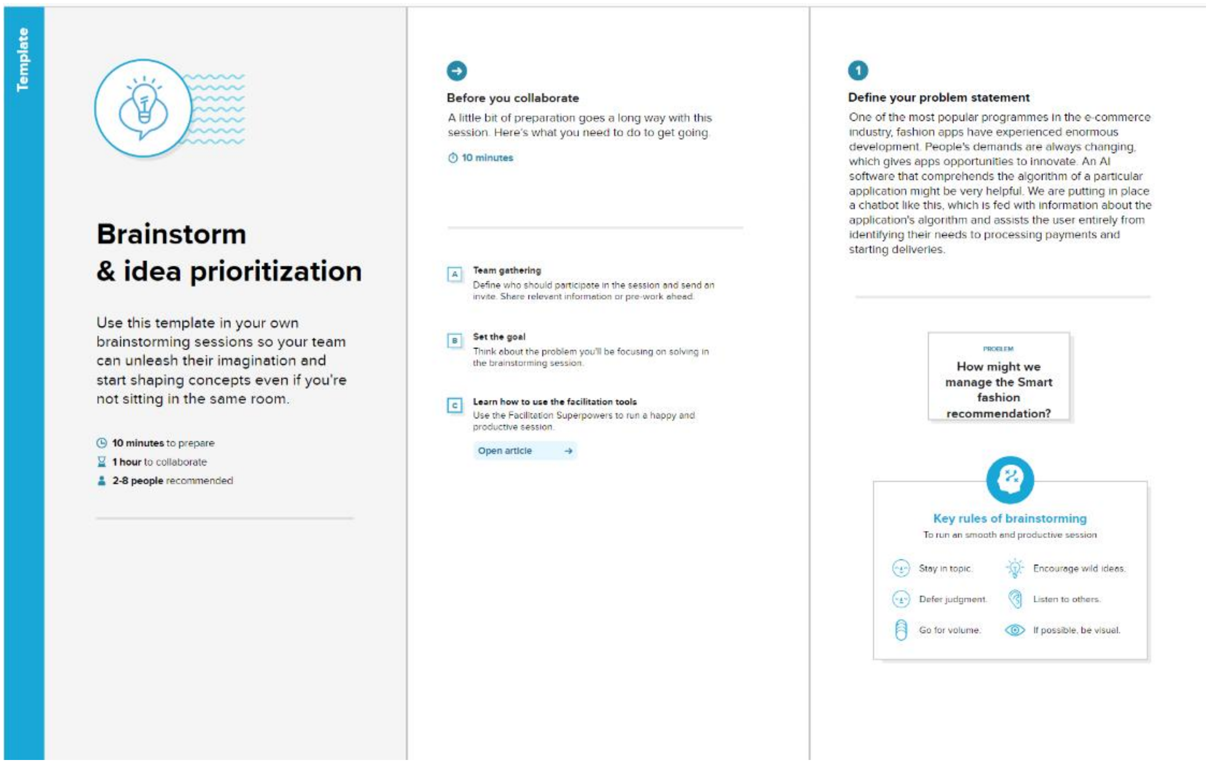


Display further recommendations based on my history
Annoying adverts
The product have to be of top quality.
User interface should be easy to access
Effective search outcomes
Undesirable products frequently advised
Says
Trendy products should need to be suggested
Pleasant user interface
Is the shipment of the products imminent?
What's best for me?
Does it contain every trendy item?
Do I have the best deal?
Thinks
Users
Does
More research
Compare to applications of a similar nature
Ask suggestions
Ask for free trail
Purchase product
Feels
Eager
Relaxed
Easy to handle
Pain
Notification issue when the product get the low deal
User Interface Complex
Annoying Ads
Gain
Robust Application
Improvements
Better search optimization

3.2 IDEATION & BRAINSTORMING

Ideation and the activity of brainstorming, a particular method for coming up with fresh ideas, are frequently closely connected. The main distinction between ideation and brainstorming is that whereas brainstorming is nearly often done in groups, ideation is typically seen as being more of a solo activity. A group of individuals are frequently gathered for a brainstorming session to generate either fresh, broad ideas or solutions to specific problems or circumstances.

The introduction of brainstorming software applications such as Brightidea and Ideawake has blurred the distinction between ideation and brainstorming. These software tools are intended to inspire company personnel to produce fresh ideas for enhancing operations and, eventually, bottom-line profitability.



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

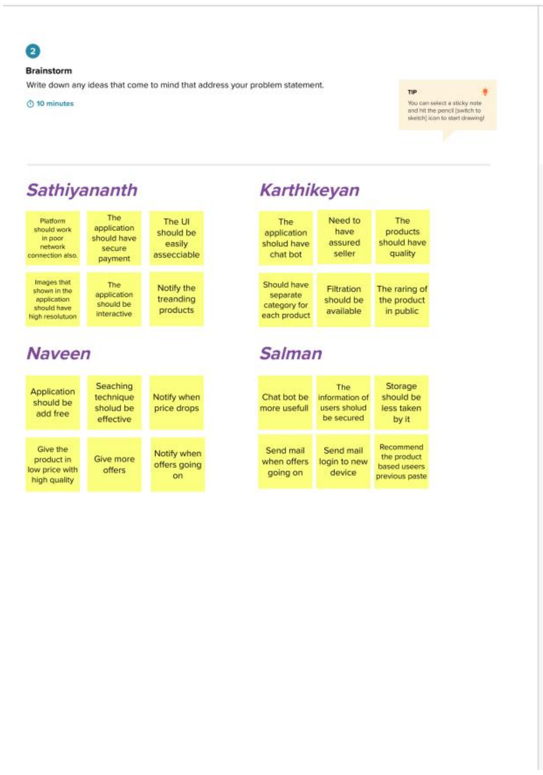
1 **Define your problem statement**

One of the most popular programmes in the e-commerce industry, fashion apps have experienced enormous development. People's demands are always changing, which gives apps opportunities to innovate. An AI software that comprehends the algorithm of a particular application might be very helpful. We are putting in place a chatbot like this, which is fed with information about the application's algorithm and assists the user entirely from identifying their needs to processing payments and starting deliveries.

PROBLEM
How might we manage the Smart fashion recommendation?

Key rules of brainstorming
To run an smooth and productive session

Stay in topic.
Encourage wild ideas.
Defer judgment.
Listen to others.
Go for volume.
If possible, be visual.



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Sathiyananth

- Platform should work in poor network connection also.
- The application should have secure payment
- The UI should be easily assecciable
- Images that shown in the application should have high resolutuon
- The application should be interactive
- Notify the treanding products

Karthikeyan

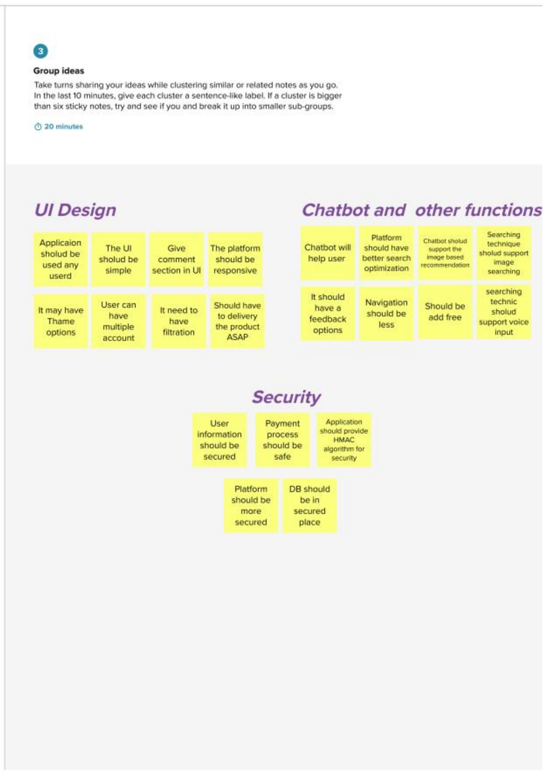
- The application sholud have chat bot
- Need to have assured seller
- The products should have quality
- Should have separate category for each product
- Filtration should be available
- The raring of the product in public

Naveen

- Application should be add free
- Seaching technique sholud be effective
- Notify when price drops
- Give the product in low price with high quality
- Give more offers
- Notify when offers going on

Salman

- Chat bot be more usefull
- The information of users sholud be secured
- Storage should be less taken by it
- Send mail when offers going on
- Send mail login to new device
- Recommend the product based useers previous paste



