A Ladies shopping search engine

A Paper Submitted to the Graduate Faculty of The Islamia University of Bahawalpur (BWN Campus) of Computer Science and IT

By

Usama Hassan

Muhammad Danial

Khizar Ijaz

In Partial Fulfillment of the Requirements for the Degree of BSCS & IT

Major Program:

CS & IT

Jan 2020

Bahawalnagar, Punjab

**The Islamia University of Bahawalpur Bahawalnagar Campus**

**Title**

Wearit.

**By**

Students of IUB( 2016-2020) Spring

The Supervisory Committee certifies that this ***disquisition*** complies with The Islamia University’s regulations and meets the accepted standards for the degree of

**BSCS & IT**

EVALUATION COMMITTEE:

|  |
| --- |
| Dr. Tenvir Ali |
| Mr. Muhammad Majid |
| Mr. Raza ul Haq |
| Mr. Adnan Akhtar |

Approved by HOD:

|  |  |  |
| --- | --- | --- |
| 1/1/2020 |  | Mr. Amir Jamshed |
| Date |  | Signature |

# ABSTRACT

Electronic commerce, also known as ecommerce is a type of industry where buying and selling of a product is conducted over electronic systems such as the internet.

The purpose of this application is to bring all the biggest local brands of Pakistan in the industry of clothing at one platform. As all the brands have shifted to ecommerce, so making a purchase for a brand conscious internet user will become significantly easy when all the brands and their products along with the price comparisons shown at a single webpage from which he/she will be redirected to buy page upon clicking on Buy Now option and that is what wearit. doing. Since data, which is to be shown after some calculations, is scattered at different ecommerce stores, therefore, a database and crawlers(to populate database) needs to be established to compare and display the result of a query faster. Summing up, the user will input their query in search box and will get products from all the local brands from where he/she can make purchasing among multiple brands.

# ACKNOWLEDGMENTS

I would like to express my deep sense of gratitude and convey thanks to everyone who helped me and supported during the completion of this project. First, I would like to express a deep sense of gratitude to Mr. Amir Jamshed for helping, guiding, and supporting me throughout my Bachelor’s degree. I also convey thanks to my all committee members for helping me from time to time and for being on my committee. I acknowledge my department for providing the courses and a great atmosphere that helped complete different chapters of this project. I especially thank my supervisor, Dr. Tenvir Ali, for supporting me and my concepts and for allowing me to do something the way I liked. Last but not least, I would like to thank my family members for their constant and unrelenting support towards my education and for their impartial love for me. I would also like to thank my friends, without whom this project would have been impossible.

Table of Contents

[ABSTRACT 3](#_Toc28710579)

[ACKNOWLEDGMENTS 4](#_Toc28710580)

[LIST OF TABLES 7](#_Toc28710581)

[LIST OF FIGURES 1](#_Toc28710582)

[CHAPTER 1. INTRODUCTION 2](#_Toc28710583)

[1.1. Motivation 3](#_Toc28710584)

[1.2. Aim of the Software 3](#_Toc28710585)

[1.3. Literature Review 4](#_Toc28710586)

[1.4. Paper Organization 8](#_Toc28710587)

[CHAPTER 2. OBJECTIVES 9](#_Toc28710588)

[2.1. Requirements Analysis 9](#_Toc28710589)

[2.1.1. Product Perspective 11](#_Toc28710590)

[2.1.1.1. User Interface 11](#_Toc28710591)

[2.1.1.2. Hardware Interface 11](#_Toc28710592)

[2.1.1.3. Software Interface 12](#_Toc28710593)

[2.1.2. Product Function 12](#_Toc28710594)

[2.1.3. User Characteristics 13](#_Toc28710595)

[2.1.4. Constraints 14](#_Toc28710596)

[2.1.5. Assumptions and Dependencies 14](#_Toc28710597)

[2.1.6. Specific Requirements 14](#_Toc28710598)

[2.1.6.1. Functional Requirements 15](#_Toc28710599)

[2.1.6.2. Performance Requirements 15](#_Toc28710600)

[2.1.7. Design Constraint 16](#_Toc28710601)

[CHAPTER 3. IMPLEMENTATION 17](#_Toc28710602)

[3.1. Detailed Scope 17](#_Toc28710603)

[3.2. Static Decomposition and Dependency Description 18](#_Toc28710604)

[3.2.1. High-Level Use Case Diagram 18](#_Toc28710605)

[3.2.2. Activity Diagram 22](#_Toc28710606)

[3.3. The Ladies shopping search engine’s Implementation 24](#_Toc28710607)

[3.4. The Ladies shopping search engine’s Interface 32](#_Toc28710608)

[CHAPTER 4. TESTING 38](#_Toc28710609)

[4.1. Methodology 38](#_Toc28710610)

[4.2. Interface Testing 39](#_Toc28710611)

[4.3. Test Cases 40](#_Toc28710612)

[4.4. Results 44](#_Toc28710613)

[CHAPTER 5. CONCLUSION/FUTURE WORK 45](#_Toc28710614)

[5.1. Conclusion 45](#_Toc28710615)

[5.2. Future Work 45](#_Toc28710616)

[REFERENCES 47](#_Toc28710617)

# LIST OF TABLES

Table Page

1. [**Table 1.** List of Functional Requirements 40](#_Toc27912179)
2. [**Table 2.** List of Test Cases. 41](#_Toc27912180)
3. [**Table 3.** List of Test-Case Results. 44](#_Toc27912181)

# LIST OF FIGURES

1. [**Figure 1.** Shopping search engine System Use-Case Diagram: User. 19](#_Toc27913525)
2. [**Figure 2.** Activity Diagram for Shopping search engine. 23](#_Toc27913526)
3. [**Figure 3.** The Database Package. 25](#_Toc27913527)
4. [**Figure 4.** The Database Package. 26](#_Toc27913528)
5. [**Figure 5.** Implementation of the Default Page for the Wearit. 27](#_Toc27913529)
6. [**Figure 6.** View.py Code Snippet 28](file:///C:\Users\Danial%20Temur\Desktop\Online-Shopping-Project-Documentation-Template.docx#_Toc27913530)
7. [**Figure 7**. View.py Code Snippet(cont.) 29](file:///C:\Users\Danial%20Temur\Desktop\Online-Shopping-Project-Documentation-Template.docx#_Toc27913531)
8. [**Figure 8.** View.py Code Snippet(cont.) 30](file:///C:\Users\Danial%20Temur\Desktop\Online-Shopping-Project-Documentation-Template.docx#_Toc27913532)
9. [**Figure 9.** View.py Code Snippet(cont.) 31](file:///C:\Users\Danial%20Temur\Desktop\Online-Shopping-Project-Documentation-Template.docx#_Toc27913533)
10. [**Figure 10.** View.py Code Snippet(cont.) 32](file:///C:\Users\Danial%20Temur\Desktop\Online-Shopping-Project-Documentation-Template.docx#_Toc27913534)
11. [**Figure 11.** Screenshot of the Home Page. 33](#_Toc27913535)
12. [**Figure 12.** Screenshot of the Electronics Category 34](#_Toc27913536)
13. [**Figure 13.** Screenshot of the Electronics Category. 35](#_Toc27913537)
14. [**Figure 14.** Screenshot of the View Cart. 36](#_Toc27913538)
15. [**Figure 15.** Screenshot of the Database 37](#_Toc27913539)

