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 system (Feri & Sidiq, 2020).

The web scraper tool has been developed following core design and software development principles. The web application has been built with security, speed, user retention, and ease of use in mind.

4.2 SYSTEM REQUIREMENTS

The system requirements are the software and hardware requirements necessary to develop and run the web application effectively. System requirements are been gathered and discovered as the development of the application progresses and can change before project completion (Faizul , Hamzah, & Chen, 2012).

Table 4.1 Software requirements

|  |  |
| --- | --- |
| Requirements | Software |
| Operating system | Windows 10 |
| Database | SQLite |
| IDE | VScode and Pycharm |
| Programming languages | Python and Javascript |
| Python version | 3.9 |
| Django version | 2.2.2 |

Table 4.2 Hardware requirements

|  |  |
| --- | --- |
| RAM | 4GB |
| Processor speed | 2.1GHz |
| Processor type | Intel Core I5 |
| Hard disk drive | 500GB |

4.3 THE IMPLEMENTATION TOOLS USED

The web application was built by integrating different components to form the functional web scraper application. Visual Studio Code was used as the code editor due to its support for python and JavaScript. The web app was tested using the chrome editor for responsive design. Chrome browser was also helpful in inspecting and debugging the frontend (HTML, CSS and JavaScript) of the web application.

The web scraper library used is Beautifulsoup, a python package for effectively scraping data from web pages. Beautifulsoup was engaged to fetch product data for the web scraper application to consume for effective service delivery.

SQLite database was used as 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 (Sebastian, Kuriakose, & Mariam , 2016).

The web scraper tool was deployed to pythonanywhere, an online editor and web hosting platform for python web applications.

4.4 THE PROGRAM MODULES AND INTERFACES

This section outlines the various modules and interfaces of the web scraper application for e-commerce. The functionalities and responsibility of each interface will also be discussed in details. Each page has a view and activity representing what is displayed and the action to be performed.

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

* The Homepage
* The Result page
* The Admin panel
* The Cart page
* Signup page
* Login/Authentication page

4.4.1 The Homepage

This is the landing page of the web application where users will be directed to when they visit the application for the first time. The homepage comprises of three major sections. The landing section, the search-categories section and the hot deals section.

* Landing Section

The landing section is the first section to be displayed on the homepage when a user visits the web application via the browser. The responsibility of this section is to capture the attention of the user and give a concise description of the service and functionalities of the web scraper tool.

Smart use of colors and image was engaged to build a proper landing section for the web scraper tool. Figure 4.1 below shows the landing section of the web scraper tool.

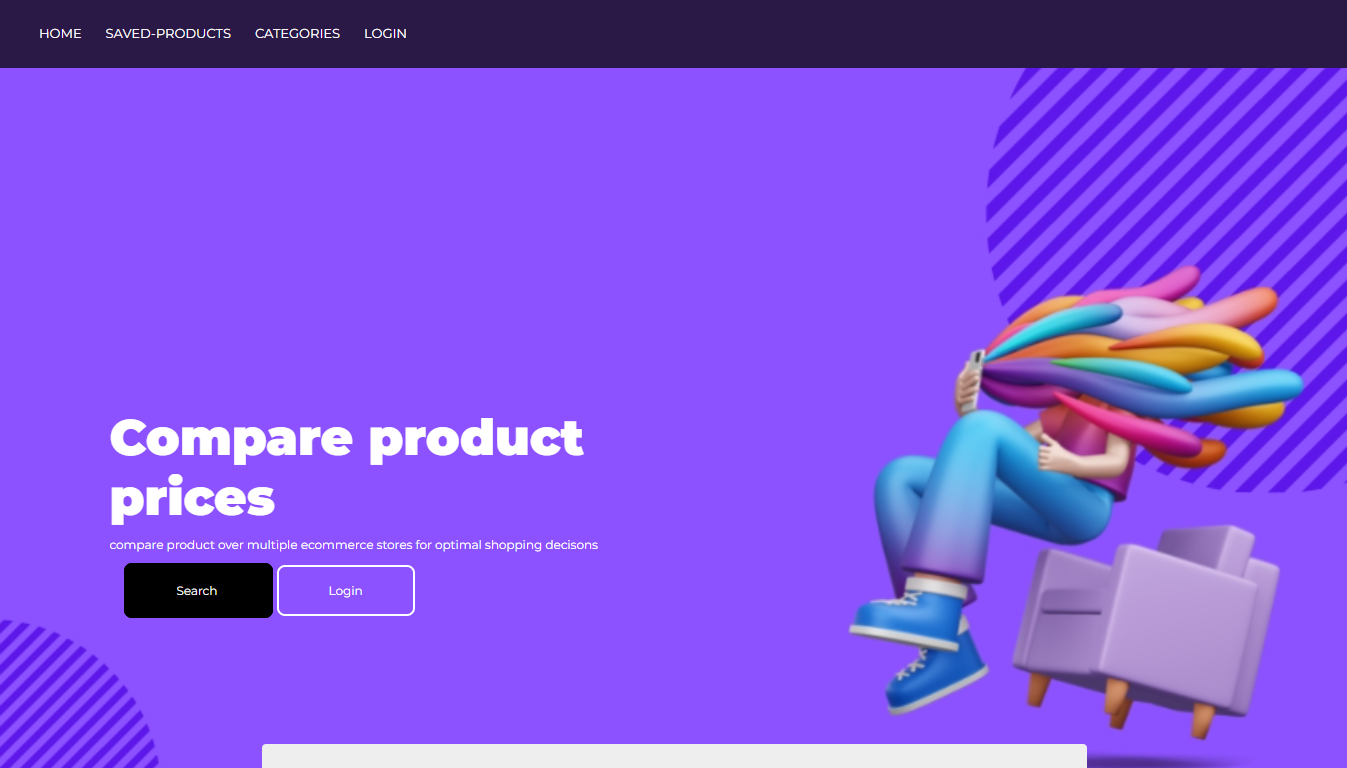


Figure 4.1 Landing section

* Search and Category Section

The search section is the next section that follows on the homepage. Users can search for a particular product using the search form on the search section by entering the product keyword into the form. Users will then be redirected to the results page where all scraped products related to the search keyword will be presented to the frontend interface.