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

UI Design

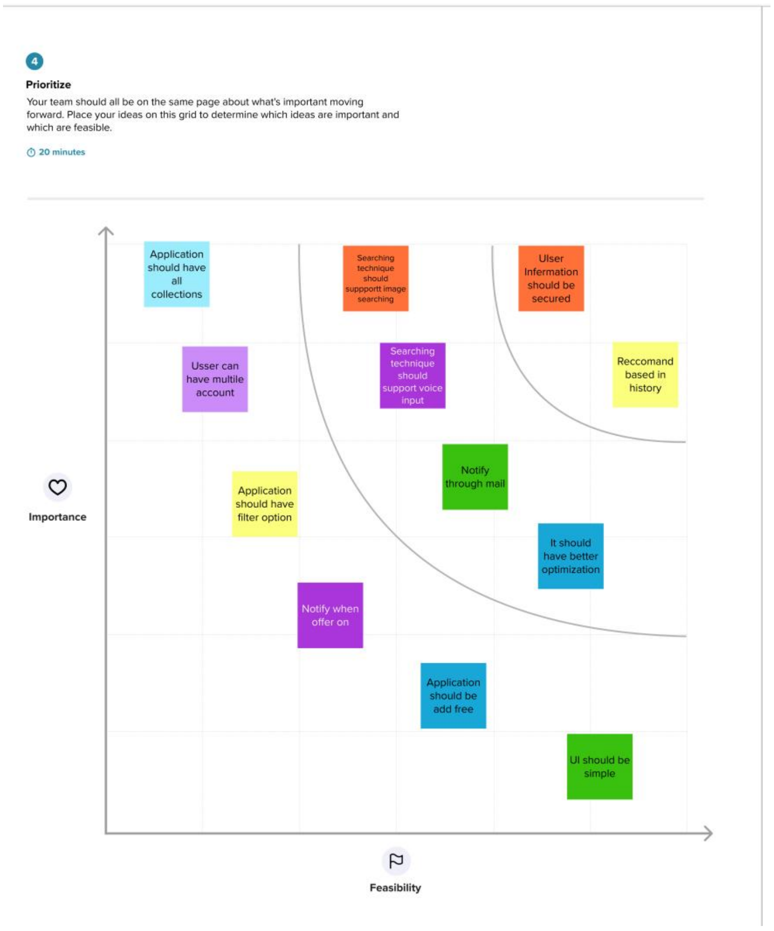
- Applicaion sholud be used any userd
- The UI sholud be simple
- Give comment section in UI
- The platform should be responsive
- It may have Thame options
- User can have multiple account
- It need to have filtration
- Should have to delivery the product ASAP

Chatbot and other functions

- Chatbot will help user
- Platform should have better search optimization
- Chatbot sholud support the image based recommendation
- Searching technique sholud support image searching
- It should have a feedback options
- Navigation should be less
- Should be add free
- searching technic sholud support voice input

Security

- User information should be secured
- Payment process should be safe
- Application should provide HMAC algorithm for security
- Platform should be more secured
- DB should be in secured place



4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

3.3 PROPOSED SOLUTION

Proposed Solution refers to the combination of software, hardware, other items or equipment and all services (including any installation, implementation, training, maintenance, and support services) required to achieve the solution indicated by Vendor in its Proposal.

Project team shall fill the following information in proposed solution;

1) Problem Statement (Problem to be solved):

Fast fashion has seen tremendous expansion in the textile and apparel industries. An effective recommendation system is needed in e-commerce platforms where there are many options available to sort, order, and effectively communicate to customers pertinent product material or information. Fast fashion companies have paid a lot of attention to image-based fashion suggestion systems because they give customers a personalized buying experience.

2) Idea / Solution description:

Here, this application will be developed using cutting-edge technologies that will let consumers utilize it effectively.

3) Novelty / Uniqueness:

We promptly handle the user's request and display recommendations based on it by using chatbots in our application.

4) Social Impact / Customer Satisfaction:

Users may utilize our application effectively by using the efficient sorting and filtering approach and chatbots.

5) Business Model (Revenue Model):

By recommending the required products to the user and very effective UI, Many users can use our application will make a more revenue.

6) Scalability of the Solution:

Any OS may be used with this application by the user utilising the Docker approach.

3.4 PROBLEM SOLUTION FIT

The Problem-Solution is a tool for entrepreneurs, marketers, and corporate innovators that helps to find ideas with higher odds of solution adoption, minimise time spent on solution testing, and gain a better understanding of the existing situation. Such information is generally acquired "on the fly," following rounds of revisions and consumer interviews, but it is critical to your success. This canvas contains everything you need to find patterns and realise what would work and why, based on the ideas of Lean Startup, and User Experience design. Simply be where your consumers are and address a genuine need, whether it's the same problem done differently or something new presented in a familiar way. In this project this are the needs for that.

1). CUSTOMER SEGMENT(S)

1. Searching for required product.
2. Ordering it.
3. Paying bill.
4. Receiving order.

2). JOBS-TO-BE-DONE / PROBLEMS

1. Check out the problems faced by customer.
2. Fix the problems.
3. Make a application user friendly.

3). TRIGGER TO ACT

1. Spend Valuable Time in our application.
2. Make the customer to feel secure.
3. Show the required product.

4). EMOTIONAL BARRIERS

1. Make the customer feel comfortable with the easy UI/UX.

5). AVAILABLE SOLUTIONS

1. Give some time to the organization for fixing the problem if any bugs occurred.
2. If any transaction problem that will be solved with customer care support.

6). CUSTOMER CONSTRAINTS

1. Need of trendy products.
2. User friendly UI.
3. Make the process simple and easy.

7). BEHAVIOUR

1. In this application customer will their required fashion and trendy collections.
2. The web application will have the user-frindly environment.

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTION REQUIREMENT

The website's functionality is determined by the functional requirements, which may vary based on the demands and company industry specialisation. For example, fashion e-commerce sites allow customers to choose numerous item features such as colour, size, sleeves, and so on; nevertheless, travel companies may require a chatbot to give user assistance; and luxury goods or jewellery firms have a zoom option on the product detail page. The following are some more critical functional requirements for your online business to consider:

4.1.1 WITH 3RD PARTIES:

It is critical to understand the third-party connectors you will require for your freshly constructed e-commerce site. This functional need focuses on optimising business operations for the clients of your website, such as flexible payment options, ERP, CRM, and PIM. Choosing the right third-party integrations for your website will help to organise it and prepare it for future company development.

4.1.2 RESPONSIVE TO MOBILE DEVICES:

Mobile devices, particularly smartphones, are where the bulk of website users visit the site in the current day. In the coming years, the population will continue to expand. As a result, investing in the development of a mobile-responsive website might yield more than simply a few bucks. Furthermore, your customer base will become more loyal to you. Using an analytical tool, you may investigate your audience and learn about their devices. Find out where the major buttons and menu choices are on the websites. Change them to improve your purchase experience.

4.1.3 PRODUCT FEATURES:

Because the PDP will have a range of product features, the e-commerce development company must be knowledgeable with how to integrate them. Isn't it true that the consumer just chooses the product's size and colour? Is it necessary to incorporate videos on a product detail page? Will specific product features be presented on a menu that resembles a mega menu? Make a list of every important product feature for your website.

4.1.4 FLOW OF ORDER AND CHECKOUT:

Orders should be handled clearly in your online store, according to the functional needs. It is necessary to determine if the website visitor want to register or utilise a guest account to make certain transactions. It may display as many order statuses as it wants, and both customers and store administrators will be able to see them. Describe how B2B orders are managed. In essence, the order and checkout procedure functions must be quite particular. This part must also mention the discount and coupon policies.

4.1.5 SOCIAL NETWORKING:

Any e-commerce company must have a social media presence. Allowing users to post content from your website on social media will increase brand recognition and bring you and your customers closer together. Learn about the social networks that your target group uses the most. It may encourage customers to share its items, blog entries, and inspiring photos by putting a similar button on the website.

4.2 NON-FUNCTIONAL REQUIREMENT

The definition of non-functional requirements is quality attributes that describe ways your product should behave. The list of basic non-functional requirements includes:

4.2.1 USABILITY:

How challenging it will be for a user to understand and utilize the system is determined by usability. This web application's low perceived workload, intuitive design, and ease of use will make it incredibly effective.

4.2.2 SECURITY:

Security criteria guarantee that unwanted access to the system and its stored data is prevented. It takes into account various degrees of authorization and authentication across different user roles.

4.2.3 RELIABILITY:

Reliability describes how possible it is that the programme will operate without interruption for a particular amount of time.

4.2.4 PERFORMANCE:

Performance is a quality attribute that describes the system's response to various user inputs. Poor performance results in a poor user experience. When the system is overwhelmed, it also threatens to undermine its safety.

4.2.5 AVAILABILITY:

It is critical to describe how the impact of maintenance may be reduced. The team must describe the most essential system components that must be available at all times while defining the availability requirements. Prepare user alerts in case the system or one of its components fail.

