CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.1 INTRODUCTION

The implementation of the web scraping tool for e-commerce websites is discussed extensively in this chapter. System implementation is the utilization of system requirements and design to build a functional new system.

4.2 SYSTEM REQUIREMENTS

4.3 THE IMPLEMENTATION TOOLS USED

The web application was built by integrating different components together to form the functional web scraper application. The Frontend or User Interface was built using Javascript as the programming language and vanilla JS as the frontend framework. The backend handling the logic or model of the web application runs on Django, a high-level python web framework that follows the MVC Architecture.

The web scraper library employed is Beautifulsoup, a python package for effectively scraping data from web pages. Beautifulsoup fetches product data for the web scraper application to consume for service delivery.

SQLite is the backend database for holding all user and product data for the web scraper application. SQLite is a flexible relational database and the default database for the Django web framework. The SQLite database is fast, lightweight and highly compatible with Django.

4.4 SOFTWARE DEVELOPMENT METHODOLOGY

4.5 THE PROGRAM MODULES AND INTERFACES

This section outlines the various modules of the web scraper application

The application comprises of different modules which interact and share information. These different modules are created to solve a specific task and has a differen t interface for displaying information to the user on the frontend.

* The Homepage
* The Result page
* The Admin Panel
* The Cart Page
* Signup page
* Authentication (Login Page)
* Hot deals
* API Implementation

4.5.1 The Homepage

The image in the figure above shows the homepage of the web scraper application. This is the landing page of the application where users will be directed to when they visit the application for the first time. The homepage comprises of different sections or containers such as the landing section, the search bar, the product categories, and the menu of the web scraper application.

The search bar is a mini and concise form with a single field where users can enter a product they want to search and be redirected to another page for product result viewing. The product categories---------

4.5.2 The Result Page

The Result page of the web scraper application is where the search result comprising of scraped products and data are displayed on the frontend for users to view. The search results are presented in nice and easy to understand user interface for customer retention and easy comprehension.

Each product is displayed with its own unique image, purchase url, name and product price. A number of tasks can be performed on the result page and they include:

* Buy product: Each product on the result page has its own unique product url. This is product the url on the merchant website for the purchasing the specified product. Clicking on the “Buy Now” Button will redirect the user to a new tab on the merchant website where they can complete purchase of the desired product.
* Save product: Users can also save a product to be purchased or reviewed later using the cart or save feature of the web scraper application. The saved product is stored in the SQLite database and can be deleted from the cart on the cart page.

4.5.3 The Admin Panel/Dashboard

The Admin panel is a page designed for the administrators of the web application. The admin dashboard is a dashboard where admins can interact, create, read, update and delete (CRUD) model instances of the web scraper tool. Some of the model instances for the application includes the User Model, the SavedProduct Model. Django Models are simply class based representation of the tables in the SQLite Database.

The User model is the model for all the users that have registered on the application. Some of the tasks admins can perform on this model from the admin panel include:

* Create a new user
* Give users basic or admin privileges
* Delete user
* Read user information
* Update user information
* Delete user
* Suspend user account on the web app. Suspended users cannot login or use their account

The SavedProduct Model is the model that represents the products saved to cart for later review and purchasing. From the admin panel, some of the tasks that can be performed on the SavedProduct Model include:

* Create a new saved product
* Alter the details of the saved product
* Read details of saved products
* Delete a saved product

4.5.4 The Cart/Saved Product Page

Figures above shows the user interface of the cart page with products saved on the web app. This is where all saved products will be displayed. The tasks that can be performed on the cart page include:

* Buying Product
* Deleting saved product from the cart page or SQLite database