The search keyword is sent to the backend where it is passed to beautifulsoup to scrape all related data from multiple ecommerce online stores. The result is then transferred from the backend to the frontend as context data in the form of a python dictionary.

The Category section shows all the top categories to be scraped in just a click. The categories to be covered ranges from electronic to fashion wears.

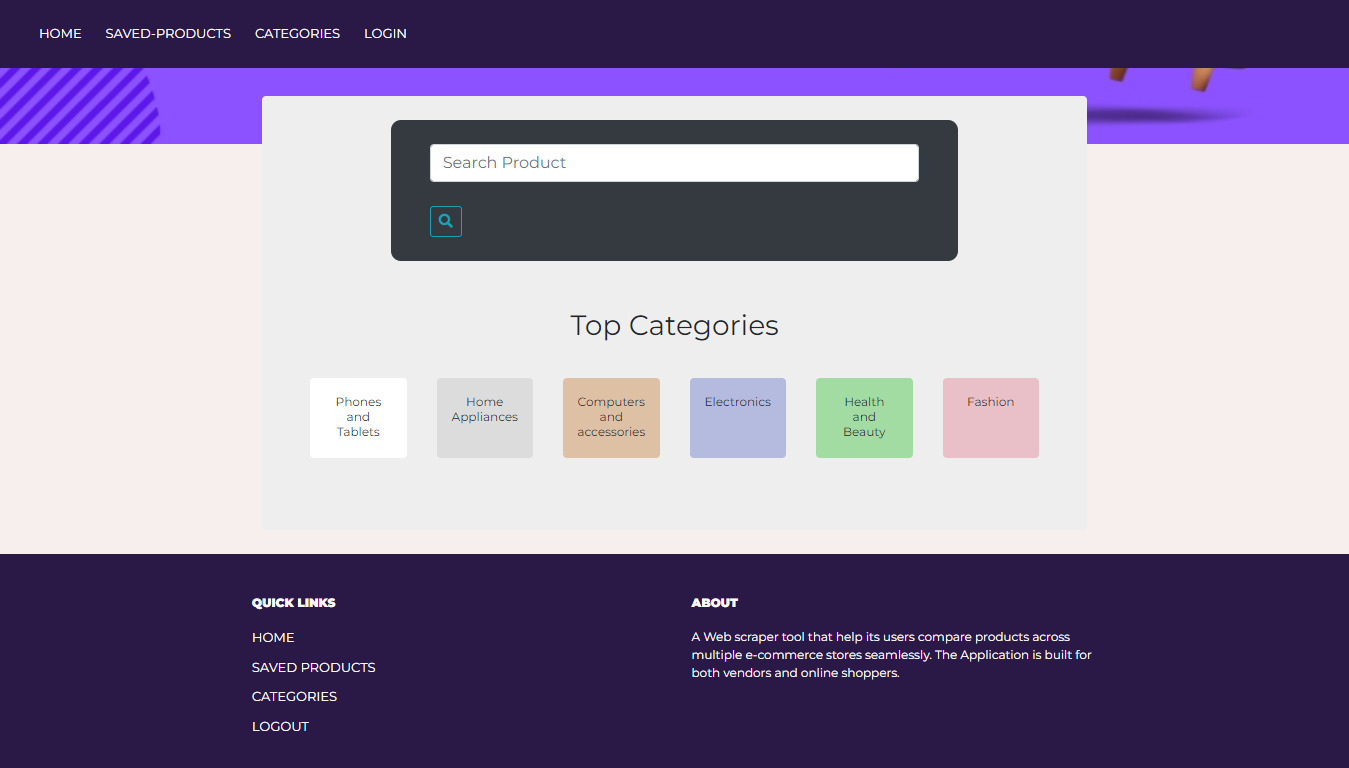


Figure 4.2 Search and category section

* Hot deals section

Ecommerce websites have best selling and hot deals displayed on a section of their website. The hot deals section of this web scraper tool represents the best selling deals scraped from multitple ecommerce stores presented in one section for easy comprehension.

This saves users the stress of visiting different ecommerce website to manually to view the hot deals available.

Hot deals from stores like Jumia, Konga and Kara are scraped and presented on the homepage of the web scraper tool in a single carousel/slider. This way users can scroll the slider and view hot deals from Jumia, Konga, Kara and others in one slide.

