Secure Inventory Management System

Aaron Anthony(10518734)
Clinton Delvin(10531003)
Padmaraju GM(10522769)
MSc.Information Systems with Computing
Dublin Business School

CONTENTS

Secure Inventory Management System

Abstract—**The purpose of this project is to highlight the versatility and functionality of flask along with SQL-alchemy by developing an Inventory and sales management system which efficiently handles the addition, updating and deletion of products and an interface for product sales. The in-depth explanation of all the technologies and its integration has been described below.**

I. Introduction

Inventory, A structured, categorised and complete collection of items or objects that give clear insights about the qualitative and quantitative attributes of these objects. In this modern age where e-commerce websites are breaking the internet and consumption for most products have exponentially increased, it is of utmost importance to maintain a robust inventory system which is capable of maintaining records like products, quantities, sales and various other transitive data needed to run a business smoothly. Hence, we have developed a robust Inventory management system which runs on flask and is connected to a database that has been implemented using Python SQL Toolkit and Object Relational Mapper: SQL-Alchemy. The front end has been developed using bootstrap and connected to the back-end using flask. The system will be login enabled and depending upon the role of an employee will redirect the user to its appropriate interfaces. The managers will be able to add, update, delete and search products from the database, retrieve it and display it on the front-end. An employee can perform a sale on the till interface.

II. Problem Definition

The **Aim** of this project is to showcase the versatility & pragmatics of utilising Flask along with SQL-Alchemy to deliver a robust inventory management and sales system.
The **Scope** of the project is to deliver a system that will successfully accept the data from the user to insert, update, delete data from the Database. The auto decremented of value of every quantity after a sale happens.

III. Proposed Solutions

For this system we shall be using the following technologies:

A. Flask

Flask is a light weight micro framework that is used to build web applications. 'Micro' does not mean that our whole web application must fit into a single Python file, nor does it mean that Flask is lacking in functionality. The 'micro' in micro framework means Flask aims to keep the core simple but extensible.[Flask, 2019] This means that there are no libraries or functionalities that are pre-bound or embedded with the flask core initially. We can choose the functionalities we want to integrate with our flask, that is it won't make many decisions for you, such as what database, security features, etc to use. Although, decisions that it does make are, what templating engine to be used, are easy to change. Everything else is up to you, so that Flask can be everything you need and nothing you don't.

Flask, by default does not include a database abstraction layer, form validation or anything else where different libraries already exist that can handle that. Instead, it supports extensions to add such functionalities to our application as if it were a core part of flask itself. There are various extensions that are provided for database integration, form validation, upload handling, various open authentication technologies, and more. Flask may be "micro", but it's ready for production use on a variety of needs.[Flask, 2019]

B. SQL-Alchemy

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. This means that SQL-Alchemy allows us to map the regular SQL functions and attributes to object oriented concepts like classes, variables, arrays etc, Thus, allowing SQL to be utilised in pythonic domain. It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.[SqlAlchemy, 2019] We can access the database with simple line of a code like TableName[(select all)]. Thus, improving the userability.

IV. Research Methodology

The entire system is built on the flask framework. For the front end technologies like HTML and Bootstrap have been used and implemented. SQLITE3 has been used as a backend technology. To access the SQLITE 3 database a tool named as SQLAlchemy has been used along with flask. The front-end and the backend of this project has been connected using flask.

Back-End Database: The technology used for the backend is SQLITE3. This database is used to store and retrieve the data that has been stored in the backend and is displayed in the front-end. The SQLITE3 database has been accessed with the help of a flask library called SQLAlchemy. With the help of the library the database can be accessed easily. The following tables have been created in the database. A) Users B) Product The attributes under the users table are id,email,first$_name, last_name, password_hash.Intheuserstabletheidco$

Front-End : The front-end has been built using HTML, BootStrap on Flask micro web framework. We have also created a virtual environment where all the important libraries are imported and installed. The completed micro web framework is ran in that specific virtual environment only as all the required packages are already installed in that environment for flask to run properly. Within this virtual environment libraries like login user,login required,logout user, render template, redirect, request, url for, flash,abort, generate password hash, check password hash are used. A flask environment enables one to create an HTML template that can be imported into different HTML files which help us achieving the DRY concept which states “Do Not Repeat Your Self” an important aspect in programming. Application has a message template which helps system in displaying Flash Messages to the user on the HTML page, a navigation template which provides users a navigation bar to navigate into different HTML pages. The complete bootstrap and formatting is used in the base.html file and then all the formatting is then imported into all the other html file using the keyword called as EXTENDS. Example:- {¡div class=”jumbotron”¿ {