# CHAPTER 1. INTRODUCTION

It is known globally that, in today’s market, it is extremely difficult to start a new small- scale business and its sustenance with competition from the well-established and settled/brand owners. Most often, even if the quality of the product is really good, due to a lack of advertisement or business at the small scale, it just becomes another face in the sea, and the product does not reach a larger group of customers. In fast paced life of today when everyone is squeezed for time, the majority of people are finicky when it comes to doing physical shopping. Logistically, a consumer finds a product more interesting and attractive when they find it on the website of a retailer directly and are able to see item’s details online.[1][2] The customers of today are not only attracted because online shopping is very convenient, but also because they have broader selections, highly competitive prices, better information about the product (including people’s reviews) and extremely simplified navigation for searching regarding the product.

Moreover, business owners often offer online shopping options at low rates because the overhead expenses in opening and running a physical store are higher. Further, with online shopping, their products have access to a worldwide market, which increases the number of customers from different ethnic groups, adds customer value, and overall sustainable in the marketing.[4]

Online shopping search engine, such as shopsy.pk and pakistanistores, have gained huge popularity over the years because one can buy almost everything at these stores with competitive price comparisons. These web stores also give an opportunity to a lot of small-scale companies and manufactures to reach the global market and to directly sell their products to people before their product can reach the shelves of a physical store.

# Motivation

The motivation for designing this shopping search engine came because everyone love online shopping rather than spending lot of time at physical markets only if the product is from a recognized brand who doesn’t compromise on quality. Moreover, We value recent learning about the Python and web development programming languages as well as seeing how powerful and dynamic they are when it comes to web designing and applications. Apart from helping computer science students understand the concepts of web-application designing, it would be very easy to incorporate the idea of using programming techniques from the available visuals to understand how a piece of code appears on a user interface. The languages used to build this application are JavaScript, HTML,CSS, and Python because I found them to be extremely useful while working on the technologies at international marketplace like fiverr,upwork.

# Aim of the Software

Already developed systems doesn’t compare prices of the ecommerce sites’ of brands which are on the top trend in Pakistan. Pakistan’s biggest brands which are like Khaadi,Juniad Jamshaid etc. have built their individual websites to sell their products online but there is not a single platform which compares their prices and present the best choices among all of them.

So this system will do that by getting data from each brand’s sites, make calculations to that data and present it on a single page within specified price.

This application is very versatile and can be enhanced by adding more functions and modified graphics for use with commercial purposes.

# Literature Review

A comparison shopping website, sometimes called a price comparison website, price analysis tool, comparison shopping agent, shopbot or comparison shopping engine, is a vertical search engine that shoppers use to filter and compare products based on price, features, reviews and other criteria. Most comparison shopping sites aggregate product listings from many different retailers but do not directly sell products themselves, instead earning money from affiliate marketing agreements. In the United Kingdom, these services made between £780m and £950m in revenue in 2005.Hence, E-commerce accounted for an 18.2 percent share of total business turnover in the United Kingdom in 2012. Online sales already account for 13% of the total UK economy, and its expected to increase to 15% by 2017. There is a huge contribution of comparison shopping websites in the expansion of current E-commerce industry.

The first widely recognized comparison-shopping agent was BargainFinder, developed by Andersen Consulting (now Accenture). The team, led by researcher Bruce Krulwich, created BargainFinder in 1995 as an experiment and published it on-line without advance warning to the e-commerce sites being compared. The first commercial shopping agent, called Jango, was produced by Netbot, a Seattle startup company founded by University of Washington professors Oren Etzioni and Daniel S. Weld; Netbot was acquired by the Excite portal in late 1997. Junglee, a Bay-area startup, also pioneered comparison shopping technology and was soon acquired by Amazon.com. Other early comparison shopping agents included pricewatch.com and killerapp.com. NexTag another entry into comparison shopping was named Times magazine world top 50 website in 2008.

Around 2010, the price comparison websites found their way to emerging markets. Especially South-East Asia has been a place for many new comparison websites. It started in 2010 with CompareXpress in Singapore, and in the following years companies like Baoxian (China), Jirnexu (Malaysia), and AskHanuman (Thailand) followed.

Meanwhile, in developed markets, Google was accused of promoting Froogle and its replacement, the paid-placement-only Google Shopping, over competitors in its search results, driving down traffic to other sites and driving some out of business. The European Commission began an investigation in 2010, which concluded in July 2017 with a €2.42 billion fine against the parent company Alphabet, and an order to change its practices within 90 days.

Price comparison sites can collect data directly from merchants. Retailers who want to list their products on the website then supply their own lists of products and prices, and these are matched against the original database. This is done by a mixture of information extraction, fuzzy logic and human labor.

Comparison sites can also collect data through a data feed file. Merchants provide information electronically in a set format. This data is then imported by the comparison website. Some third party businesses are providing consolidation of data feeds so that comparison sites do not have to import from many different merchants. Affiliate networks aggregate data feeds from many merchants and provide them to the price comparison sites. Many of the popular shopping websites provide direct affiliation to the customer who wants to become affiliate partner. They provide their own API to the affiliate partner to show their products with specifications to the affiliate partner's website. This enables price comparison sites to monetize the products contained in the feeds by earning commissions on click through traffic. Other price comparison sites have deals with merchants and aggregate feeds using their own technology.

In recent years, many off the shelf software solutions have been developed that allow website owners to take price comparison websites' inventory data to place retailer prices on their blog or content the only website. In return, the content website owners receive a small share of the revenue earned by the price comparison website. This is often referred to as the revenue share business model.

Another approach is to crawl the web for prices. This means the comparison service scans retail web pages to retrieve the prices, instead of relying on the retailers to supply them. This method is also sometimes called 'scraping' information. Some, mostly smaller, independent sites solely use this method, to get prices directly from the websites that it is using for the comparison.

Yet another approach is to collect data is through crowdsourcing. This lets the price comparison engine collect data from almost any source without the complexities of building a crawler or the logistics of setting up data feeds at the expense of lower coverage comprehensiveness. Sites that use this method rely on visitors contributing pricing data. Unlike discussion forums, which also collect visitor input, price comparison sites that use this method combine data with related inputs and add it to the main database though collaborative filtering, artificial intelligence, or human labor. Data contributors may be rewarded for the effort through prizes, cash, or other social incentives.

However, some combination of these two approaches is most frequently used. Some search engines are starting to blend information from standard feeds with information from sites where product stock-keeping units (SKUs) are unavailable.