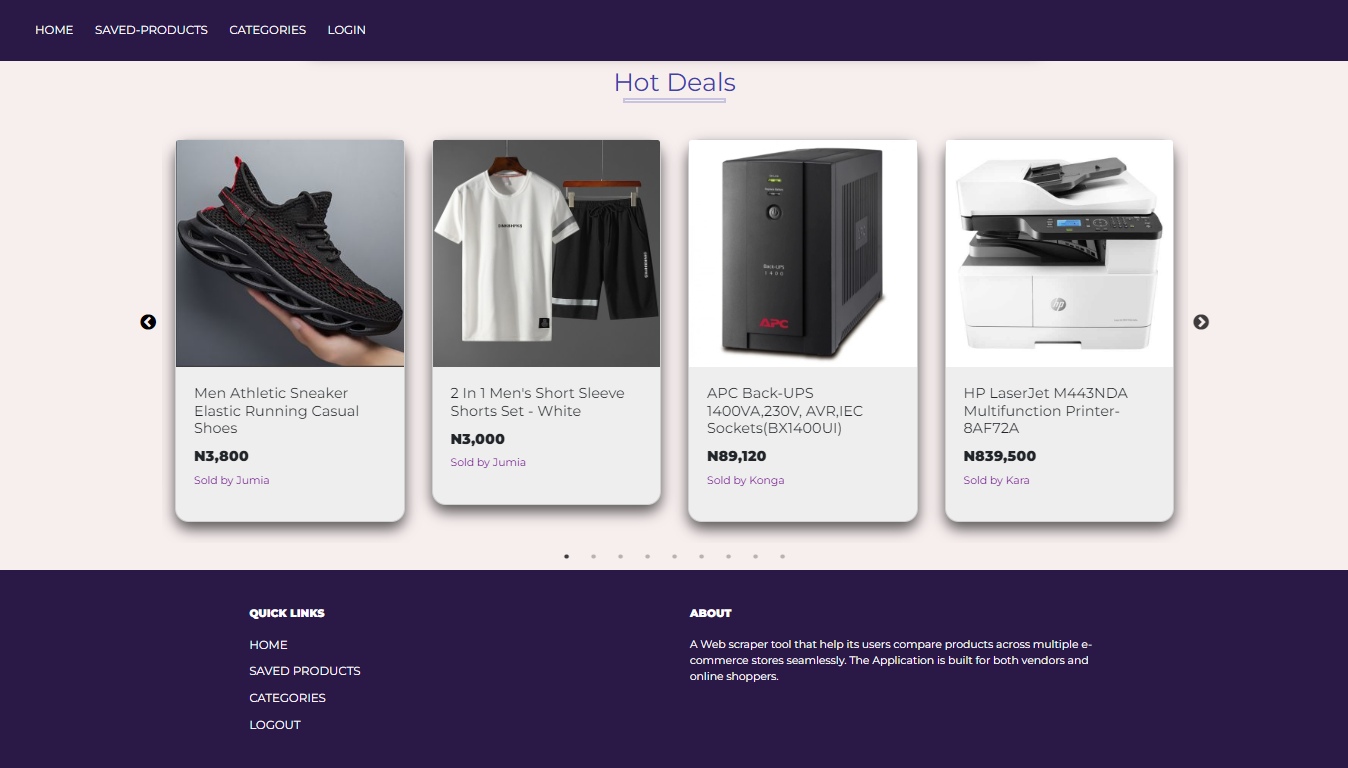


Figure 4.3 Hot deals section

4.4.2 Search Result Page

The Result page of the web scraper application is where the scraped results of the search-keyword are displayed on the frontend for users to view and complete purchase. The scraped products are presented in nice and easy to understand user interface for easy customer comprehension and interaction.

Each scraped product is displayed with fields like product name, product price, image url, vendor and product url which redirects users to the merchant store for purchase completion. Some activities can be performed for each unique product on the results page, and they include:

* Buy product: Each product on the result page has its own unique product url. This is the url of the product on the merchant website for completing purchase. 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. Only registered and logged in users can save products or use the cart feature. This is because each saved item has to be linked to a user as its author.