4.2.6 SCALABILITY:

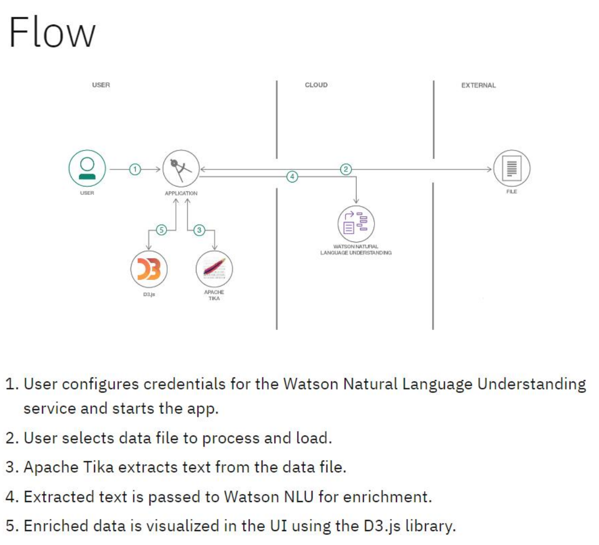
Scalability criteria specify how the system must grow without affecting its performance. Scalability affects both hardware and software. For example, one may boost scalability by adding RAM, servers, or disc space, as well as compressing data and using optimization methods.

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

A data flow diagram illustrates the movement of information through a system or process. It includes the multiple subprocesses through which data travels, as well as data inputs and outputs and data repositories. DFDs are built using standardised symbols and vocabulary to define various entities and their interactions.


Flow

1. User configures credentials for the Watson Natural Language Understanding service and starts the app.
2. User selects data file to process and load.
3. Apache Tika extracts text from the data file.
4. Extracted text is passed to Watson NLU for enrichment.
5. Enriched data is visualized in the UI using the D3.js library.

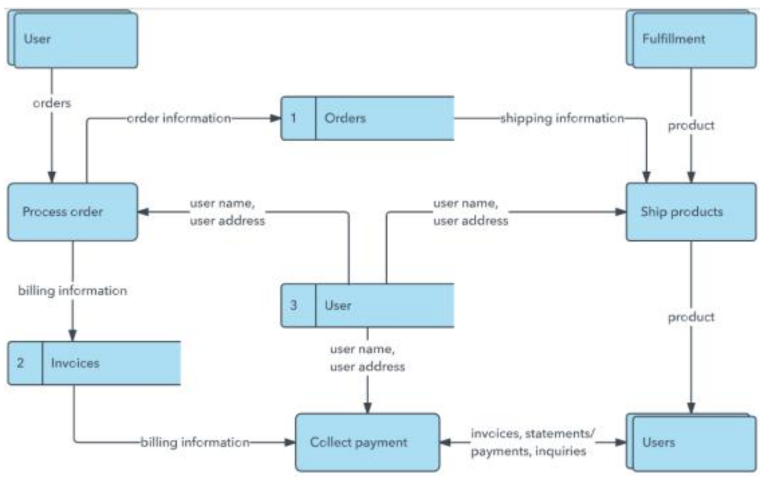
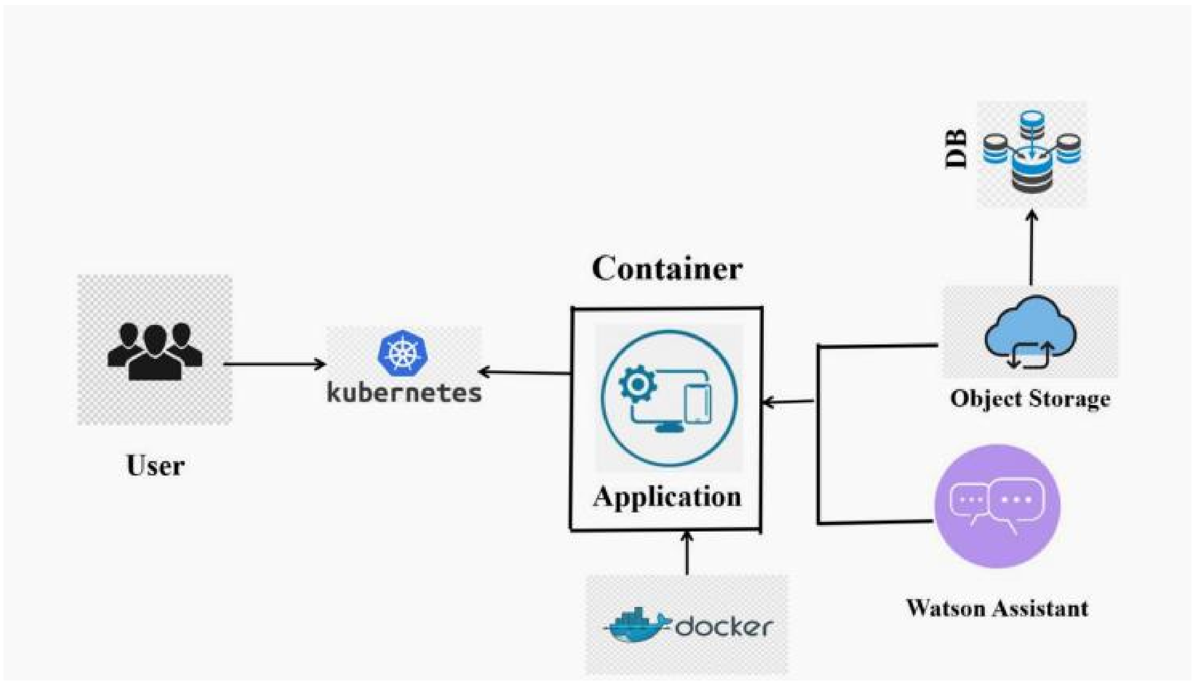


FIG: DATA FLOW DIAGRAM

5.2 SOLUTION & TECHNICAL ARCHITECTURE

A solutions architect creates the overall technical vision for a specific method to solving a business problem. A solutions architect creates the overall technical vision for a specific method to solving a business problem. They devise, explain, and supervise the approach.



5.3 USER STORIES

An informal, broad explanation of a software feature written from the viewpoint of the end user or client is known as a "user narrative." A user narrative illustrates how a piece of work will give a specific value to the customers.

S.No	Component	Description	Technology
1.	User Interface	This platform is a web-based application that allows users to quickly utilise and understand the application.	HTML, CSS, JavaScript ,bootstrap.
2.	Application Logic-1	Recommender systems seek to provide consumers with personalized online product or service suggestions in order to solve the rising issue of online information overload and improve customer relationship management.	Python/JavaScript
3.	Chat-bot	On this platform, chat bots provide quick communication with customers. It is used to infer	IBM Watson Assistant

		client preferences and provide visitors with customized experiences. E commerce chatbots may be programmed to: Assist a buyer in completing a transaction. The reasoning for a procedure in the application	
4.	Application Logic-2	A set of algorithms called as collaborative filtering provides multiple techniques for discovering comparable users or products, as well as numerous methods for determining ratings based on the ratings of comparable users. Depending on your choices, you may use a collaborative filtering technique.	Machine Learning
5.	Database	People's information, such as that of clients or users, is typically saved in databases. Databases, for instance, are used by social media platforms to store user data such as names, email addresses, and usage trends.	MySQL
6.	Cloud Database	IBM Db2 provides the capabilities required for developers, DBAs, and enterprise architects to perform real-time analytics and low-latency transactions for even the most demanding workloads.	IBM DB2
7.	File Storage	File storage, also known as file-level storage or file-based storage, is a hierarchical storage system used to organise and store data on a network-attached storage (NAS) device or on a computer hard drive.	IBM Block Storage
8.	External API-1	Docker, a software platform, makes it simple to design, test, and deploy applications. Docker packages software into standardised units called	Docker

		containers, which contain all of the required code, libraries, system tools, and runtime.	
9.	External API-2	Kubernetes automates operational activities connected with container management and provides built-in commands for application deployment, update rollout, scaling up and down to fit changing requirements, monitoring, and more.	Kubernetes
10.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