Empirical projects that assessed the functionality and performance of page-wise SSC engines (AKA bots) exist. These studies demonstrate that no best or parsimonious shopping bot exists with respect to price advantage.[3]

The application proposed in this paper is more focused on developing a simple, yet complete. This application performs all the basic functions that the above-mentioned applications do, such as searching an item and moving to buy now page or adding it to the shopping cart, checkout of the item, etc. Other functions that can be added to this application are proposed in the future work, and they would be necessary under a more complete

and complex design.

# Paper Organization

The rest of the document is divided into three parts: Objectives, Implementation, and Testing. The Objectives chapter lists the need for building the system. It provides use cases to help the business and technical users with their understanding. It also gives a detailed explanation for each use case to help with design and implementation, and outlines the constraints regarding the software. The Implementation chapter contains the detailed design of the system, including the Class Diagram, Activity Diagram, and Component Diagram. This chapter also includes a detailed explanation for each component as well as the interaction of the class and its components with each other when carrying out certain tasks, besides software’s mock screen shots.

# CHAPTER 2. OBJECTIVES

All the steps required in the software-analysis process related to this project (product function, user characteristics, functional and nonfunctional requirements, constraints, assumptions, and dependencies for the online shopping cart application) are described in the following sections.

# Requirements Analysis

The requirements analysis and gathering processes are critical for the success of any software engineering project. Requirements analysis in software engineering is a process that determines the tasks that are required to determine the needs and conditions to design a new product or to make modifications in any existing product/application. This process considers all the stakeholders’ conflicting requirements, and analyzes the documentation and validation of the system. The requirements should be actionable, measurable, testable, and related to the defined needs of the system design. From the software-engineering perspective, requirements analysis is a three-step process.

1. Requirements Elicitation: Elicitation of requirements, also known as requirements gathering, includes the task of identifying various requirement types from stakeholders or from project documentation.
2. Requirements Analysis: Analysis of requirements determines if the gathered requirements are clear, complete, and consistent. The analysis also handles any ambiguous requirements that do not clearly state what needs to be implemented, which could create a loss of resources and time if identified later in the development or testing phase.

Requirement analysis requires identifying the stakeholders and taking their needs into account to help them understand the implications of designing the new system, along with

what modules are worth implementing and which ones are more cost efficient, and then to create a software-requirement specification document. To clearly elicit the stakeholders’ requirements, different processes, such as developing a scenario or user stories, and identifying the use case which is being used for the project, can be utilized.

Stakeholder analysis says that, to clearly gather the requirements of the project, analysts first need to identify the stakeholders. Stakeholders are people or organizations that have a valid interest or use in the system. The steps to identify the stakeholders are as follows:

* + Anyone who operates the system.
  + Anyone who benefits from the system
  + Anyone who is directly or indirectly involved in purchasing the system
  + People or organizations opposed to the system
  + Organizations responsible for the system design
  + Organizations that regulate the financial or safety aspects of the system

Once the stakeholders are successfully identified, interviews are conducted through different processes; the needs and requirements of the system are identified, and a requirements specification document is prepared. The document is then discussed with the major stakeholders to identify any ambiguity with the requirements and understanding of the system.

1. Requirements Documentation: This step involves documenting the requirements in various forms, including summary lists, natural language documents, visual documents, use cases, user stories, or process specifications. A requirement specification document is categorized in different ways according to the stakeholders’ need, helping to create a clear contract between development and business. The following sections include the different

categories of requirements specification document that are essential for designing this application: the functional requirements, constraints, system requirements, etc.

# Product Perspective

The online shopping search engine is a web-based system. It can be accessed using Internet Explorer 8.0 and above, Mozilla Firefox 2.0, and Google Chrome.

# User Interface

In the system, user interacts with the application through a graphical user interface. The inputs to the system are the Search and Filter criteria provided by the user and a new review written by the user. Also, the output is in the form of Repeater and grid views which present the users with list of Products available. The users can view complete specification, view Images and reviews by other users (if any).

Interface is described, in detail, in the External Interface Requirement section of Chapter 3.

# Hardware Interface

This application shall provide minimum hardware requirements. The following hardware configurations are required for a PC using the ladies shopping search engine:

* + Intel core 2 duo
  + 40 GB of free hard-drive space
  + 1 GB of RAM

# Software Interface

This section lists the requirements that are needed to run the system efficiently. The operating system needed for the system to run effectively, the interface to run the application, the libraries for running Django web applications, the integrated development environment to develop the application, and the third-party tool used for editing purposes are as follows:

1. Operating System: Windows (Windows 7+)
2. Web Brower: Internet Explorer (8.0 and above), Mozilla Firefox (3.0 and above), or Google Chrome
3. Language/Tool: Python 3.6, MongoDB
4. Integrated Development Environment: Atom, Sublime, PyCharm Editors

# Product Function

The shopping search engine would have the following basic functions:

1. In search bar user can enter a keyword which will then upon pressing enter produces some results
2. The data will be populated into thumbnail forms
3. Data must gathered from top brand’s websites of ladies dresses of Pakistan
4. Upon clicking of each thumbnail, the user must redirected to the webpage of brand’s site where he/she can do purchasing
5. If user entered a keyword which is not present in any of the website, then a proper message should be displayed

# User Characteristics

The users of the application, based on their roles are customers (users) These users are identified based on their experience and technical expertise.

1. **Users:** The users of this application are all customers who would shop to test the application. These users are anyone with shopping experience and the know-how to browse through a price comparison application. They must have basic understandings about computers and the internet. The users should be able to perform the following functions using this system:
   * View, browse, and select a product on the home page.

# Constraints

1. Hardware Limitations
2. Accessibility: Initially, the software should be available as a wab application

# Assumptions and Dependencies

The assumptions and dependencies are as follows:

1. The system is dependent on the availability of a MongoDB Server to run.
2. We assume that system users adhere to the system’s minimum software and hardware requirements.

# Specific Requirements

This section contains details about all the software that is required for designers to create a system to satisfy the users’ requirements and for testers to test the given requirements. This section contains the interface description of each GUI for the different system users. These sections also give descriptions about all the system inputs, all the functions performed by the system, and all the system output (responses).

# Functional Requirements

This section contains the requirements for this system. Functional requirements are those which describes about what the system should do. Functional requirements are listed below.

1. A webpage containing search bar and logo of wearit.pk
2. In search bar user can enter a keyword which will then upon pressing enter produces some results
3. The data will be populated into thumbnail forms
4. Data must gathered from top brand’s websites of ladies dresses of Pakistan
5. Upon clicking of each thumbnail, the user must redirected to the webpage of brand’s official site where he/she can do purchasing
6. If user entered a keyword which is not present in any of the website, then a proper message should be displayed.
7. Search and Filter criteria provide the user all the products within specified price range

.

# Performance Requirements

This section lists the performance requirements expected from this comparison application.

1. **PR01:** The users shall be able to products off all brands in fewer than 5 seconds.
2. **PR02:** The users shall be able to view information about the products in fewer than 5 seconds.
3. **PR03:** The users shall be able to reach the Buy Now page of the official brand in just one click
4. **PR04:** The navigation between pages shall take fewer than 5 seconds.