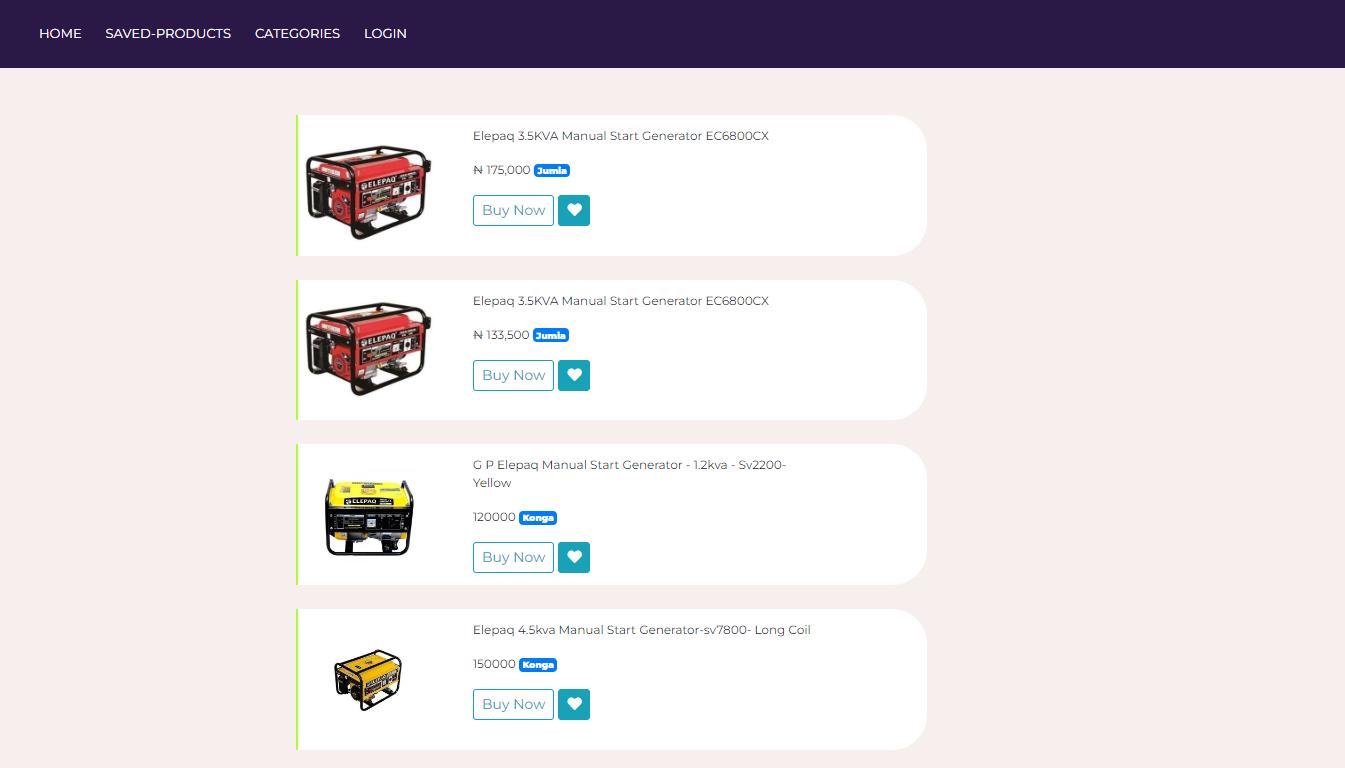


Figure 4.4 Search Result Page 1

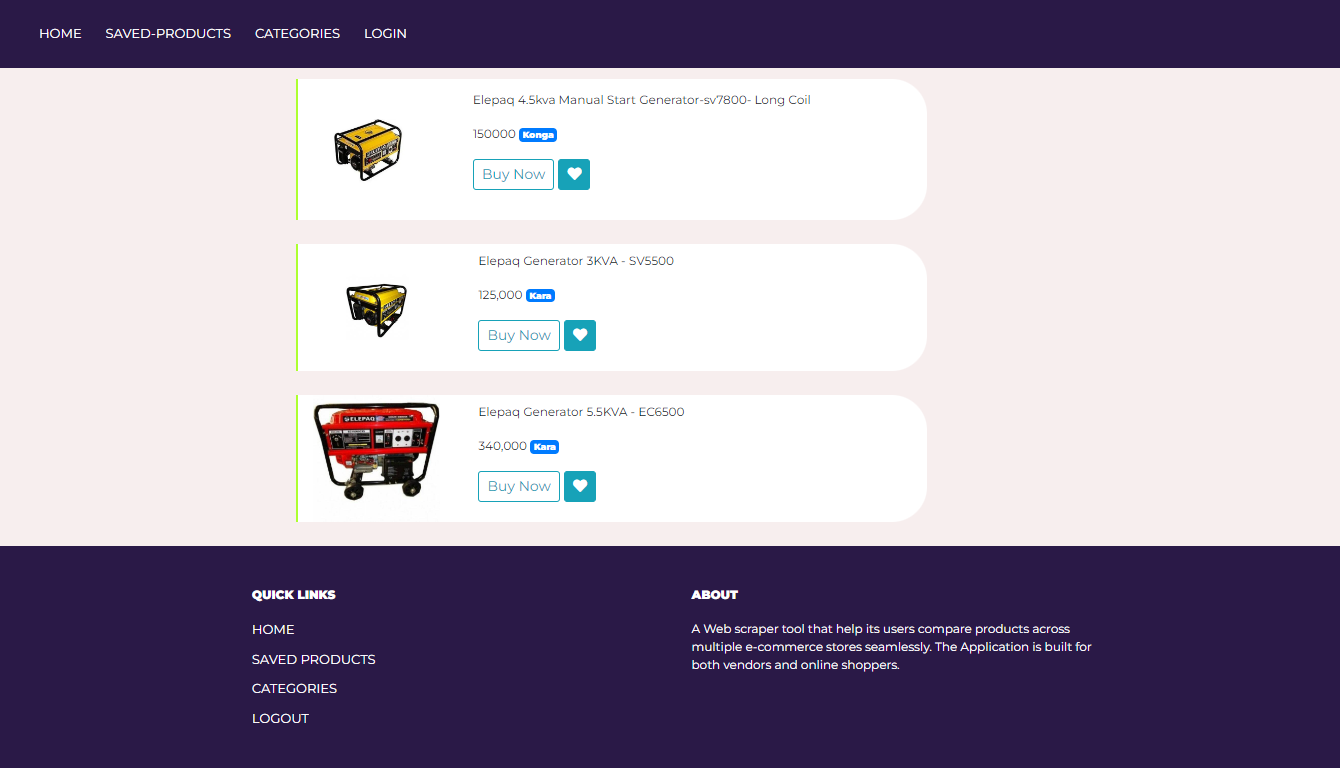


Figure 4.5 Search Result Page 2

4.4.3 The Admin Panel/Dashboard

The Admin panel is a page designed for the administrators of the web scraper 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 and the SavedProduct Model. Django Models are simply class based representation of the tables in the SQLite Database.