CHAPTER 6

PROJECT PLANNING & SCHEDULING

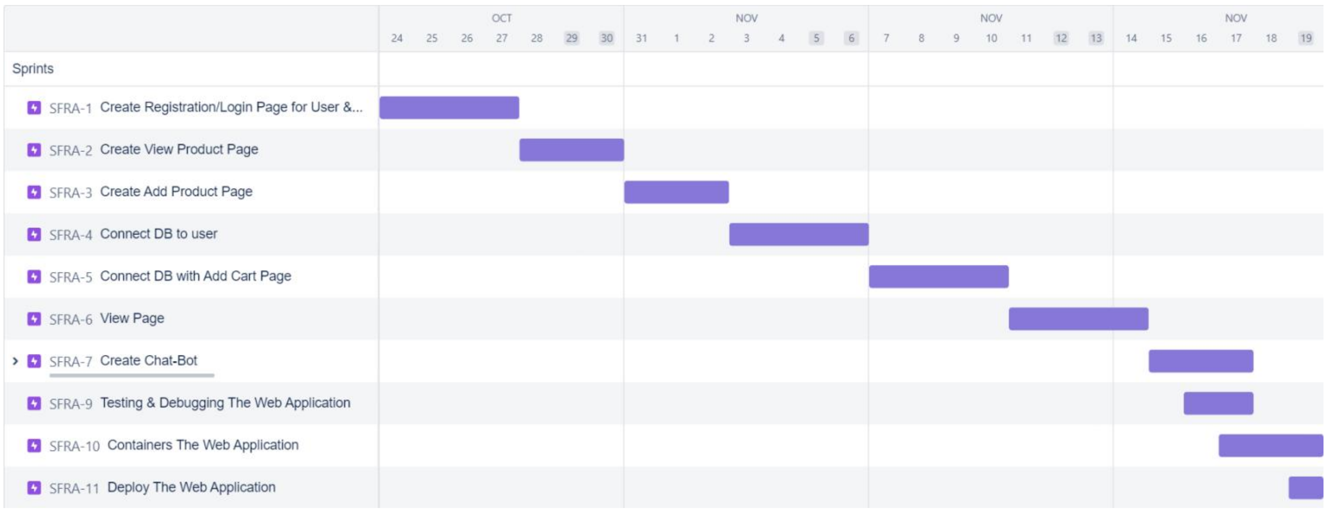
6.1 SPIRIT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user will log in to the website and browse the items accessible there.	7	High	SATHIYANANTH P KARTHIKEYAN T NAVEEN V SALMAN MJM
Sprint 1	Admin Panel	USN-2	The admin's responsibility is to verify the stock database and keep track of anything that the users buy.	8	High	SATHIYANANTH P KARTHIKEYAN T NAVEEN V SALMAN MJM
Sprint-1	Database Connectivity	USN-3	Connecting the DB to the user reg/login page and add-cart page and store the data of the user.	5	Medium	SATHIYANANTH P KARTHIKEYAN T NAVEEN V SALMAN MJM
Sprint-2	Add Cart Page	USN-4	In this page, admin will add products details and some products.	10	High	SATHIYANANTH P KARTHIKEYAN T NAVEEN V SALMAN MJM
Sprint-2	View Page	USN-5	In this page, Admin will add a view product page	10	High	SATHIYANANTH P KARTHIKEYAN T NAVEEN V SALMAN MJM
Sprint-3	Chat Bot	USN-4	The user may communicate directly with the Chatbot about the items. Get recommendations depending on the user's input.	20	High	SATHIYANANTH P KARTHIKEYAN T NAVEEN V SALMAN MJM
Sprint-4	Final Delivery	USN-5	Containerization of apps with Docker Kubernetes and application deployment Create the documents and submit the application in its entirety.	20	High	SATHIYANANTH P KARTHIKEYAN T NAVEEN V SALMAN MJM

6.2 SPRINT DELIVERY SCHEDULE

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 REPORT FROM JIRA



Sprints

SFRA-1 Create Registration/Login Page for User &...
SFRA-2 Create View Product Page
SFRA-3 Create Add Product Page
SFRA-4 Connect DB to user
SFRA-5 Connect DB with Add Cart Page
SFRA-6 View Page
SFRA-7 Create Chat-Bot
SFRA-9 Testing & Debugging The Web Application
SFRA-10 Containers The Web Application
SFRA-11 Deploy The Web Application

CHAPTER 7

CODING & SOLUTIONING

7.1 FEATURE

7.1.1 CHATBOT:

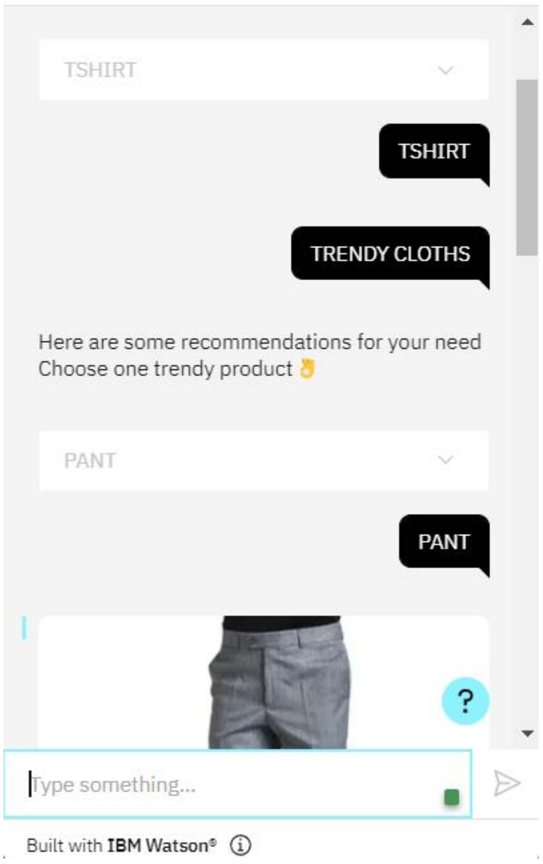
Customers are inclined to choose faster and friendlier encounters. As a result, an IBM Watson Assistant is delivered. It gives you a conversational AI platform to help you attain that full functionality. Based on the benefits of AI, it was created to help alleviate the strain of traditional assistance and to provide excellent customer service.

Get a brand-new, user-friendly interface that allows any firm to develop and maintain AI-powered chatbots and virtual agents without writing a single line of code. This improves the client experience while also providing enhanced chat, intelligent AI virtual agent, and human agent contact centre help.

7.1.2 CODING:

```
<script>
  window.watsonAssistantChatOptions = {
    integrationID: "077aae19-3307-4a90-90b3-9dc9fd7916fd", // The ID of
this integration.
    region: "au-syd", // The region your integration is hosted in.
    serviceInstanceID: "3f42b0bd-fe7d-49ed-b006-c88f5472a62f", // The ID
of your service instance.
    onLoad: function(instance) { instance.render(); }
    };
    setTimeout(function(){
        const t=document.createElement('script');
        t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/"
+ (window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
        document.head.appendChild(t);
    });
</script>
```

OUTPUT:



TSHIRT

TSHIRT

TRENDY CLOTHS

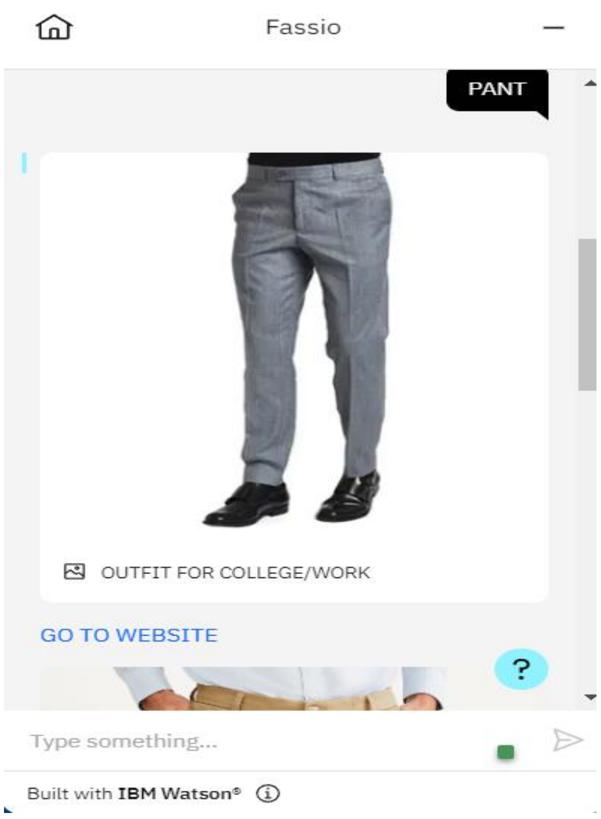
Here are some recommendations for your need
Choose one trendy product

PANT

PANT

Type something...

Built with IBM Watson®



Fassio

PANT

OUTFIT FOR COLLEGE/WORK

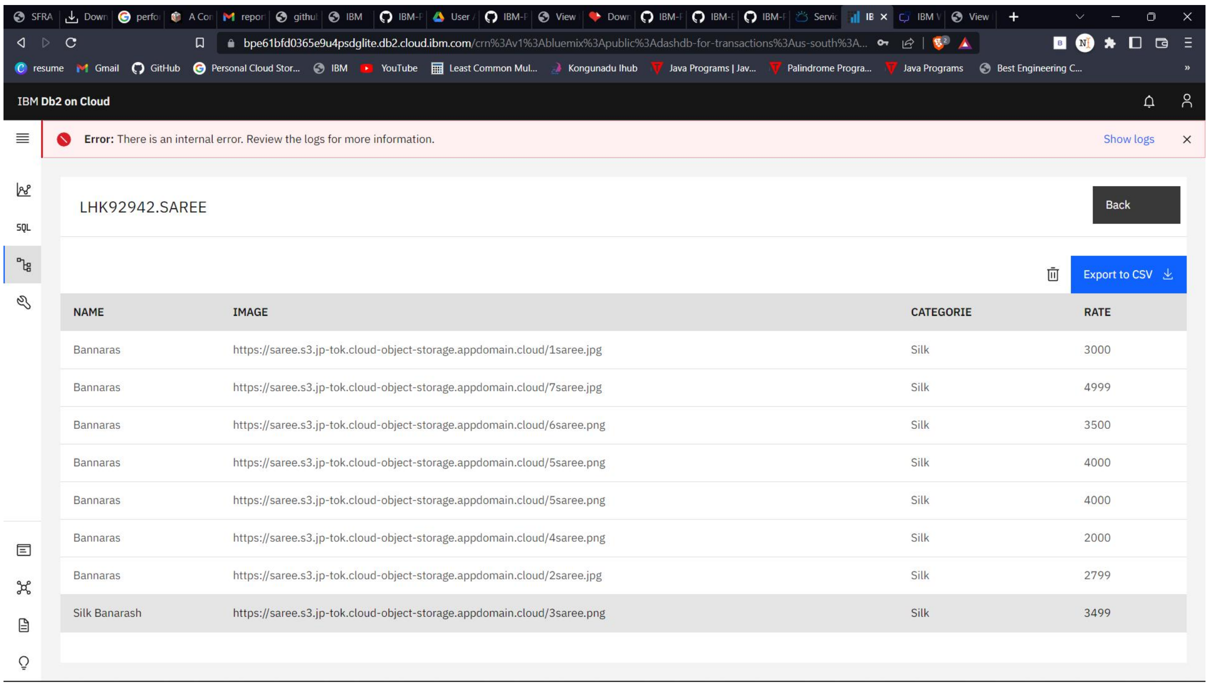
GO TO WEBSITE

Type something...