# Design Constraint

This section lists the design requirements for the shopping search engine.

1. **DC01:** The user interface (UI) must have specific fonts and font sizes. The system shall match the fonts and font sizes used for all the pages of the application.

# CHAPTER 3. IMPLEMENTATION

This chapter includes the detailed design used to build the shopping search engine. The system's design is used to create the functions and operations of the gathered requirements in detail, including screen layouts, business rules, process diagrams, and other documentation. The output of this chapter describes the new system which is defined as a collection of modules and subsystems. This design stage takes the initial input requirements that were identified in the approved requirements specification document. For each requirement, there is a set of one or more design elements that are produced using the different prototypes.

These design elements describe the desired software features, in detail, including functional hierarchy diagrams, screen layouts, activity diagrams, and class diagrams. The intention of these diagrams is to describe the software in detail so that the system can develop the application with less additional design input. The system’s mock screen shots are shown later in this chapter.

# Detailed Scope

This project is supposed to be delivered in two modules, with each module being an add-on to the project that makes it more usable and acceptable.

1. In the first module, front end work is done like.
   * Browse/ Search brands products on the home page
   * View information about a product like title, price, image, brand name etc.
   * Add an item to the shopping cart by redirecting to brand site.
   * Continue shopping.
2. In second module, data is gathered from brand’s site by using Scrapy (A python framework to scrape data.) and then uploaded that data to MongoDB. The scraped data involved following parameters
   * Image of the product
   * Title
   * Price
   * Brand Name
   * Url of Buy Now page
   * Category of the product

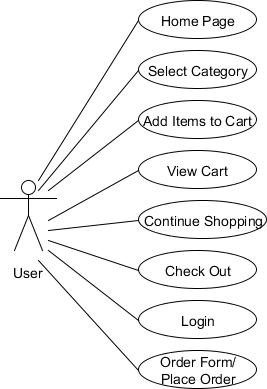
# Static Decomposition and Dependency Description

This section contains the system use-case diagram for the shopping search engine and also has a detailed explanation for each use case in the system.

# High-Level Use Case Diagram

The system’s use case shows the user a detailed view of the system and how the actors would interact with each other and with the system. The explanation for each use case is then provided below the system use case for the user (Figure 1), helping the user to understand who are the actors areas as well as giving the description for each use case along with its pre- and post-conditions that should be satisfied once the use case is implemented in the software.

Figure 1 demonstrates the use case for users where they have access to the shopping search engine



**Figure 1.** Shopping search engine System Use-Case Diagram: User.

Below are the different use cases in the system, the use case, and the actors associated with each use case. The description is used for a novice user to better understand the workings of the system and the pre-conditions that should be satisfied before invoking each use case.

1. **Use-Case Number:** US-001

**Application:** Shopping search engine

**Use-Case Name:** Home page

**Use-Case Description:** This use case lets the user view the home page

**Primary Actor:** User

**Precondition:** Run the application.

**Post-condition:** The user successfully runs the application and is able to view the home page with the different products.

**Basic Flow:**

* + Run the application
  + View the home page
  + Browse the products

1. **Use-case number:** US-002

**Application:** Shopping search engine

**Use-case name:** Select category

**Use-case description:** This use case details the category for selecting a process where the user can browse through the different products and select one products to view items.

**Primary actor:** User

**Precondition:** The user successfully runs the application to view the home page and browse different categories.

**Post-condition:** The user successfully selects a product to view items in a particular brand.

**Basic Flow:**

* + Run the application
  + View the home page
  + Browse the query
  + Select a product
  + Reach the Buy Now page

1. **Use-Case Number:** US-003

**Application:** Shopping search engine

**Use-Case Name:** Add Item

**Use-Case Description:** This details the item-adding process for the system to access it. The user should be able to add items to the cart and view information about the item by going to brands’ official page.

**Primary Actor:** User

**Precondition:** The items available for shopping are available for the user to browse.

**Post-condition:** The user successfully adds items to the cart.

**Basic Flow:**

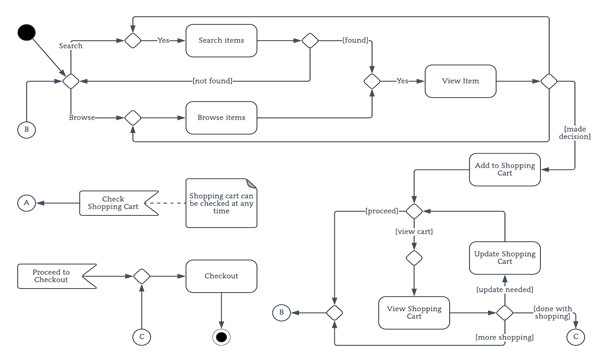
* + View the home page
  + Browse the categories
  + Select a category
  + Browse through the items
  + Add item to the cart.

# Activity Diagram

This section lists the activity diagram and describes the flow of activities in the system. A detailed description is then given after the figure for each activity. Figure 2 provides the overview of the activity of ladies shopping search engine.

The figure below demonstrates the activity flow for this shopping search engine. The flow begins when the user first runs the application home screen of shopping search engine that appears in the web browser. The user can browse through and can view all the available list of products and can choose either to select a product or to directly view the cart. In the thumbnail of the product, a user can click to view more information for details about a particular item before deciding to add it to the shopping cart by clicking on anywhere in the thumbnail. The user can then decide to either continue shopping by clicking the continue shopping button or can check out by clicking on the checkout option available on the brand’s official site. If there are no items in the cart, then the user does not have an option to click checkout. The user can check out after doing the user authentication by logging in with the username and password. Once the user successfully logins/registers, the order form, where the user can put the correct information to place the order appears. If the user includes incorrect or incomplete

information, then placing the order is not allowed. After the user successfully inputs the correct information, placing order is successful, and the user can see the success message. The additional flow step for the administrators is that they can view the user’s information, the user’s checkout, and the product details by using the database after the user successfully places an order. And note that, all this order placing procedure will happen on the official site of brand.