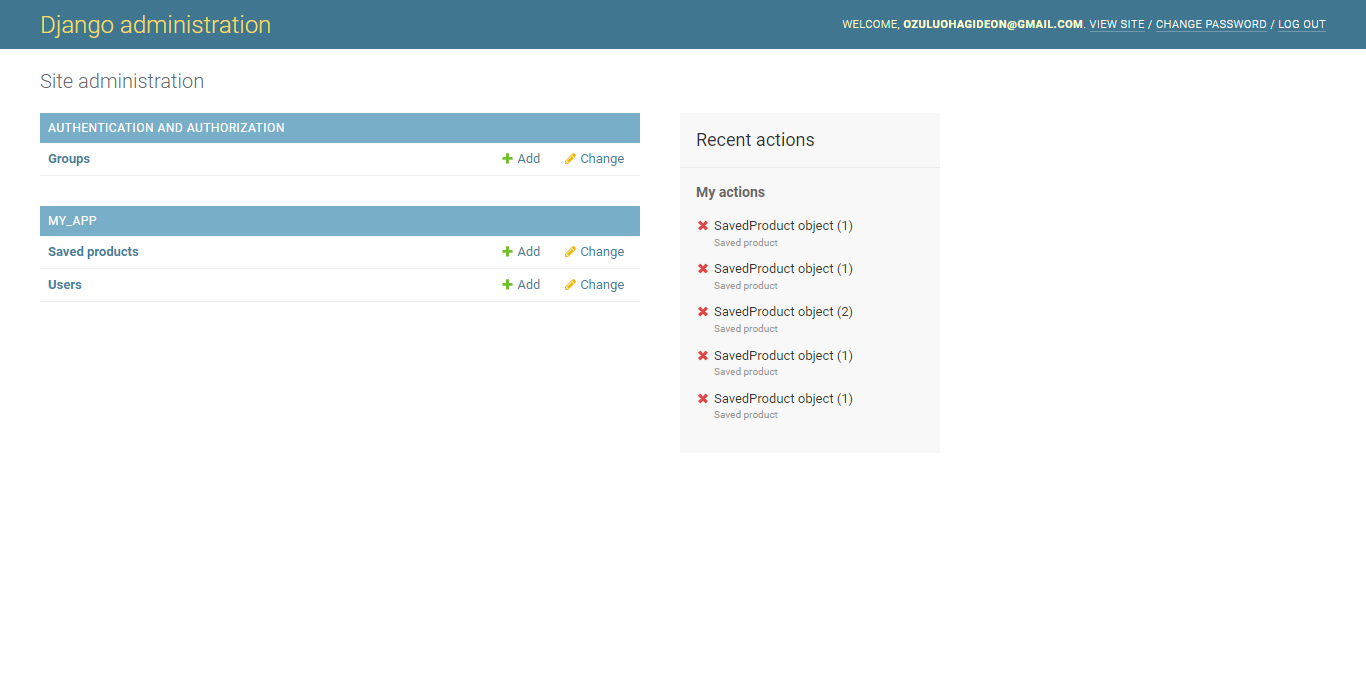


Figure 4.6 Admin Dashboard

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 or fields
* Suspend user account on the web app. Suspended users cannot login or use their account
* Grant or revoke admin privileges to user