Built with IBM Watson®

7.2 DATABASE SCHEMA

The term "database schema" refers to a structure or layout that identifies a collection of data. In other words, it describes the data's structure and contentedness. Schema items like as tables, views, fields, relationships, packages, indexes, types, and many other components may therefore be found in a database schema.

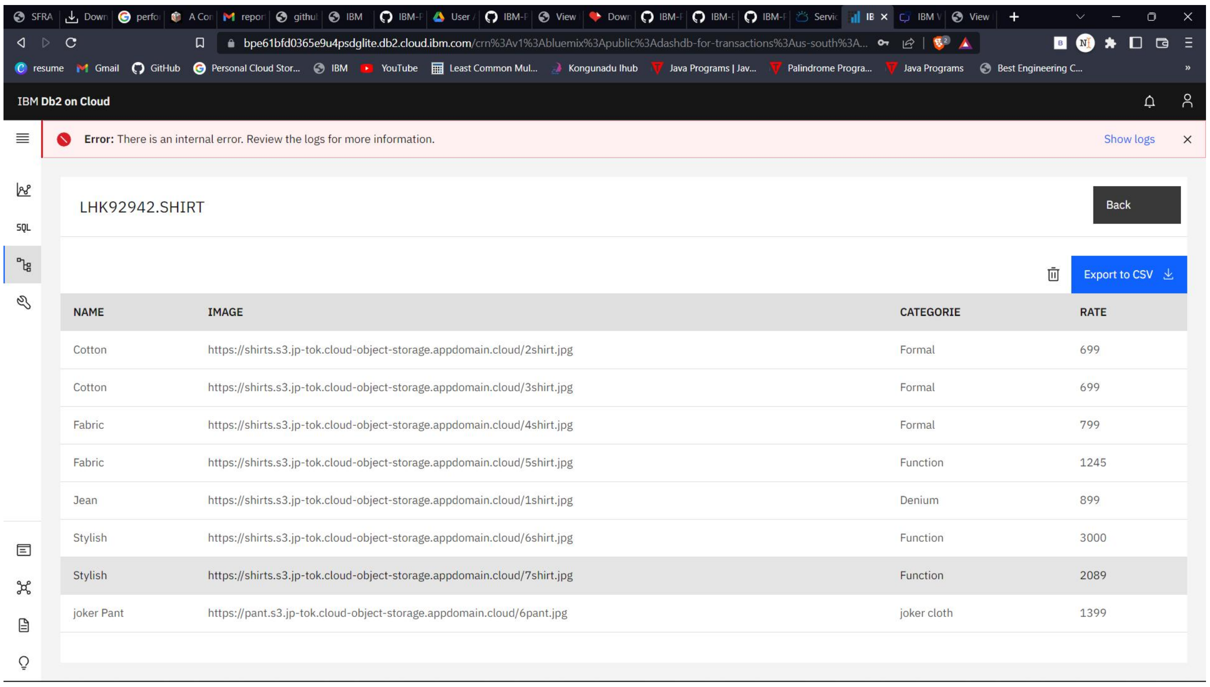


bpe61bfd0365e9u4psdglite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aus-south%3A...

Error: There is an internal error. Review the logs for more information.

LHK92942.SAREE

NAME	IMAGE	CATEGORIE	RATE
Bannaras	https://saree.s3.jp-tok.cloud-object-storage.appdomain.cloud/1saree.jpg	Silk	3000
Bannaras	https://saree.s3.jp-tok.cloud-object-storage.appdomain.cloud/7saree.jpg	Silk	4999
Bannaras	https://saree.s3.jp-tok.cloud-object-storage.appdomain.cloud/6saree.png	Silk	3500
Bannaras	https://saree.s3.jp-tok.cloud-object-storage.appdomain.cloud/5saree.png	Silk	4000
Bannaras	https://saree.s3.jp-tok.cloud-object-storage.appdomain.cloud/5saree.png	Silk	4000
Bannaras	https://saree.s3.jp-tok.cloud-object-storage.appdomain.cloud/4saree.png	Silk	2000
Bannaras	https://saree.s3.jp-tok.cloud-object-storage.appdomain.cloud/2saree.jpg	Silk	2799
Silk Banarash	https://saree.s3.jp-tok.cloud-object-storage.appdomain.cloud/3saree.png	Silk	3499


IBM Db2 on Cloud

Error: There is an internal error. Review the logs for more information.