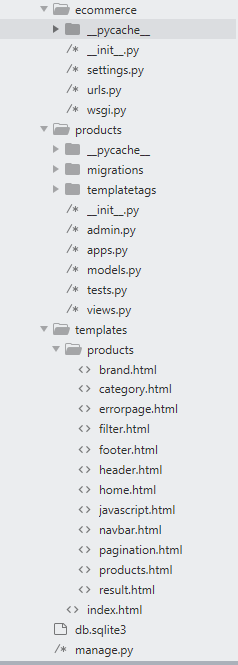


**Figure 2.** Activity Diagram for Shopping search engine.

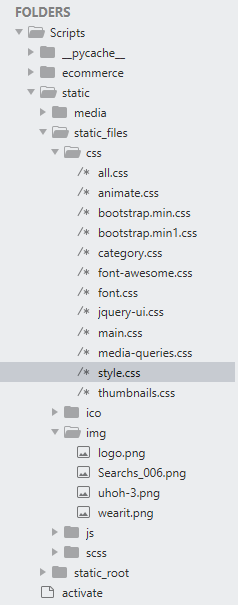
# The Ladies shopping search engine’s Implementation

This section contains the implementation details for different packages and classes of this application. It also contains the coding snippets that help computer science students understand what a particular section of code represents. The following steps are required to run the application successfully:

* 1. Install the Sublime Editor and Python 3.6
  2. Add Python to environmental path, and run “pip install django==2.2.6” and then run with pymongo and virtualenv
  3. Create virtual environment by running ‘python –m venv folder\_name’ in cmd
  4. Activate the virtual environment by running ‘ cd folder\_name/Scripts/activate
  5. Now run ‘django-admin startproject ecommerce in cmd
  6. In the package explorer of sublime, you’ll see ecommerce folder within virtual environment containing some files like in Fig.3
  7. In ecommerce folder, some files added by default and some files are created from the scratch
  8. Main files, in which most of the coding done, are setting.py, urls.py,views.py, and templates folder contains all the html files of this application
  9. One more folder, which is static folder, contains all the css,media, and JavaScript files shown in Fig. 4



**Figure 3.** The Database Package.



**Figure 4.** The Database Package.

Index.html is a client-side implementation and is responsible for displaying the application’s default main page. The code is implemented in HTML and Django which contains the template for the user-interface display. The coding snippet in Figure 5 shows the setup of the main default page that contains the application title and calls the display of other html page.

<!doctype html>

<html lang="en">

<head>

{%include 'products/header.html'%}

</head>

<body>

{% block content%}

{% endblock %}

</body>

</html>

**Figure 5.** Implementation of the Default Page for the Wearit.

The shopping-session represents server-side and client-side implementation of this shopping search engine. All the code is divided into some functions which perform some specific processing. The business interface for this shopping search engine is a fairly plain Django interface that defines all the business logic implemented in the view.py file. The snippet in Figure 6,contains some important functions libraries, and database connection.

**Functions explanation:**

1. **Get\_random():**

Used to sort all the products in the database in ascending order corresponding to their prices and return list of dictionaries.

1. **Index():**

The most basic function in django, used to display the home page of the the app.

**Figure 6.** View.py Code Snippet

**from django.shortcuts import render**

**from django.core.paginator import Paginator**

**from django.conf import settings**

**import pymongo**

**import re**

**client = pymongo.MongoClient("localhost",27017)**

**db=client.scraped\_data.all\_brands**

**# Create your views here.**

**def get\_random():**

**li=list(db.find().sort([("Price", pymongo.ASCENDING)]))**

**return li**

**random\_products=get\_random()**

**def index(request):**

**template='products/home.html'**

**documents=pagination(request,random\_products)**

**return render(request,template)**

1. **Pagination():**

This method divides all the products in the ratio of 15 per page to display.

1. **Brands():**

This method return a list of the brands whom products are present the database.

1. **Count\_brand():**

It takes a list of dictionaries, calculate and return total count of product’s brand name by removing duplicates.

**def pagination(request,doc):**

**Figure 7**. View.py Code Snippet(cont.)

**paginator=Paginator(doc,15)**

**page=request.GET.get('page')**

**doc=paginator.get\_page(page)**

**return doc**

**def brands():**

**raw\_data=random\_products**

**all\_brands=[]**

**for doc in raw\_data:**

**if doc.get('Brand') not in all\_brands:**

**all\_brands.append(doc.get('Brand'))**

**return all\_brands**

**brands=brands()**

**def count\_brands(li):**

**raw\_data=li**

**all\_brands=[]**

**for doc in raw\_data:**

**if doc.get('Brand') not in all\_brands:**

**all\_brands.append(doc.get('Brand'))**

**return len(all\_brands)**

1. **filter\_form(request):**

This method calls upon whenever a Get request is sent after applying filters on the price range. It either returns a template of products within specified price range or error page showing some message regarding to that search.

**def filter\_form(request):**

**Figure 8.** View.py Code Snippet(cont.)

**if request.method == 'GET':**

**query\_dict=request.GET**

**select\_brands= query\_dict.\_\_getitem\_\_('searched')**

**price\_range=query\_dict.\_\_getitem\_\_('range').split('-')**

**min\_price=int(re.search(r'([\D]+)([\d,]+)',price\_range[0]).group(2).replace(',',''))**

**max\_price=int(re.search(r'([\D]+)([\d,]+)',price\_range[-1]).group(2).replace(',',''))**

**listOfDocumets=[]**

**listOfDocumets=list(db.find({'$or' :[{'title': {'$regex': "(?i)"+select\_brands} ,'Price': { '$gte': min\_price,**

**'$lte': max\_price }} , {'Brand': {'$regex': "(?i)"+select\_brands} ,'Price': { '$gte': min\_price,**

**'$lte': max\_price }} , {'cat\_name': {'$regex': "(?i)"+select\_brands} ,'Price': { '$gte': min\_price,**

**'$lte': max\_price }}]}).sort([("Price", pymongo.ASCENDING)]))**

**if len(listOfDocumets)>0:**

**paginated\_data=pagination(request,listOfDocumets)**

**return render(request, 'products/products.html',{"documents": paginated\_data, 'searched': select\_brands,'Total\_products':len(listOfDocumets), "count\_brands":count\_brands(listOfDocumets), 'min\_price':listOfDocumets[0]['Price'], 'max\_price':listOfDocumets[-1]['Price']})**

**else:**

**return render(request,'products/errorpage.html')**

1. **search\_query(request):**

This method call upon whenever a user enter something into the search field. It either returns a template of products or error page showing some message regarding to that search.

**def search\_query(request):**

**if request.method == 'GET':**

**query\_dict=request.GET**

**select\_brands= query\_dict.\_\_getitem\_\_('search')**

**if select\_brands!='':**

**listOfDocumets=list(db.find({'$or' :[{'title': {'$regex': "(?i)"+select\_brands}} , {'Brand': {'$regex': "(?i)"+select\_brands}} , {'cat\_name': {'$regex': "(?i)"+select\_brands}}]}).sort([("Price", pymongo.ASCENDING)]))**

**if len(listOfDocumets)>0:**

**documents=pagination(request,listOfDocumets)**

**return render(request, 'products/products.html',{"documents": documents, 'searched': select\_brands,'Total\_products':len(listOfDocumets), "count\_brands":count\_brands(listOfDocumets), 'min\_price':listOfDocumets[0]['Price'], 'max\_price':listOfDocumets[-1]['Price']})**

**Figure 9.** View.py Code Snippet(cont.)

**else:**

**return render(request,'products/errorpage.html')**

**else:**

**return(index(request))**

1. **error(request,exception=None):**

As it show, this method takes two parameters(one is none), and it returns a proper message whenever a user makes some mistakes.