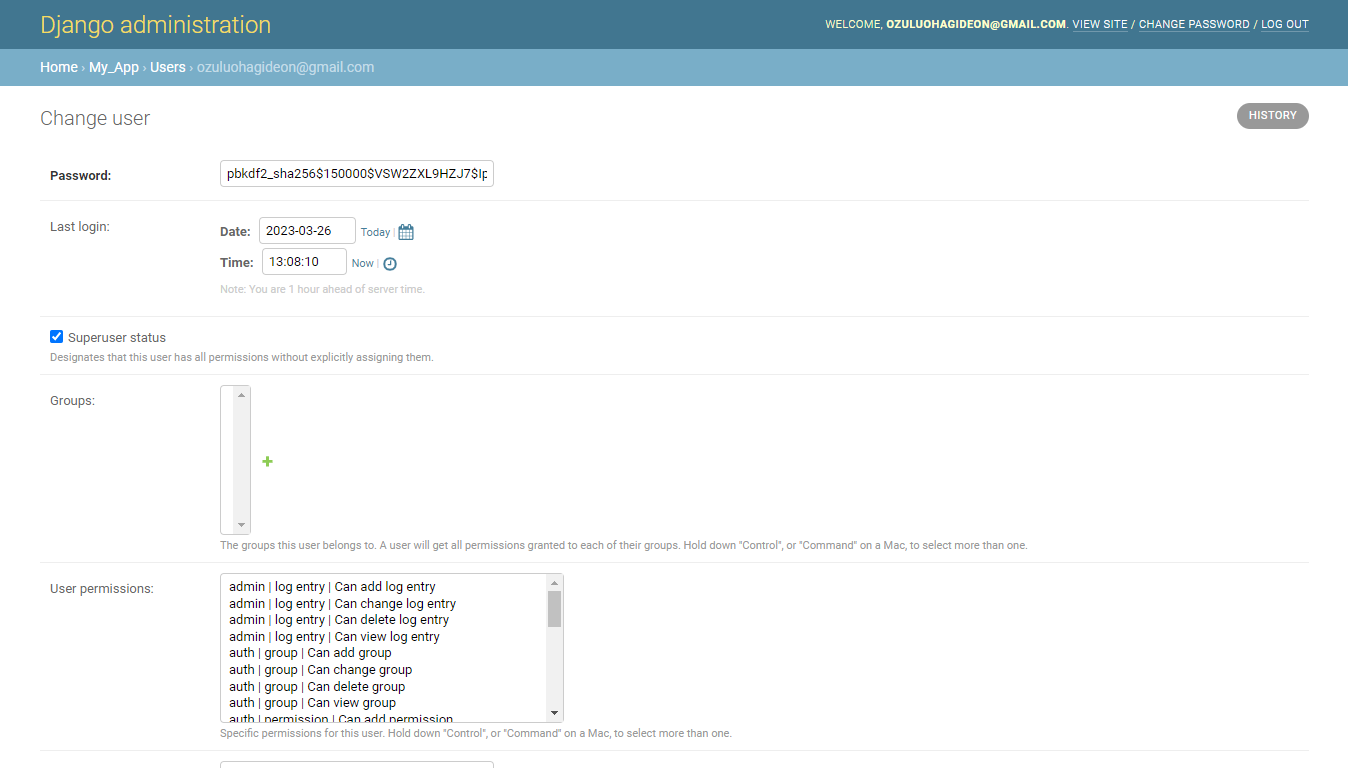


Figure 4.7 User permission and password update

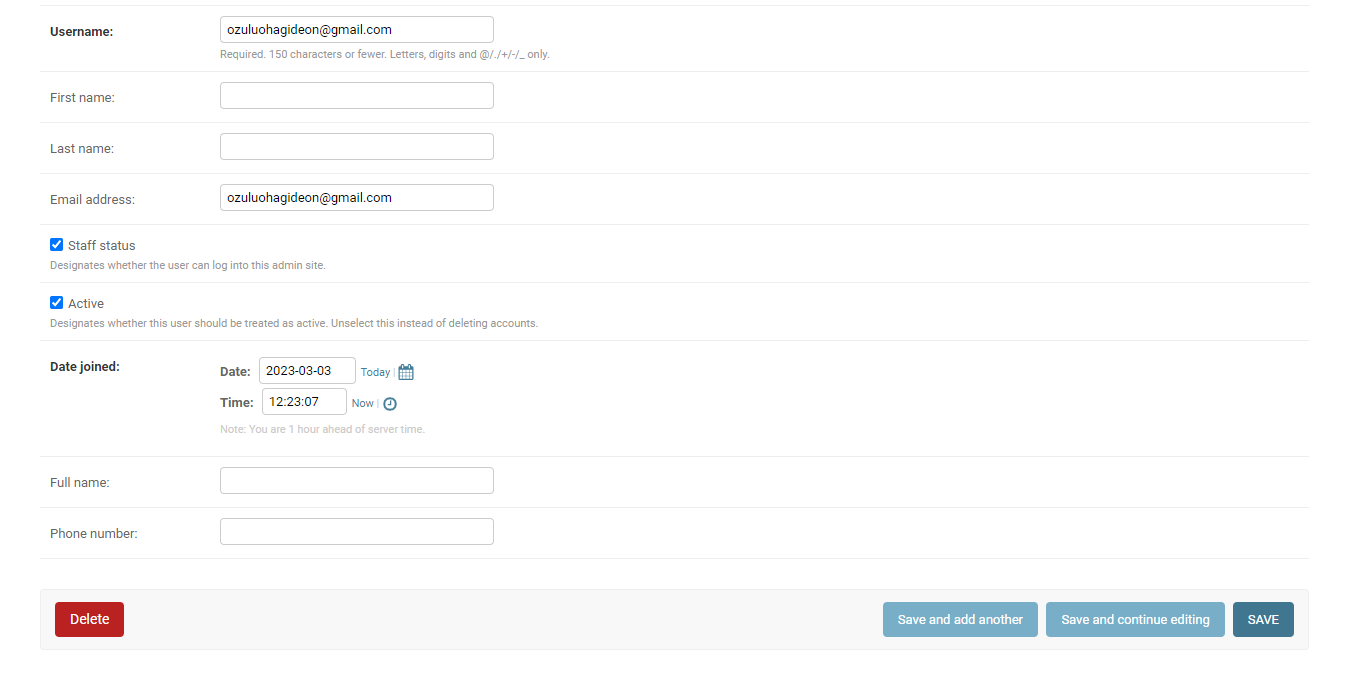


Figure 4.8 User fields update page

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

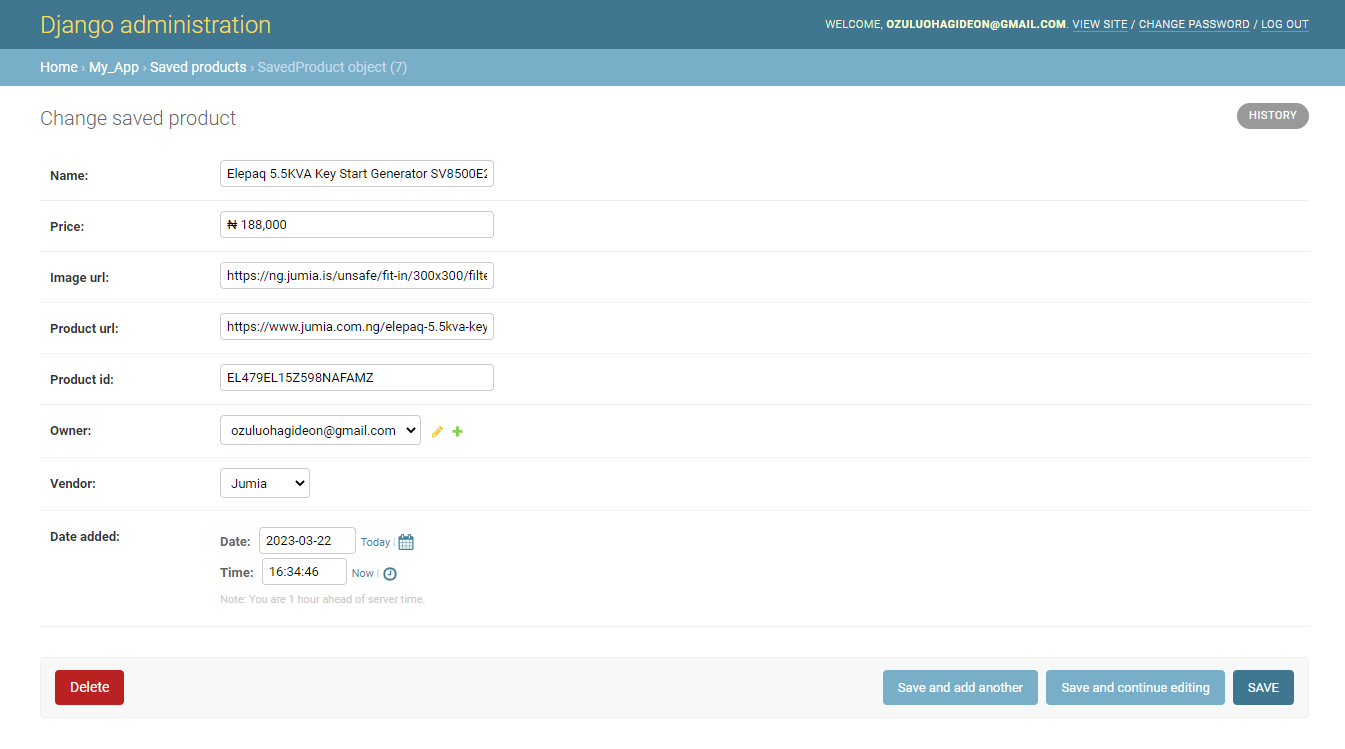


Figure 4.9 Saved products update page

4.4.4 The Cart/Saved Product Page

Users can decide to save a product they are interested for later purchase or processing. The cart page is designed to display all saved products for each user. Figure 4.9 shows the user interface of the cart page with products saved in the database. 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