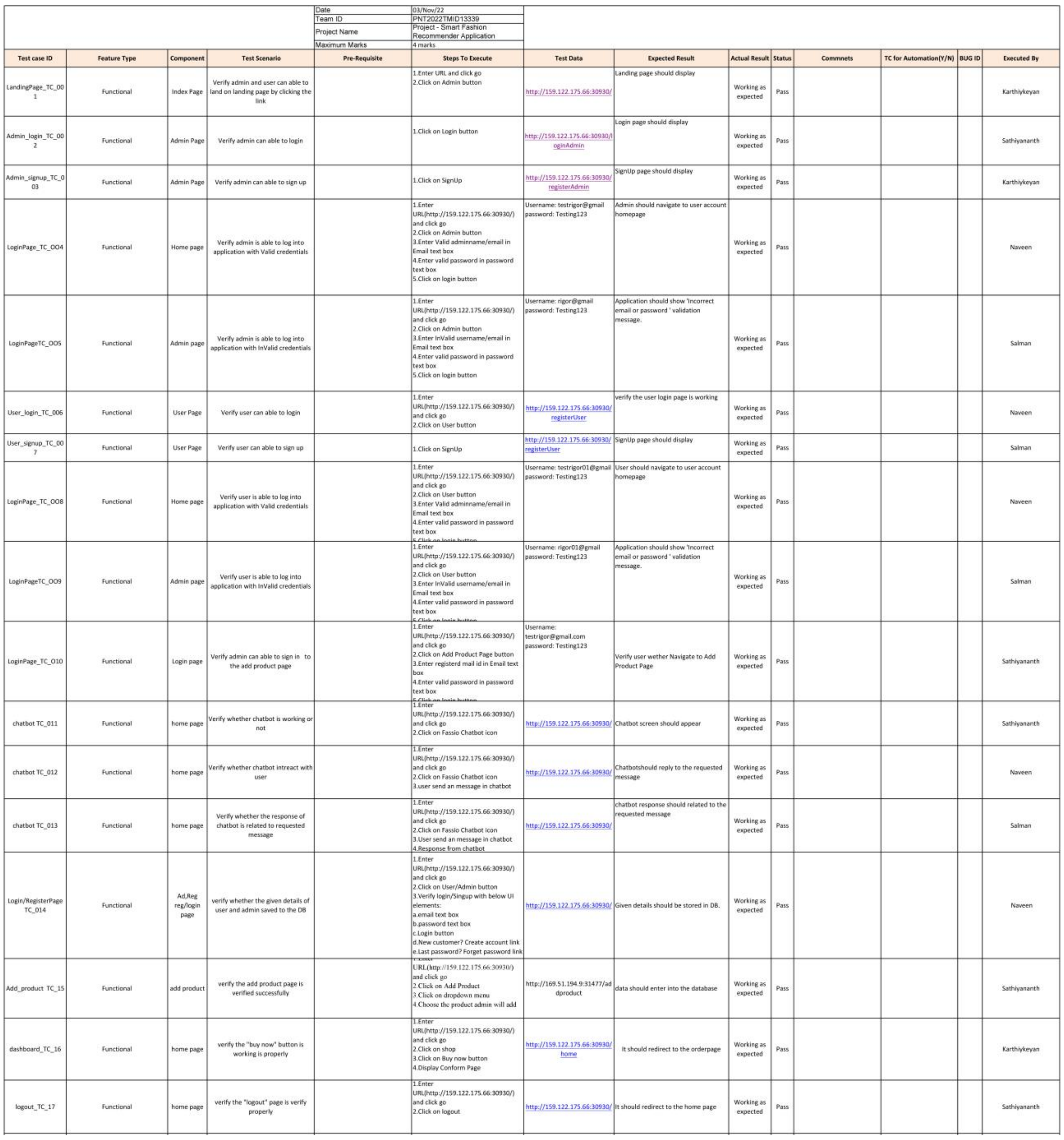
LHK92942.SHIRT

NAME	IMAGE	CATEGORIE	RATE
Cotton	https://shirts.s3.jp-tok.cloud-object-storage.appdomain.cloud/2shirt.jpg	Formal	699
Cotton	https://shirts.s3.jp-tok.cloud-object-storage.appdomain.cloud/3shirt.jpg	Formal	699
Fabric	https://shirts.s3.jp-tok.cloud-object-storage.appdomain.cloud/4shirt.jpg	Formal	799
Fabric	https://shirts.s3.jp-tok.cloud-object-storage.appdomain.cloud/5shirt.jpg	Function	1245
Jean	https://shirts.s3.jp-tok.cloud-object-storage.appdomain.cloud/1shirt.jpg	Denium	899
Stylish	https://shirts.s3.jp-tok.cloud-object-storage.appdomain.cloud/6shirt.jpg	Function	3000
Stylish	https://shirts.s3.jp-tok.cloud-object-storage.appdomain.cloud/7shirt.jpg	Function	2089
joker Pant	https://pant.s3.jp-tok.cloud-object-storage.appdomain.cloud/6pant.jpg	joker cloth	1399

CHAPTER 8

TESTING

8.1 TEST CASE



Date	03/Nov/22
Team ID	PNT2022TMID13339
Project Name	Project - Smart Fashion Recommender Application
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
LandingPage_TC_001	Functional	Index Page	Verify admin and user can able to land on landing page by clicking the link		1.Enter URL and click go 2.Click on Admin button	http://159.122.175.66:30930/	Landing page should display	Working as expected	Pass				Karthiykeyan
Admin_login_TC_002	Functional	Admin Page	Verify admin can able to login		1.Click on Login button	http://159.122.175.66:30930/loginAdmin	Login page should display	Working as expected	Pass				Sathiyananth
Admin_signup_TC_003	Functional	Admin Page	Verify admin can able to sign up		1.Click on SignUp	http://159.122.175.66:30930/registerAdmin	SignUp page should display	Working as expected	Pass				Karthiykeyan
LoginPage_TC_OO4	Functional	Home page	Verify admin is able to log into application with Valid credentials		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on Admin button 3.Enter Valid adminname/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: testrigor@gmail password: Testing123	Admin should navigate to user account homepage	Working as expected	Pass				Naveen
LoginPageTC_OO5	Functional	Admin page	Verify admin is able to log into application with InValid credentials		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on Admin button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: rigor@gmail password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				Salman
User_login_TC_006	Functional	User Page	Verify user can able to login		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on User button	http://159.122.175.66:30930/registerUser	verify the user login page is working	Working as expected	Pass				Naveen
User_signup_TC_007	Functional	User Page	Verify user can able to sign up		1.Click on SignUp	http://159.122.175.66:30930/registerUser	SignUp page should display	Working as expected	Pass				Salman
LoginPage_TC_OO8	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on User button 3.Enter Valid adminname/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: testrigor01@gmail password: Testing123	User should navigate to user account homepage	Working as expected	Pass				Naveen
LoginPageTC_OO9	Functional	Admin page	Verify user is able to log into application with InValid credentials		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on User button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: rigor01@gmail password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				Salman
LoginPage_TC_O10	Functional	Login page	Verify admin can able to sign in to the add product page		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on Add Product Page button 3.Enter registerd mail id in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: testrigor@gmail.com password: Testing123	Verify user wether Navigate to Add Product Page	Working as expected	Pass				Sathiyananth
chatbot TC_011	Functional	home page	Verify whether chatbot is working or not		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on Fassio Chatbot icon	http://159.122.175.66:30930/	Chatbot screen should appear	Working as expected	Pass				Sathiyananth
chatbot TC_012	Functional	home page	Verify whether chatbot intreact with user		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on Fassio Chatbot icon 3.user send an message in chatbot	http://159.122.175.66:30930/	Chatbotshould reply to the requested message	Working as expected	Pass				Naveen
chatbot TC_013	Functional	home page	Verify whether the response of chatbot is related to requested message		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on Fassio Chatbot icon 3.User send an message in chatbot 4.Response from chatbot	http://159.122.175.66:30930/	chatbot response should related to the requested message	Working as expected	Pass				Salman
Login/RegisterPage TC_014	Functional	Ad,Reg reg/login page	verify whether the given details of user and admin saved to the DB		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on User/Admin button 3.Verify login/Singup with below UI elements: a.email text box b.password text box c.Login button d.New customer? Create account link e.Last password? Forget password link	http://159.122.175.66:30930/	Given details should be stored in DB.	Working as expected	Pass				Naveen
Add_product TC_15	Functional	add product	verify the add product page is verified successfully		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on Add Product 3.Click on dropdown menu 4.Choose the product admin will add	http://169.51.194.9:31477/addproduct	data should enter into the database	Working as expected	Pass				Sathiyananth
dashboard_TC_16	Functional	home page	verify the "buy now" button is working is properly		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on shop 3.Click on Buy now button 4.Display Conform Page	http://159.122.175.66:30930/home	It should redirect to the orderpage	Working as expected	Pass				Karthiykeyan
logout_TC_17	Functional	home page	verify the "logout" page is verify properly		1.Enter URL(http://159.122.175.66:30930/) and click go 2.Click on logout	http://159.122.175.66:30930/	It should redirect to the home page	Working as expected	Pass				Sathiyananth

8.2 USER ACCEPTANCE TESTING

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [Smart Fashion Recommender Application] project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	5	1	4	20
Duplicate	2	0	2	0	4
External	3	2	0	7	12
Fixed	16	6	2	6	30
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	0	1
Won't Fix	0	5	0	0	5
Totals	31	18	7	17	73

3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

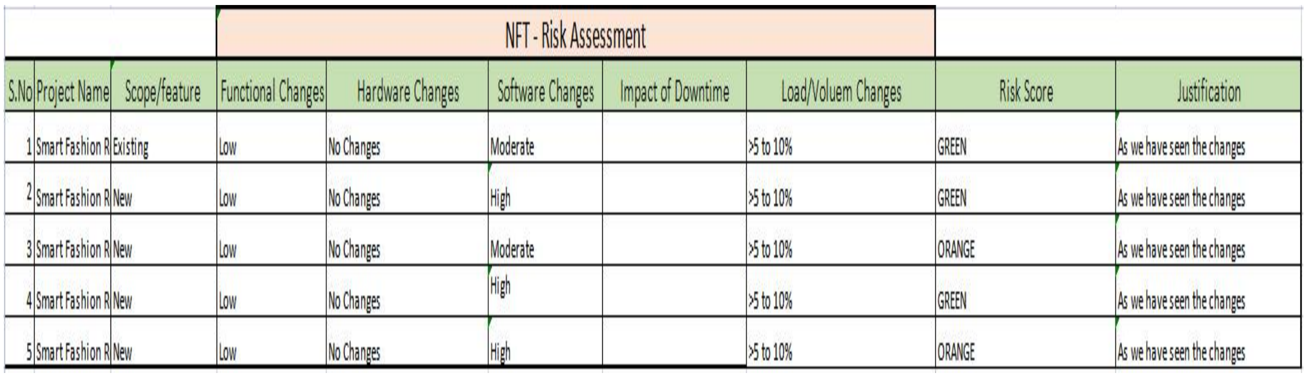
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	5	0	0	5
Client Application	10	0	0	10
Security	2	0	0	2
Outsource Shipping	1	0	0	2
Exception Reporting	3	0	0	3
Final Report Output	6	0	0	6
Version Control	1	0	0	1

CHAPTER 9

RESULTS

9.1 PERFORMANCE METRICS

NFT-Risk assessment:

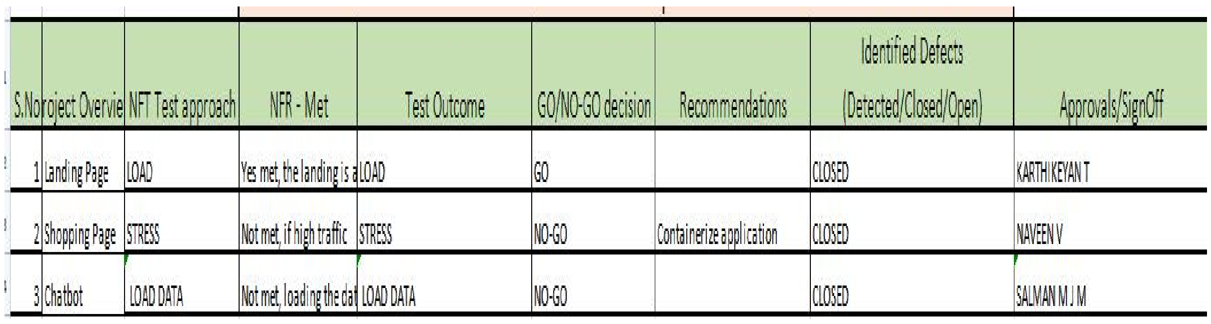


			NFT - Risk Assessment						
S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Voluem Changes	Risk Score	Justification
1	Smart Fashion R	Existing	Low	No Changes	Moderate		>5 to 10%	GREEN	As we have seen the changes
2	Smart Fashion R	New	Low	No Changes	High		>5 to 10%	GREEN	As we have seen the changes
3	Smart Fashion R	New	Low	No Changes	Moderate		>5 to 10%	ORANGE	As we have seen the changes
4	Smart Fashion R	New	Low	No Changes	High		>5 to 10%	GREEN	As we have seen the changes
5	Smart Fashion R	New	Low	No Changes	High		>5 to 10%	ORANGE	As we have seen the changes

