Faculty of Electronics and Computer Technology and Engineering

DEVELOPMENT OF WEB APPLICATION BASED INVENTORY SYSTEM USING JAVASCRIPT AND PYTHON FOR STOCK MANAGEMENT EFFICIENCY IN BUSINESSES

NG KAH YAU

Bachelor of Computer Engineering Technology (Computer Systems) with Honours

2024

DEVELOPMENT OF WEB APPLICATION BASED INVENTORY SYSTEM USING JAVASCRIPT AND PYTHON FOR STOCK MANAGEMENT EFFICIENCY IN BUSINESSES

NG KAH YAU

A project report submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Computer Engineering Technology (Computer Systems) with Honours

Faculty of Electronics and Computer Technology and Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJUTERAAN ELEKTRONIK DAN KOMPUTER

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek : DEVELOPMENT OF WEB APPLICATION BASED INVENTORY SYSTEM USING JAVASCRIPT AND PYTHON FOR STOCK MANAGEMENT EFFICIENCY IN BUSINESSES :

Sesi Pengajian : 2023/2024

Saya NG KAH YAU mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

☐	**SULIT***	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD***	(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
✓	**TIDAK TERHAD**	

Disahkan oleh:

[illegible]

(TANDATANGAN PENULIS)

[illegible]

(COP

TS. IMRAN BIN HINDUSTAN
Pensyarah
Fakulti Teknologi Dan Kejuruteraan Elektronik Dan Komputer (FTKEK)
Universiti Teknikal Malaysia Melaka (UTeM)

Alamat Tetap: NO 10, JALAN PANDAN INDAH 5/8, PANDAN INDAH 55100 KL

Tarikh: 15/2/2024

Tarikh: 15/2/2024

*CATATAN: Jika laporan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I declare that this project report entitled "DEVELOPMENT OF WEB APPLICATION BASED INVENTORY SYSTEM USING JAVASCRIPT AND PYTHON FOR STOCK MANAGEMENT EFFICIENCY IN BUSINESSES" is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	[illegible]
Student Name	:	NG KAH YAU
Date	:	15/2/2024

APPROVAL

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Engineering Technology (Computer Systems) with Honours.

Signature	:	[illegible] TS. IMRAN BIN HINDUSTAN Pensyarah Fakulti Teknologi Dan Kejuruteraan Elektronik Dan Komputer (FTKEK) Universiti Teknikal Malaysia Melaka (UTeM)
Supervisor Name	:	TS IMRAN BIN HINDUSTAN
Date	:	15/2/2024
Signature	:	
Co-Supervisor Name (if any)	:	
Date	:	

DEDICATION

To my beloved family and friends.

ABSTRACT

This project is about developing an inventory management system that is able to help record and track the number of products in database. An inventory system is based on the concept of database systems which is used in almost all types of large business or cooperates that require to deal with a huge amount of data. A human's memory capability is limited, therefore, the existence of an inventory system will help increase efficiency and productivity of a business. Inventory systems are able to help prevent product and production shortages prevent excess stock and too many raw materials. Moreover, by using a web based system, we are able to allow clients or customers to access the system through a website. The objectives of this final year project would be to formulate a database management system that shows accurate number of items for efficient item tracking purposes, to develop a notification message between management and staff user interface and dashboard for businesses to manage their inventory levels to verify whether the developed web application able to run effectively and smoothly for tablet, laptop and mobile devices. The methodology of project uses incremental development to allow this product to evolve in a shorter development life cycle.Moreover the tools used for this would be microsoft visual studio, one of the most used integrated development environment for programming, python flask ask backend language along side with MySQL while the frontend framework would be Bootstrap. In conclusion, this project was able to to design a database management system that shows accurate number of items for efficient item tracking purposes and able to be deploy the web app in multiple devices. But there are a few limitations in developing a notification due to the need of using Firebase API or Web socket libraries.

ABSTRAK

Projek ini adalah tentang membangunkan sistem pengurusan inventori yang mampu membantu merekod dan menjejaki bilangan produk dalam pangkalan data. Sistem inventori adalah berdasarkan konsep sistem pangkalan data yang digunakan dalam hampir semua jenis perniagaan besar atau bekerjasama yang memerlukan untuk menangani sejumlah besar data. Keupayaan ingatan manusia adalah terhad, oleh itu, kewujudan sistem inventori akan membantu meningkatkan kecekapan dan produktiviti perniagaan. Objektif projek tahun akhir ini adalah untuk merumuskan sistem pengurusan pangkalan data yang menunjukkan bilangan item yang tepat untuk tujuan penjejakan item yang cekap, untuk membangunkan mesej pemberitahuan antara pengurusan dan antara muka pengguna kakitangan dan papan pemuka untuk perniagaan mengurus tahap inventori mereka dan mengesahkan sama ada aplikasi web yang dibangunkan dapat berjalan dengan berkesan dan lancar untuk tablet, komputer riba dan peranti mudah alih. Metodologi projek menggunakan pembangunan tambahan untuk membolehkan produk ini berkembang dalam kitaran hayat pembangunan yang lebih pendek. Lebih-lebih lagi alat yang digunakan untuk ini ialah studio visual microsoft, salah satu persekitaran pembangunan bersepadu yang paling banyak digunakan untuk pengaturcaraan, kelalang python bertanya bahasa belakang bersama MySQL manakala rangka kerja bahagian hadapan ialah Bootstrap. Kesimpulannya, projek ini dapat mereka bentuk sistem pengurusan pangkalan data yang menunjukkan bilangan item yang tepat untuk tujuan penjejakan item yang cekap dan dapat menggunakan aplikasi web dalam berbilang peranti. Tetapi terdapat beberapa batasan dalam membangunkan pemberitahuan kerana keperluan menggunakan Firebase API atau Web Socket