V. Implementation and Analysis

Based on the requirements of the secured inventory management system the important features include:
A) Register Page: As one of the main functional requirement, a new user must be able to register into the system so that he/she can login and do his/her duty. To register the new user must enter his email, username, first name, lastname and password fields. Once the data is entered in the html form then this details are further validated in the below criteria: • The email address must be unique and if the same email address already exists prompt in the sqlite3 database then a popup will appear to the user stating that the email address already exists, please use a different email address to register a new user.
• Username that has been entered must be unique in the database, if the username is already present in the database, prompt the user to enter a different username as it is already taken by someone else.
• If the password entered in “Password” and “confirm password” are not the same then user will be prompted stating that, “Passwords do not match! !! Please enter the same password in both the fields.”
If all the above, check are successfully validated then the user data is then inserted in the user database and the password is being hashed before storing it into the database. To hash the password, “generate password hash” function from the “werkzeug.security” module is used. Once the user is registered successfully then the user will be redirected to the login page directly.Ref[Fig.1]
B) Login Page: Once the user is registered, then the user will be directly redirected to the login page, or if the user is registered already then he can navigate to the login page directly and then login. Once the user has entered the username and password to login into the system and then clicked on Submit. The form will be submitted and validations will be done using the POST method used in that page. The POST request captures the data entered using the “request.form” method. The entered user name and password will then be validated against the data that is present in the database.

Below are the scenarios that will be taken into consideration for implementing the login page:

Incorrect Password: If the username entered is correct then the password will be taken into consideration. Since the password that has been stored into the database is in the hashed format, the password that is in the database will first be unhashed and then the entered password will be compared. If the unhashed password is same as that of the entered password then the login will be successful or else the user will be prompted stating that,” The entered password is incorrect !!! Please try again “ and will be redirected to the same page again.[Ref. Fig.3]

Login Successful: If the user is successfully authenticated, system authorizes the user to login and access all the other functionalities of the page. A select query is also fired that will grab the entered username in the database and will display a message to the user stating,” Congrats! You are logged in User name !”. This message will be displayed in the redirected page that is the welcome page. [Ref.Fig2]

C) Product Page: The product page will consist a list of all the products that has been entered into the system along with its complete product details. This page runs a select query to grab all the products that are present in the product table and then all the products are displayed to the authenticated user. [Ref.Fig.4] **D) Add New Product:** In this tab of Add New Product any new product can be added. Once the required details like the name of the product, category, description, barcode, price and quantity has been entered then a insert query will be performed after validating that the barcode is unique or not. If the barcode is unique then system will identify that the product is unique and insert query will be performed on the product table and a new product will be added this way. This input will be taken via an HTML form using a POST request where the data is being stored in a variable named as “request.form” then this data is being retrieved from “ request.form” variable and then after validation an insert query is performed. If the entered barcode already exists then the user will get a prompt stating that, ”Product already exists, Try with a new barcode or check if the barcode is correct” and if the entered details are correct then the user will be redirected to the Products page where the list of all the products will be displayed or else if the details are incorrect then the user will be redirected to the same page so that he/she could add the new product with the correct details.

E) Edit a product: Once the user has added a product, he/she can see all the products that is present in the inventory. But if a user has to edit the quantity of a particular product based on the barcode then navigation to the “Update existing product” has to be done. Over here the barcode has to be entered and then new quantity can be set over here.
Once the user has updated his product , a “POST” request is made into the system and the same data is updated in the database using an “Update” query, as we are not passing a new data into the database.
If a barcode entered is incorrect or not found in the database then the user will be prompted stating, ”Barcode entered is incorrect or does not exist in the database”.
If the entered barcode is incorrect then the update query is performed on the selected product and then the user will

be redirected to the products page where the list of all the products will be displayed.[Ref.fig.5]

F) Delete A Product: Users may want to delete their procut, in that scenario a delete method has been used which takes the barcode as the parameter, once the user selects the delete option a delete query a is sent to the database and that corresponding entry is deleted from the database and the user is flashed a message stating, "Selected Product is successfully deleted" and the user is then redirected to the home page. [Ref Fig.6]