NFT - Detailed Test Plan:

NFT - Detailed Test Plan				
S.No	Project Overview	NFT Test approach	Assumptions/Dependencies/Risks	Approvals/SignOff
1	Landing Page	LOAD	It may be not opening due to slow internet connectivity	SATHIYANANTH P , KARTHIKEYAN T
2	Shopping Page	STRESS	Due lot objects stored on it may slow down	NAVEEN V , SALMAN M J M
3	Chatbot	LOAD DATA	May be takes time to display the data	SATHIYANANTH P

End Of Test Report:



S.No	roject Overvie	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
1	Landing Page	LOAD	Yes met, the landing is a	LOAD	GO		CLOSED	KARTHIKEYAN T
2	Shopping Page	STRESS	Not met, if high traffic	STRESS	NO-GO	Containerize application	CLOSED	NAVEEN V
3	Chatbot	LOAD DATA	Not met, loading the dat	LOAD DATA	NO-GO		CLOSED	SALMAN M J M

CHAPTER 10

ADVANTAGES & DISADVANTAGES

10.1 ADVANTAGES

- The user will be able to directly communicate with the chatbot in this web application and tell it what they need to.
- Chatbot will let the consumer know whether they ordered anything through the application.
- Chatbot also assists in gathering user feedback.
- Additionally useful is keeping customer's and order information in the database.

10.2 DISADVANTAGES

- In an existing application, the user will travel to numerous screens that are different pages in order to access the desired product that was searched.
- The majority of fashion recommender applications just include a search capability.
- Most web applications lack a chatbot function, or if they do, it is usually used to assist customers with problems in the application.

CHAPTER 11

CONCLUSION

The Fashion Recommendation System can assist someone who lacks fashion sense by proposing the finest clothing combinations for their wardrobe. Because it is limited to the clothing items in the user's closet, the algorithm may not always offer the best potential outfit to wear to an occasion. Another factor is how much the era influences fashion. The system, on the other hand, excels in assisting users in developing a sense of fashion and can deliver the best recommendations based on the user's outfit. Because the system is created as a website, end users may quickly access and use it. The system's reach can be expanded by expanding its ability to recognise various garment designs and patterns, as well as increasing the number of occasions where this system can be used.

CHAPTER 12

FUTURE SCOPE

The fashion business is about to undergo unheard-of transformation. Recent advancements in the study of fashion suggestion systems will soon be advantageous to both customers and businesses. We outline some potential future lines of inquiry for recommendation systems for clothing in this section. Given the quick expansion of multimedia data, where visual information will play a crucial role. To build recommendation systems that are capable of thoroughly profiling users, further in-depth study in the applications of multi-model fusion and multi-task learning in the fashion industry is needed.

In addition, even though most studies on fashion recommender systems focus on similarity-based retrieval strategies; further study is needed to develop new features that will be crucial for future fashion recommender systems, such creating clothing. Additionally, the majority of the current fashion datasets lack outfit compatibility annotations or have a restricted number of them or are only annotated for a specific type of item. Due of the labour-intensive process of building datasets, the majority of academics do not make their datasets available to the general public for additional research. Therefore, the other potential route for future research may concentrate on developing automatic annotation techniques, creating vast, rich annotated data sets for specific job descriptions in fashion recommendation systems. Since it hasn't been investigated in nearly any of the studies that have been considered in this thesis, a full study of fashion recommender systems is also necessary from an ethical standpoint.

CHAPTER 13

APPENDIX

Source Code:

App.py

```
import os
import secrets
import isort
import autopep8
import black
from turtle import title
from unicodedata import category
from flask import Flask, render_template, request, redirect, url_for, session
from marshmallow import Schema, fields
import ibm_db
import bcrypt
import base64
import io
import mypy

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32731;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=lhk92942;PWD=Cht7cZes9VeaXQ5N","","")

# url_for('static', filename='style.css')

app = Flask(__name__)
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'
```

```
@app.route("/")
def index():
   return render_template('index.html',name='Home')

@app.route("/order")
def order():
   return render_template('order.html',name='Home')

@app.route("/registerUser",methods=['GET'])
def register():
   if request.method == 'POST':
      username = request.form['username']
      email = request.form['email']
      phoneno = request.form['phoneno']
      password = request.form['password']

      if not username or not email or not phoneno or not password:
         return render_template('registerUser.html',error='Please fill all fields')
      hash=bcrypt.hashpw(password.encode('utf-8'),bcrypt.gensalt())
      query = "SELECT * FROM user_detail WHERE email=? OR phoneno=?"
      stmt = ibm_db.prepare(conn, query)
      ibm_db.bind_param(stmt,1,email)
      ibm_db.bind_param(stmt,2,phoneno)
      ibm_db.execute(stmt)
      isUser = ibm_db.fetch_assoc(stmt)
      if not isUser:
         insert_sql = "INSERT INTO user_detail(username, email, phoneno, password)
VALUES (?,?,?,?)"
         prep_stmt = ibm_db.prepare(conn, insert_sql)
         ibm_db.bind_param(prep_stmt, 1, username)
         ibm_db.bind_param(prep_stmt, 2, email)
         ibm_db.bind_param(prep_stmt, 3, phoneno)
         ibm_db.bind_param(prep_stmt, 4, hash)
         ibm_db.execute(prep_stmt)
```

```
            return render_template('registerUser.html',success="You can login")
        else:
            return render_template('registerUser.html',error='Invalid Credentials')

    return render_template('registerUser.html',name='Home')

@app.route("/loginUser",methods=['GET','POST'])
def login():
        if request.method == 'POST':
            email = request.form['email']
            password = request.form['password']

            if not email or not password:
                return render_template('loginUser.html',error='Please fill all fields')
            query = "SELECT * FROM user_detail WHERE email=?"
            stmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(stmt,1,email)
            ibm_db.execute(stmt)
            isUser = ibm_db.fetch_assoc(stmt)
            print(isUser,password)

            if not isUser:
                return render_template('loginUser.html',error='Invalid Credentials')

            isPasswordMatch = bcrypt.checkpw(password.encode('utf-
8'),isUser['PASSWORD'].encode('utf-8'))

            if not isPasswordMatch:
                return render_template('loginUser.html',error='Invalid Credentials')

            session['email'] = isUser['EMAIL']
            return redirect(url_for('home'))
        return render_template('loginUser.html',name='Home')
```

```
@app.route("/registerAdmin",methods=['GET','POST'])
def registerAd():
   if request.method == 'POST':
      adminname = request.form['adminname']
      email = request.form['email']
      phoneno = request.form['phoneno']
      password = request.form['password']

      if not adminname or not email or not phoneno or not password:
         return render_template('registerAdmin.html',error='Please fill all fields')
      hash=bcrypt.hashpw(password.encode('utf-8'),bcrypt.gensalt())
      query = "SELECT * FROM admin_detail WHERE email=? OR phoneno=?"
      stmt = ibm_db.prepare(conn, query)
      ibm_db.bind_param(stmt,1,email)
      ibm_db.bind_param(stmt,2,phoneno)
      ibm_db.execute(stmt)
      isUser = ibm_db.fetch_assoc(stmt)

      if not isUser:
         insert_sql = "INSERT INTO admin_detail(adminname, email, phoneno, password)
VALUES (?,?,?,?)"
         prep_stmt = ibm_db.prepare(conn, insert_sql)
         ibm_db.bind_param(prep_stmt, 1, adminname)
         ibm_db.bind_param(prep_stmt, 2, email)
         ibm_db.bind_param(prep_stmt, 3, phoneno)
         ibm_db.bind_param(prep_stmt, 4, hash)
         ibm_db.execute(prep_stmt)
         return render_template('registerAdmin.html',success="You can login")
      else:
         return render_template('registerAdmin.html',error='Invalid Credentials')
```

```
    return render_template('registerAdmin.html',name='Home')

@app.route("/loginAdmin",methods=['GET','POST'])
def loginAd():
        if request.method == 'POST':
            email = request.form['email']
            password = request.form['password']

            if not email or not password:
                return render_template('loginAdmin.html',error='Please fill all fields')
            query = "SELECT * FROM admin_detail WHERE email=?"
            stmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(stmt,1,email)
            ibm_db.execute(stmt)
            isUser = ibm_db.fetch_assoc(stmt)
            print(isUser,password)

            if not isUser:
                return render_template('loginAdmin.html',error='Invalid Credentials')

            isPasswordMatch = bcrypt.checkpw(password.encode('utf-
8'),isUser['PASSWORD'].encode('utf-8'))

            if not isPasswordMatch:
                return render_template('loginAdmin.html',error='Invalid Credentials')

            session['email'] = isUser['EMAIL']
            return redirect(url_for('addproduct'))

        return render_template('loginAdmin.html',name='Home')


@app.route("/addProduct", methods=['GET','POST'])
def addproduct():
    if request.method == 'POST':
        types=request.form['cc']
        name = request.form['name']
        image = request.form['image']
        categorie = request.form['categorie']
        rate = request.form['rate']

        if types =='shirt':
            insert_sql = "INSERT INTO SHIRT(name, image, categorie, rate) VALUES
(?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)

        if types =='pant':
            insert_sql = "INSERT INTO PANT(name, image, categorie, rate) VALUES (?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)

        if types =='hat':
            insert_sql = "INSERT INTO HAT(name, image, categorie, rate) VALUES (?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)
        if types =='tops':
            insert_sql = "INSERT INTO TOPS(name, image, categorie, rate) VALUES (?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)

        if types =='saree':
            insert_sql = "INSERT INTO SAREE(name, image, categorie, rate) VALUES 
    (?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)

        if types =='leggings':
            insert_sql = "INSERT INTO LEGGINGS(name, image, categorie, rate) VALUES 
    (?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)

        if types =='tshirts':
            insert_sql = "INSERT INTO TSHIRTS (name, image, categorie, rate) VALUES 
    (?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)

        return render_template('addproduct.html',success="You can login")


            @app.route("/home",methods=['GET','POST'])
            def home():
        shirt_list=[]
        pant_list=[]
        hat_list=[]
        saree_list=[]
        tshirt_list=[]
        tops_list=[]
        leggings_list=[]

        #selecting_shirt
        sql = "SELECT * FROM SHIRT"
        stmt = ibm_db.exec_immediate(conn, sql)
        shirt = ibm_db.fetch_both(stmt)
        while shirt != False :
            shirt_list.append(shirt)
            shirt = ibm_db.fetch_both(stmt)
        print(shirt_list)

      #selecting_pant

        sql1="SELECT * FROM PANT"
        stmt1 = ibm_db.exec_immediate(conn, sql1)
        pant=ibm_db.fetch_both(stmt1)
        while pant != False :
            pant_list.append(pant)
            pant = ibm_db.fetch_both(stmt1)
        print(pant_list)

            #selecting_hat
        sql2="SELECT * FROM HAT"
        stmt2 = ibm_db.exec_immediate(conn, sql2)
        hat=ibm_db.fetch_both(stmt2)
        while hat != False :
            hat_list.append(hat)
            hat = ibm_db.fetch_both(stmt2)
        print(hat_list)

        #selecting_saree
        sql3="SELECT * FROM SAREE"
        stmt3 = ibm_db.exec_immediate(conn, sql3)
        saree=ibm_db.fetch_both(stmt3)
        while saree != False :
            saree_list.append(saree)
            saree = ibm_db.fetch_both(stmt3)
        print(saree_list)

        sql3="SELECT * FROM TSHIRTS"
        stmt3 = ibm_db.exec_immediate(conn, sql3)
        tshirt=ibm_db.fetch_both(stmt3)
        while tshirt != False :
            tshirt_list.append(tshirt)
            tshirt = ibm_db.fetch_both(stmt3)
        print(tshirt_list)

        sql3="SELECT * FROM TOPS"
        stmt3 = ibm_db.exec_immediate(conn, sql3)
        tops=ibm_db.fetch_both(stmt3)
        while tops != False :
            tops_list.append(tops)
            tops = ibm_db.fetch_both(stmt3)
        print(tops_list)

        sql3="SELECT * FROM LEGGINGS"
        stmt3 = ibm_db.exec_immediate(conn, sql3)
        leggings=ibm_db.fetch_both(stmt3)
        while leggings != False :
            leggings_list.append(leggings)
            leggings = ibm_db.fetch_both(stmt3)
        print(leggings_list)

        #returning to HTML
        return                                    render_template('home.html',dictionary=
            shirt_list,pants=pant_list,hats=hat_list,sarees=saree_list,tshirts=tshirt_list,topss=tops_list,legg
            ings=leggings_list)

            @app.route("/data")
            def display():
        shirt_list=[]
        pant_list=[]
        hat_list=[]
        saree_list=[]
        tshirt_list=[]
        tops_list=[]
        leggings_list=[]

        #selecting_shirt
        sql = "SELECT * FROM SHIRT"
        stmt = ibm_db.exec_immediate(conn, sql)
        shirt = ibm_db.fetch_both(stmt)
        while shirt != False :
            shirt_list.append(shirt)
            shirt = ibm_db.fetch_both(stmt)
        print(shirt_list)

      #selecting_pant

        sql1="SELECT * FROM PANT"
        stmt1 = ibm_db.exec_immediate(conn, sql1)
        pant=ibm_db.fetch_both(stmt1)
        while pant != False :
            pant_list.append(pant)
            pant = ibm_db.fetch_both(stmt1)
        print(pant_list)

            #selecting_watch
        sql2="SELECT * FROM HAT"
        stmt2 = ibm_db.exec_immediate(conn, sql2)
        hat=ibm_db.fetch_both(stmt2)
        while hat != False :
            hat_list.append(hat)
            hat = ibm_db.fetch_both(stmt2)
        print(hat_list)

        #selecting_rings
        sql3="SELECT * FROM SAREE"
        stmt3 = ibm_db.exec_immediate(conn, sql3)
        saree=ibm_db.fetch_both(stmt3)
        while saree != False :
            saree_list.append(saree)
            saree = ibm_db.fetch_both(stmt3)
        print(saree_list)
        sql3="SELECT * FROM TSHIRTS"
        stmt3 = ibm_db.exec_immediate(conn, sql3)
        tshirt=ibm_db.fetch_both(stmt3)
        while tshirt != False :
            tshirt_list.append(tshirt)
```

```
        tshirt = ibm_db.fetch_both(stmt3)
    print(tshirt_list)
    sql3="SELECT * FROM TOPS"
    stmt3 = ibm_db.exec_immediate(conn, sql3)
    tops=ibm_db.fetch_both(stmt3)
    while tops != False :
        tops_list.append(tops)
        tops = ibm_db.fetch_both(stmt3)
    print(tops_list)
    sql3="SELECT * FROM LEGGINGS"
    stmt3 = ibm_db.exec_immediate(conn, sql3)
    leggings=ibm_db.fetch_both(stmt3)
    while leggings != False :
        leggings_list.append(leggings)
        leggings = ibm_db.fetch_both(stmt3)
    print(leggings_list)
    #returning to HTML
    return                                   render_template('home.html',dictionary=
shirt_list,pants=pant_list,hats=hat_list,sarees=saree_list,tshirts=tshirt_list,topss=tops_list,legg
ings=leggings_list)

@app.route("/orderplaced",methods=['GET','POST'])
def dis():
    if request.method == 'POST':
        pname=request.form['name']
        img=request.form['image']
        rate=request.form['rate']
        categorie=request.form['categorie']
    return render_template('order.html',pname=pname,img=img,rate=rate,categorie=categorie)

@app.route("/complete",methods=['GET','POST'])

def orderdisplay():
    if request.method == 'POST':
        name = request.form['order_name']
        image = request.form['order_image']
        rate = request.form['order_rate']
        categorie = request.form['order_categorie']
        insert_sql = "INSERT INTO DISPLAYORDER(name, image,rate, categorie) VALUES
(?,?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, name)
        ibm_db.bind_param(prep_stmt, 2, image)
        ibm_db.bind_param(prep_stmt, 3, rate)
        ibm_db.bind_param(prep_stmt, 4, categorie)
        ibm_db.execute(prep_stmt)
    return render_template('success.html',success="You can login")

@app.route("/displayorder")
def displayorder():
    details_list=[]
    sql = "SELECT * FROM DISPLAYORDER"
    stmt = ibm_db.exec_immediate(conn, sql)
    detail = ibm_db.fetch_both(stmt)
    while detail != False :
        details_list.append(detail)
        detail = ibm_db.fetch_both(stmt)
    print(details_list)
    return render_template('displayorder.html',details=details_list)

@app.route('/logout')
def logout():
        session.pop('email', None)
        return redirect(url_for('index'))
if __name__ == '__main__':
        port=int(os.environ.get('PORT',5000))
        app.run(port=port,host='0.0.0.0')
        app.debug=True
```

13.2GitHub & Project Demo Link:

GitHub Link: https://github.com/IBM-EPBL/IBM-Project-2709-1658481645

Project Demo Link:

APP NAME: FASSIO

PROJECT NAME: SMART FASHION RECOMMENDER APPLICATION.

PROJECT LINK: http://159.122.175.66:31180/

DEMO LINK: https://youtu.be/58TdeVMxBLM