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, Ts Imran Bin Hindustan a for their precious guidance, words of wisdom and patient throughout this project. The meetings with him has thought me a lot about project development that requires patience and collaboration.

I am also indebted to Universiti Teknikal Malaysia Melaka (UTeM) for giving me a chance to undergo my final year project as one of my final evaluation in my degree journey. Without this project, it would be difficult to prepare myself for the upcoming Pre Employment Internship where we are needed to be part of industrial level projects.

My highest appreciation goes to my parents and family members for their love and prayer during the period of my study. An honourable mention also goes to my mother for all the motivation and understanding. Their words of encouragement has always been my source of motivation in doing my best everyday.

Finally, I would like to thank all my friends at Universiti Teknikal Malaysia Melaka (UTeM), the faculty members, as well as other individuals who are not listed here for being co-operative and helpful throughout this final year project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

This project is about developing an inventory management system that is able to help record and track the number of products in a database. An inventory system is based on the concept of database systems which is used in almost all types of large business or cooperates that require to deal with a huge amount of data. A human's memory capability is limited; therefore, the existence of an inventory system will help increase efficiency and productivity of a business [1]. Inventory systems are able to help prevent product and production shortages prevent excess stock and too many raw materials. Moreover, by using a web-based system, we can allow clients or customers to access the system through a website. Then, they can know the number of stocks as well to help ease the customer's time if they desire to obtain a certain product. Companies find new potential for themselves in efficient stock management when implementing web-based inventory management solutions. Additionally, they can implement the systems with carefully chosen features so that they fully satisfy their business requirements. [2].

1.2 Usage of manual inventory systems is still widely practiced.

There are still a lot of startup businesses that use manual inventory systems. Manual inventory systems mainly include hardcopy documents to keep track of their warehouse items and updating the inventory list by hand [3]. An organization is said to be using manual inventory system if their staff members needed to know the number of a specific item, they enter the storage room to physically calculate it. Moreover, staff will repetitively try to update the stock levels to their management which results in time consumption. To address this issue, it is important to track inventory accurately and efficiently for a business. Therefore, a web-based inventory system with a database would help update inventory efficiently and accurately in a business [4]. Manual tracking offers a more affordable option to inventory tracking software, which new and small firms may not be prepared to spend in. These techniques are frequently employed by companies with less inventory that can be easily tracked without the use of tracking software. A web-based inventory system allows tracking order, analyzing recent changes in inventory by using forecasting inventory techniques, QR code scanner for quicker product identification and managing different warehouse inventory levels under the same database. [5]

1.3 Problem Statement

Manual managed inventory systems still exists in lots of places. Manually managed inventory systems are considered to be too slow and inefficient. Many organizations still retain manual inventory systems due to low cost and implementing a web based inventory system may include subscription and development fees. This creates a challenge to business to change their approach in dealing with inventory. Moreover, miscommunications often occur in managing inventory systems. A web based inventory system will help a lot of businesses in improving efficiency and productivity.[6]

1.4 Project Objective

The main aim of this project is to propose a systematic and effective methodology to develop a web-based inventory system. Specifically, the objectives are as follows:

a) To design a database management system that shows accurate number of items for efficient item tracking purposes.
b) To develop a notification message between management and staff user interface and dashboard for businesses to manage their inventory levels.
c) To verify whether the developed web application can run effectively and smoothly for tablet, laptop, and mobile devices.

1.5 Scope of Project

The scope of this project are as follows:

a) Development of frontend interface using HTML and Bootstrap. This includes the skeletal frame provided by HTML and the user interface libraries by Bootstrap.

b) The backend side of web app is written in Python framework Flask. It will interact with the front end and database of the project.

c) The database that is used would be MySQL. It will store item information and quantity.

d) The system consists of a login page for user authentication which then will direct the user to the inventory of items.

e) Only admin will be allowed to view tables from both store 1 and store 2 for statistical reviews.

f) The system allows management staff to carry out CRUD (Create, Read, Update and Delete) operations from the database.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Inventory management is one of the most crucial aspects of operation management. Considering that inventory is one of the major financial assets of a company that can indirectly affect profitability, it is also a vital component of management. Businesses with effective inventory management can generate total profit, which allows working capital to be improved, production, and customer happiness. An effective inventory management strategy may have a substantial impact on the performance of the company in the retail sector [7]. An inventory management system is a collection of techniques and methods for managing and controlling inventories. It depends on the goals and scale of the company thus making it to be able to be utilized in a wide range of methods from small scale to large scale systems. Moreover, an inventory system varies based on the features and practicality of the software as well. System software is a crucial and useful resource for all businesses that deal with inventory. It controls how stock is managed, keeping records of stock levels for all items and gives users access to sales data and analytics, and assists businesses in defining precise safety stock requirements. A company's inventory management is sort of its lifeblood because it drives sales, which then drives profitability as well [8].