**Figure 10.** View.py Code Snippet(cont.)

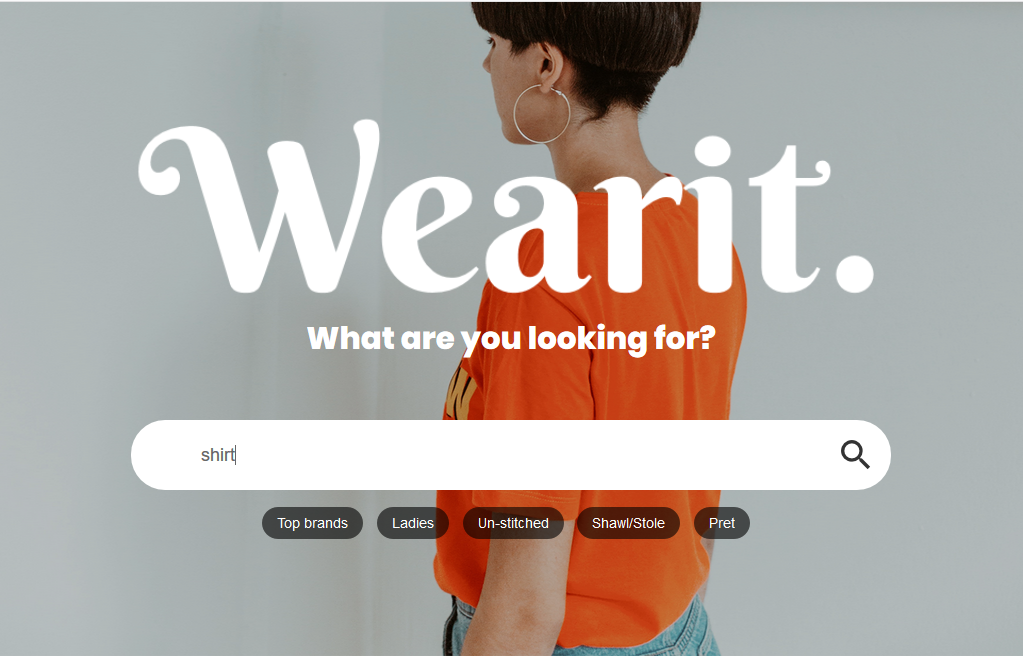
**def error(request,exception=None):**

**return render(request,'products/errorpage.html')**

# The Ladies shopping search engine’s Interface

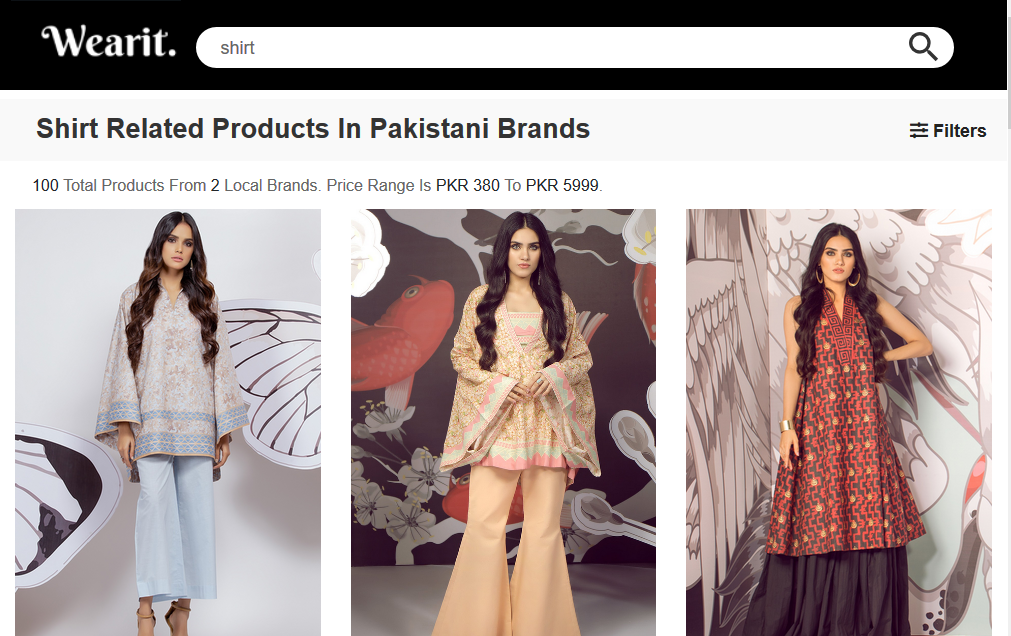
This section describes the different interfaces for the shopping search engine. It contains a detailed description about each interface along with a screen shot of the interface.

1. **Home Page:** The home page of the application (Figure 11) is common to all the system users. This interface is available through the web application. Here the user will interact with the logo of the search engine along with a search field where, he/she will be able to find the products of different brands. Like the user may enter shirts,kurti,shalwar,shawl,stole etc. to find the related products from all the brands.



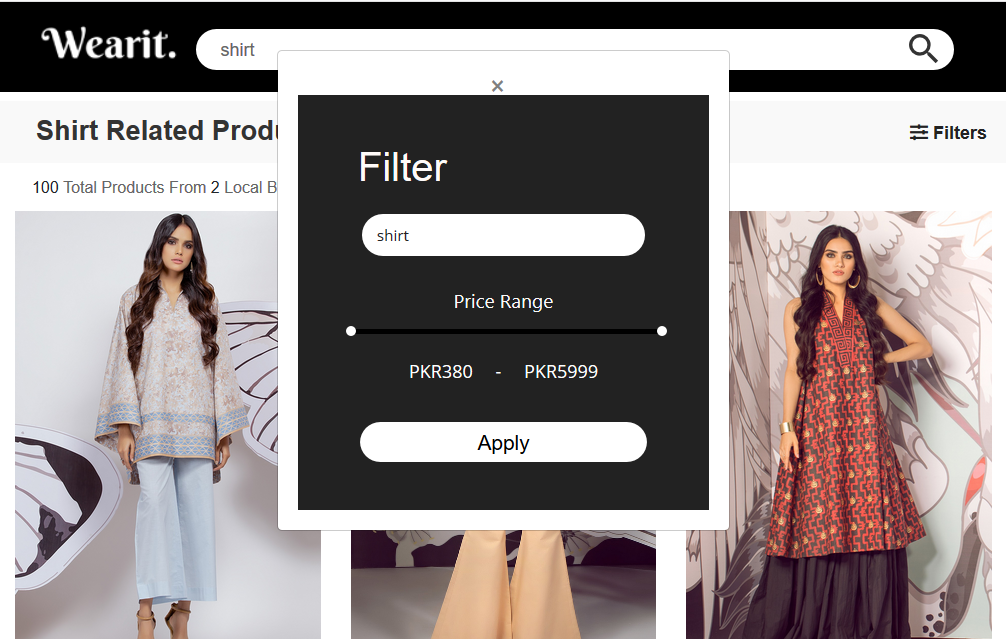
**Figure 11.** Screenshot of the Home Page.