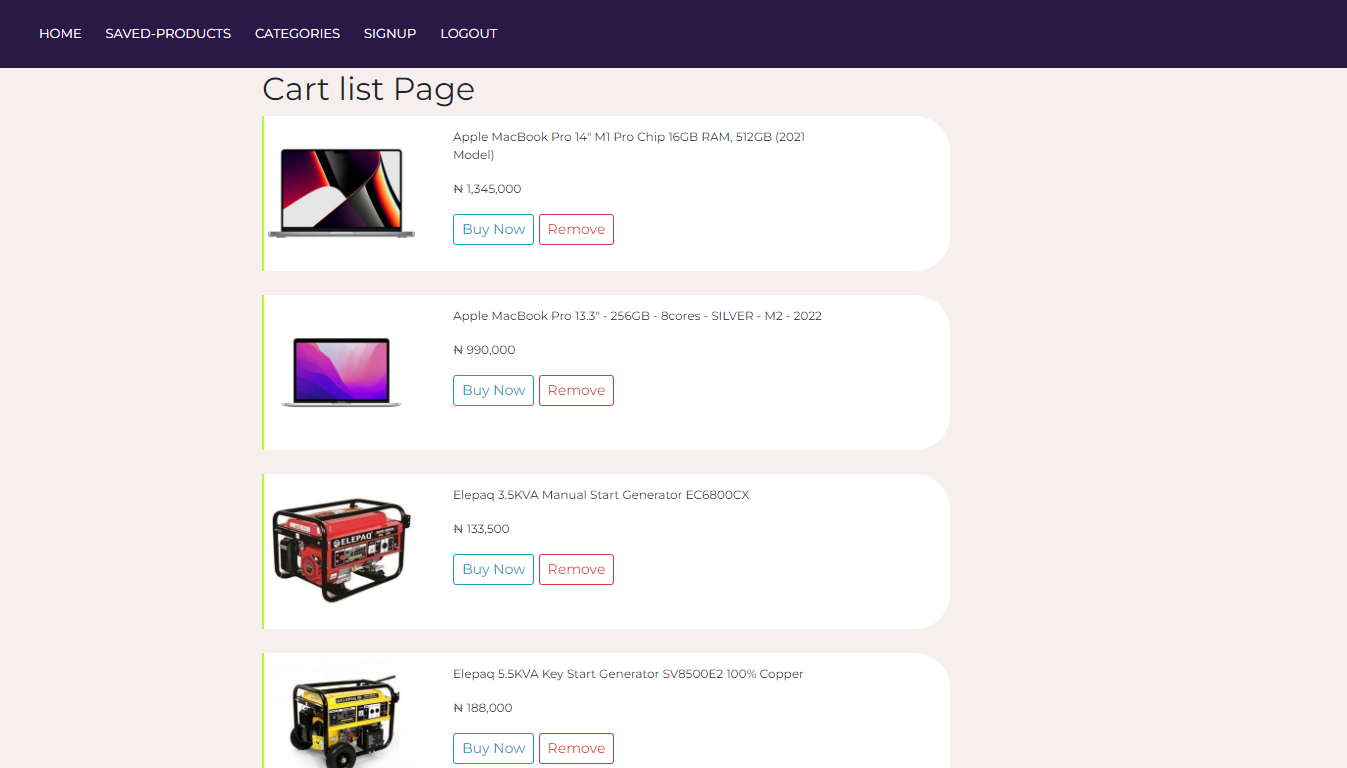


Figure 4.10 Cart/saved product page

4.4.5 Signup page

The signup page was designed with a form for registering new users to the app. The signup form was developed with five form fields for receiving information from users namely: full name, email address, phone number, password and password confirmation field. On form submission, A new user instance is created and saved into the database. The user’s password is encrypted before saving such that even administrators cannot identify the user password from the backend. The password encryption is implemented by a Django password hasher.

For the signup form to be successfully submitted to the backend for further processing, both the password and password confirmation field must perfectly match. To ensure proper validation of the user’s password.