Search a product: User may want to search for a particular product based on barcode or name or category. To make that easy an html form is created where the product details could be entered and then the form will send the data that is entered by the user to app.py file. From the app.py file a select query will be fired from to the database where a list of products based on barcode, name or category will be fetched and will be displayed back to the same search.html page. [Ref.fig.7]

G) Sell Product: User may want to remove the stocks from the inventory and send that particular stock to a different store or to a different place. At such a scenario the barcode of the product has to be entered with the quantity to subtract from the total quantity on the html form. If the entered details are correct then the follow query will be then executed: data2=db.engine.execute(”select quantity from product where barcode=?”,Barcode)
names = [row[0] for row in data2]
Total=names[0]-Quantity
data=db.engine.execute(”update product set quantity=? where barcode=?”,Total,Barcode)[Ref.fig.8]

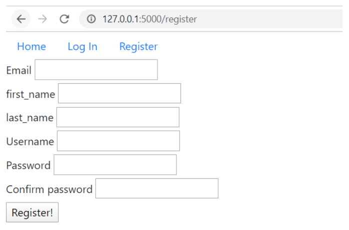
VI. Conclusions and Recommendations

In Conclusion, the system developed categorically stores, deletes, updates and maintains a structured collection of product details and records. We were successfully able to add, delete, modify any product and its attributes. The system was extremely secure due to the hashing technology as well as the security key feature from the library which ensures that any data being sent from the front end will always be encrypted and forwarded to the back end. Hence, there will be no loss of data of any kind.

Appendix A
SCREENSHOT

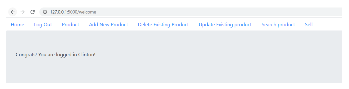
Here are the screenshots of all the pages and few codes:

Fig.1:



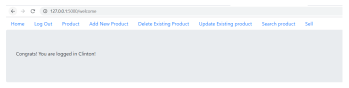
Email
first_name
last_name
Username
Password
Confirm password
Register!

Fig.2:



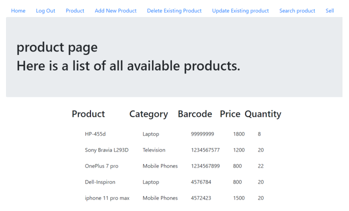
Congrats! You are logged in Clinton!

Fig.3:



Congrats! You are logged in Clinton!

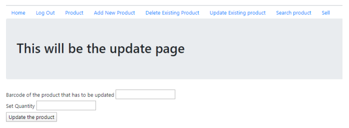
Fig.4:



product page
Here is a list of all available products.

Product	Category	Barcode	Price	Quantity
HP-455d	Laptop	99999999	1800	8
Sony Bravia L293D	Television	1234567577	1200	20
OnePlus 7 pro	Mobile Phones	1234567899	800	22
Dell-Inspiron	Laptop	4576784	800	20
iphone 11 pro max	Mobile Phones	4572423	1500	20

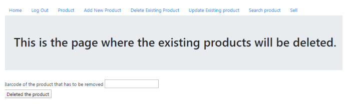
Fig.5:



This will be the update page

Barcode of the product that has to be updated
Set Quantity
Update the product

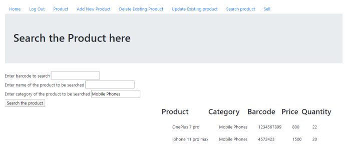
Fig.6:



This is the page where the existing products will be deleted.

Barcode of the product that has to be removed

Fig.7:



Search the Product here

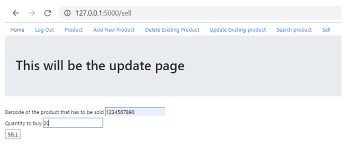
Enter barcode to search

Enter name of the product to be searched

Enter category of the product to be searched Mobile Phones

Product	Category	Barcode	Price	Quantity
OnePlus 7 pro	Mobile Phones	1234567899	800	22
iphone 11 pro max	Mobile Phones	4572423	1500	20

Fig.8:



127.0.0.1:5000/sell

This will be the update page

Barcode of the product that has to be sold 1234567890

Quantity to buy 20

References

[1] Flask, *Flask*.http://flask.palletsprojects.com/en/1.1.x/foreword/. Accessed on: Dec 2, 2019

[2] SQLAlchemy, *SQLAlchemy* https://www.sqlalchemy.org/. Accessed on: Dec 16, 2019.