1. **Product Interface:** This interface shows all the related products in the form of grids along with all the details about the brands, price, image etc. as shown below



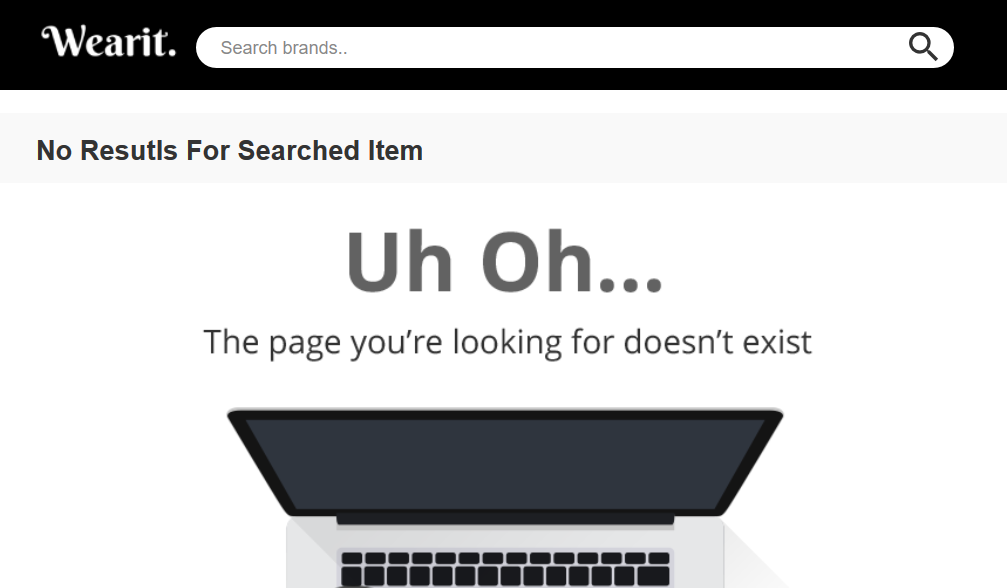
**Figure 12.** Screenshot of the Electronics Category

1. **Apply Filter:** This feature allow user to filter the products within specified price range by moving sliders like



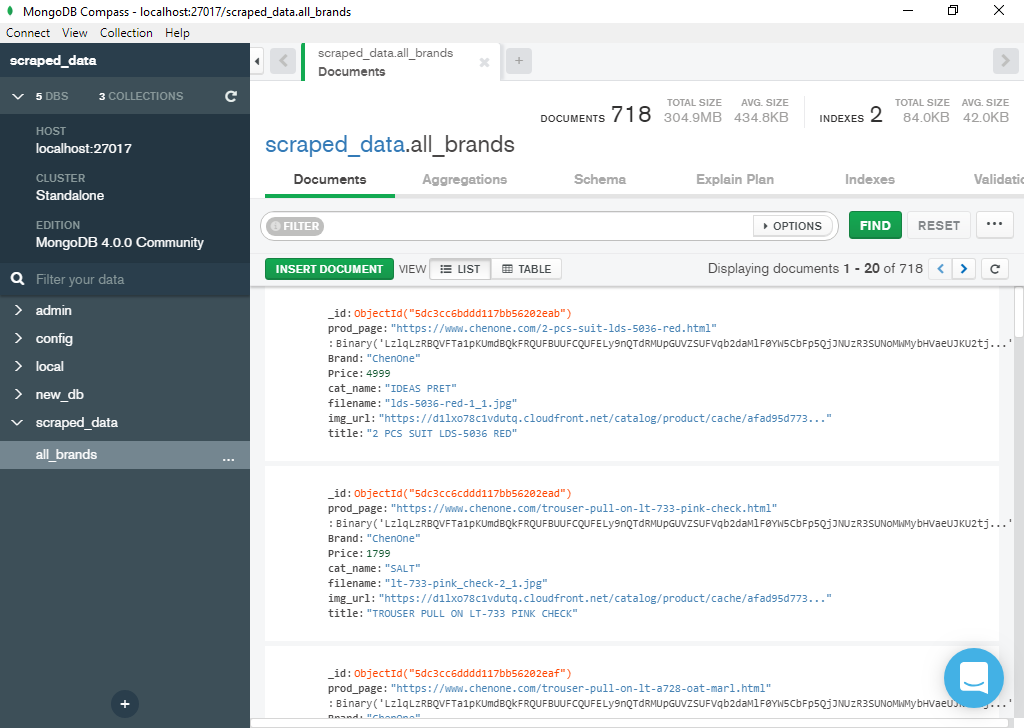
**Figure 13.** Screenshot of the Electronics Category.

1. **Error Page:** This page display a message when user enters something which is not available in the database.



**Figure 14.** Screenshot of the View Cart.

1. **Database Interface:** In this portion, MongoDB Compass is used to display the information stored into the database. Here it is



**Figure 15.** Screenshot of the Database

# CHAPTER 4. TESTING

This chapter includes the methods that were used for testing, validating, and evaluating the system. The Conclusion and the Future Work for the software are also given.

# Methodology

With this testing approach, all the specs were ready for a prototype, and a plan was already built to be shown; the tester started writing his or her code and saw if he or she could obtain the same results that the specs mentioned. This way, the specs were tested on each prototype, and continuous testing was applied. This also helped to minimize the testing that would have to be implemented at the end of the software lifecycle. In the process, all aspects of the software were tested. Steps to follow while implementing the methodology are as follows:

1. Start with a base function that you want to implement.
2. Create a document with the detailed requirement definition, an activity diagram with a description of the flow, database tables to be used, a component diagram, and a description of each component with the precondition and tables that would be affected by the component.
3. Give the document to the tester, and work with the tester while he or she writes the code to check if the steps in the document can be implemented and if the result of each use case can be achieved.
4. If the tester finds a step difficult to implement or thinks he or she is missing additional information to implement the functionality, then go to step 2; otherwise, go to step 3.
5. Ask the tester to log on all the errors and difficulties he or she encountered while working on the prototype implementation.
6. Once the prototype is done and the results between the developer’s prototype and tester’s

prototype match, work on the other requirement, and expand the prototype to final software.

1. When the testing approach was implemented, the following pros and cons regarding the testing approach were realized.

Pros of using the methodology

* + Helps give a better understanding about the requirements.
  + Better design at the end of the cycle.
  + Reduced testing to be performed at the end of the cycle
  + Documents produced would be of higher quality.

Cons of using the methodology

* + The person working on the document should be experienced.
  + There are increased time and money involved with testing.
  + Different viewpoints for the same problem can lead to varying results.

# Interface Testing

This section lists the functional requirements used for creating the test-case table, the test cases that were used to verify the interface table, and the results for the test-cases table.