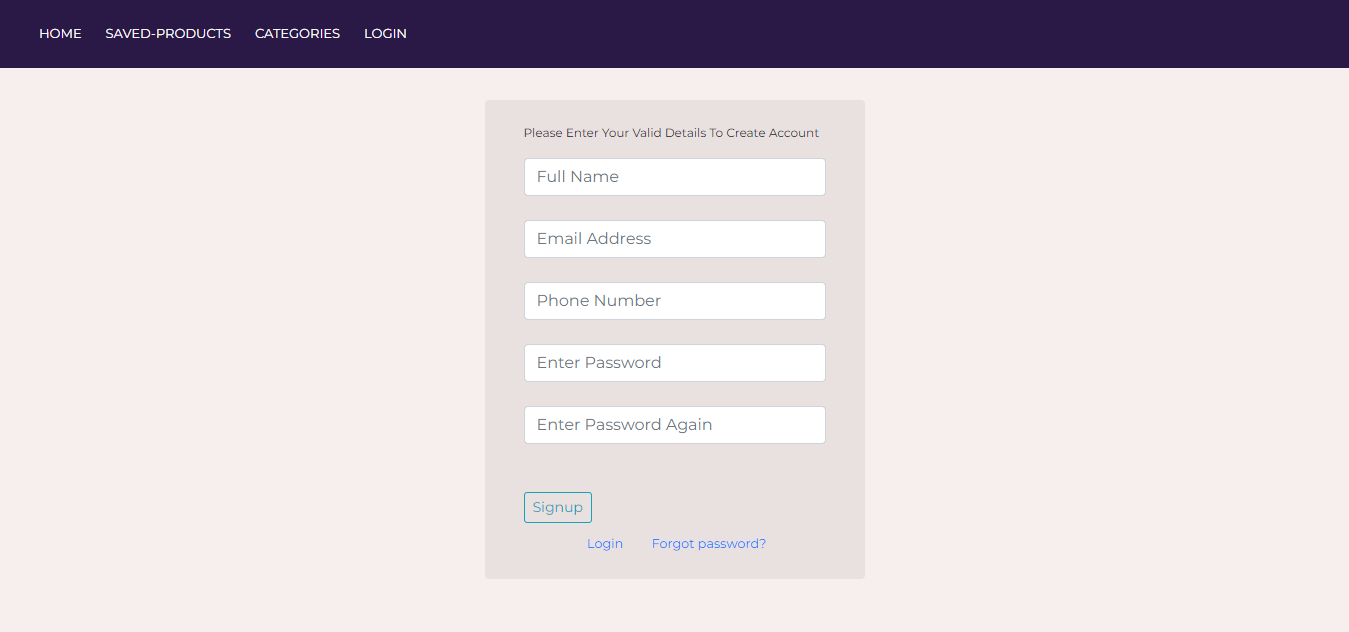


Figure 4.11 Signup page

4.4.6 Login page

The login page is the user authentication page through which users can log into the web scraper application. The login page was designed with a login form for receiving user inputs. The fields on the login page are the email address and password field.

Some pages have been restricted for only authenticated users. One of such pages is the cart page. Only logged in users can view the cart page. Likewise, only authenticated users can save products in the database to be processed later.

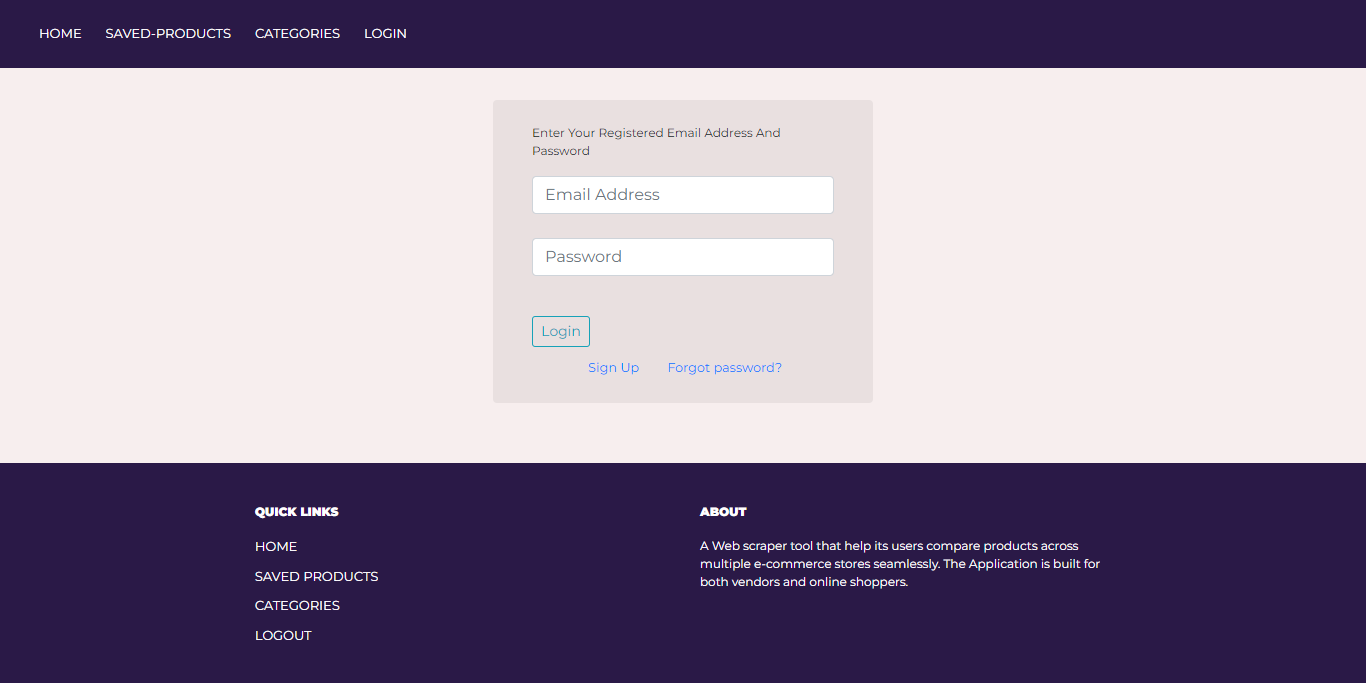


Figure 4.12 Login page

# REFERENCES

Faizul , M.-R. A., Hamzah, A.-R., & Chen, W. (2012). Risk mitigation strategies for software development projects: Professional perspectives. *African Journal of Business Management Vol. 6(9)*, 3528-3538.

Feri, T. E., & Sidiq, M. (2020). Analysis of Sales System Implementation in Primary Cooperative Tribuana II. *International Journal of Computer and Information System (IJCIS)*, 63.

Sebastian, A., Kuriakose, R., & Mariam , S. V. (2016). NEO4J, SQLite AND MySQL for hospital localization. *Advanced Computing: An International Journal (ACIJ), Vol.7, No.3*, 29.