Table 1 lists the functional requirements for the interface built for this application, along with a short description of each requirement

**Table 1.** List of Functional Requirements

|  |  |
| --- | --- |
| Functional Requirement Number | Functional Requirement Short Description |
| FR01 | The shopping search engine shall have a search bar which allows user to enter a keyword which will then upon pressing enter produces some results |
| FR02 | The shopping search engine shall be accessible  to all the users to browse all the products and the brands. |
| FR03 | The data will be populated into thumbnail forms |
| FR04 | Data must gathered from top brand’s websites of ladies dresses of Pakistan |
| FR05 | Upon clicking of each thumbnail, the user must redirected to the webpage of brand’s official site where he/she can do purchasing in just one click |
| FR06 | If user entered a keyword which is not present in any of the website, then a proper message should be displayed. |
| FR07 | Search and Filter criteria provide the user all the products within specified price range |

# Test Cases

Table 2 shows the functional requirements used to write the test cases along with the test- case numbers for each test case and a short description of the test cases.

**Table 2.** List of Test Cases.

|  |  |  |
| --- | --- | --- |
| Functional Requirement No. | Test Case  No. | Test-Case Short Description |
| FR01 | TC01 | To test the search bar functionality by writing different queries and pressing enter button |
| FR02 | TC02 | To test if all the products are accessible or not |
| FR03 | TC03 | To test, if the products are well organized in the form of thumbnails |
| FR04 | TC04 | To test, whether all the top brands’ products showing or not |
| FR05 | TC05 | To test, if the user will redirected to the brands’ official site to purchase in one click |
| FR06 | TC01 | To test, if the user get proper message in case of un-recognized query. |
| FR07 | TC06 | To test, Search and Filter criteria provide the user all the products within specified price range |

The following list includes the steps that should be taken by the user, the conditions that should be met for the successful execution of the test case, and the end result that should be met for the test cases to pass.

1. TC01: To test the search bar functionality
   * Input: Become an online shopping person and enter a key word
   * Processing: Hit enter and see if you get any result
   * Valid Range: Keyword Alphanumeric
   * End Messages/Result
2. If (keyword == Matches), a response page will be received
3. If (Keyword != Matches), an error message is displayed on the interface.
4. TC02: To test, if all the products are accessible or not
   * Description of the search: The system shows all the saved items in the interface for a particular user along with a detailed description about how many products and brands are the result of entered query with the minimum and maximum price range.
   * Input: The user entered a keyword in search field.
   * Output: The system’s page pops up, showing the products along with a description of search.
   * End messages/Result
5. If (keyword == Matches), the user is able to view all the products related to that search.
6. If (keyword != Matches) an error message is displayed on the interface.
7. TC03: To test, if the products are well organized.
   * Description: If the user enters a keyword, the products should be shown in well-organized way so he/she can make better decision.
   * Input
8. Keyword
   * Output: In the form of well-organized thumbnails.
9. TC04: To test whether all the top brands’ products showing or not
   * Description: This system lists about more than top 10 brand’s products. And every search should display products from almost all the brands if that searched keyword exist in any of them.
   * Input
10. Keywords
    * Output: List of all the products exists in any brand relevant to that keyword
11. TC05: To test, if the user will redirected to the brands’ official site to purchase in one click
    * Description: For all the users, there is button ‘Buy Now, which upon clicking will redirect users to the brand’s official site where he/she will immediately continue filling shipping information to get home delivery.
12. TC06: To test, Search and Filter criteria provide the user all the products within specified price range
    * Description: There is a button named filters, which upon clicking will pop up a window, through that the user can filter products according to their wish list of price range.
    * Input
13. Click Filters option
14. Move Price Sliders to their budget price
15. Hit Apply Button
    * Output: Display of all the products in specified range.
    * End messages/Result
    1. If (keyword == Matched & Price-Range == exists), then display of the all products
    2. If (keyword == Matched & Price-Range != exists), then display of the error message.

# Results

This section lists the results that were produced by running the test cases. Table 3 lists the test cases that were used while testing the interface along with the expected result and the actual results for each test case.

**Table 3.** List of Test-Case Results.

|  |  |  |
| --- | --- | --- |
| Test Case Number | Expected Result | Actual Result |
| TC01 | Pass | Pass |
| TC02 | Pass | Pass |
| TC03 | Pass | Pass |
| TC04 | Pass | Pass |
| TC05 | Pass | Pass |
| TC06 | Pass | Pass |

# CHAPTER 5. CONCLUSION/FUTURE WORK

This chapter includes the Conclusion reached after creating the current version of the software to meet the system objectives. The comparison is done between the system that was built and original requirements that were designed at the beginning of the project. It also describes the Future Work that is intended to be accomplished with later versions of the software.

# Conclusion

The main objective of the application is to bring all products of the top brands of Pakistan in one platform along with their price comparisons from where user can make better decisions in purchasing regardless of different alternatives.

# Future Work

The following section discusses the work that will be implemented with future releases of the software.

1. **Detailed products:** Future work could involve adding more products which are more detailed and have additional HQ images.
2. **Watch/Wish List:** Work can add a watch list or wish list so that users can add an item to a list to watch for item prices to go down or to see when there is a sale on any of those items.
3. **Enhanced User Interface:** Work on enhancing the user interface by adding more user- interactive features.
4. **Recommended Items:** Add a bar that would display the most-recommended items which would depend on the number of times a product has been purchased by any users.
5. **Recent History:** Display the user’s recently browsed items in the recent-history tab.

# REFERENCES

1. [Treutner, M. F., and Ostermann](http://benthamscience.com/open/toisj/articles/V005/8TOISJ.pdf), H. Evolution of Standard Web Shop Software Systems: A Review and Analysis of Literature and Market Surveys*.*
2. Jarvenpaa, S. L., and Todd, P. A. (1997). Consumer reactions to electronic shopping on the World Wide Web. International Journal of Electronic Commerce, 1:59–88.
3. ["Comparison shopping website [STUDY]."](https://en.wikipedia.org/wiki/Comparison_shopping_website)
4. King, Stephen F. en Juhn-ShiuanLiou (2004), [A framework for internet channel evaluation](http://www.slideshare.net/m9821735856/internet-channel-evalution). International Journal of Information & Management 24:473–488.
5. Falk, Louis K., Sockel, Hy, and Chen, Kuanchin. (2005). E-Commerce and consumer's expectations: what makes a website work. Journal of Website Promotion, 1:65–75.
6. Elliot, Steve, and Fowell, Sue. (2000). Expectations versus reality: a snapshot of consumer experiences with Internet retailing", International Journal of Information Management 20: 323–336.
7. Commerce4j, [Java Based e-Commerce Application and Online Catalog Management System](https://code.google.com/p/commerce4j/). , Retrieved on May 18, 2010. "https://code.google.com/p/commerce4j/"
8. CS-Cart.com (Simbirsk Technologies Ltd), © 2004-[2013.htt](http://www.cs-cart.com/)p:/[/www](http://www.cs-cart.com/).c[s-cart.com/](http://www.cs-cart.com/)
9. Comparison of shopping cart software., Retrieved on June 28, 2013. "<http://en.wikipedia.org/wiki/Comparison_of_shopping_cart_software>"