2.2 Usage of manual inventory systems is still widely practiced.

Since there may be so many parties involved and their objectives are in conflict, inventory management is complicated. The type of inventory may affect these objectives and key players. A plant or maintenance manager is involved with this form of inventory since spare parts are essential for a manufacturing facility. Likely due to warehouse restrictions, the final inventory level is crucial for a manager of sales as it directly impacts customer service and logistics. Businesses that are tiny or just starting out and are not yet ready to invest in a pricey automatic tracking system prefer traditional surveillance because it may be very cost-effective and doesn't require many instruments to be performed. On the other hand, using manual inventory tracking methods with various programs and spreadsheets is time-consuming, redundant, and prone to mistakes. An integrated central inventory monitoring system with accounting capabilities might be helpful for even small organisations [9].

Reduced stock and lost sales are more likely to occur in businesses with a manual sales and inventory system. Stock reduction happens when the number of items you actually have on hand and the documented stock count don't match. This usually happens at companies where inventory management is done manually. Excessive inventory errors are associated with manual inventory management systems. A manufacturing the business's usage of a manual inventory tracking system increases the possibility of human error. Inventory problems can be expensive as they involve misplacement, waste, and theft [10].

2.3 Database Management System

Every inventory system is built based by a database management system. Nowadays Global Era Management Information systems, which produce output (output) utilizing input (input) and various procedures necessary to achieve specific goals in an organizational activity, are an integral aspect of a business. Software that can be used to define, construct, manage, and control database access is known as a database management system [11].

There are many types of databases such as Relational Database, Operational Database, Database Warehouses, Distributed Databases and End-User Databases. Relational databases will allow data to be stored in multiple tables. Therefore, tables are related to each other with several key fields. Operational databases are mostly used in finance where data is generated widely. It contains information such as customer databases, personal database and inventory databases. Database warehouse can keep data for years, which is useful to study key trends that are taking place. Moreover, distributed database usually involves organizations of different locations and lastly end-user databases involve spreadsheets, presentations, word files, notepads and downloaded files [12].

2.4 Web Application

Web applications are programs that allow for user input and program state change and comprise a Web server, network, HTTP, and browser. Since web applications are accessed through a network, downloading is not necessary. Through a web browser like Google Chrome, Mozilla Firefox, or Safari, users can access a web application [13].Web-based apps generally differ from other traditional programs in that they are more secure, extremely useful, reliable, have higher technology, take less time to build, have shorter product life cycles, and require continuous maintenance. They are also highly user-friendly and require less ongoing maintenance than traditional programs [14].

2.5 Python Language

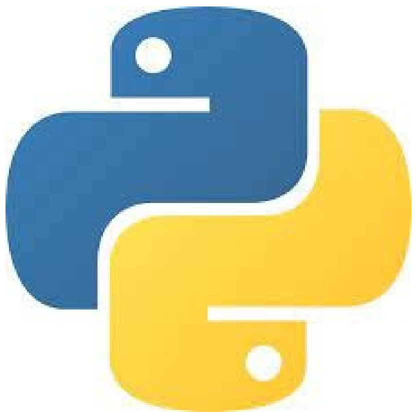


Figure 2.1 Python Logo

In 1989, Guido van Rossum created the Python programming language. Python is a high-level, object-oriented programming language. It is employed in software prototyping, data science, web development, and other fields. Python is a great language to learn to programme with because of its simple syntax. Python is mainly used because it has extensive library support and multiple frameworks, allowing programmers to utilise many of its functions. Moreover, python is also used for server side backend development with the help of popular frameworks such as django and flask while frontend usually can be implemented alongside with Javascript [15].

2.5.1 Comparison between Python backend web application frameworks.

Table 2.1 Comparison between Python backend web application frameworks.

	django	Flask
Language	Python	Python
Project size	Suitable for large companies.	Suitable for smaller scale projects
Support	Has a larger community support by being the more mature framework. Launched in 2005	Has a smaller community support,being the younger framework. Launched in 2010
Features	Has “batteries” to support multiple built in functions	Does not rely on external libraries to perform framework tasks
Type	Full Stack	Light Weight

2.6 Comparison between Journals and this project

Table 2.2 Comparison between Journals and this project.

Title (author/year)	Similarity	Differences	Comment
Computerized Inventory System – Program Development and Execution **(**Shazia Arshad, Muhammad Shoaib and Muhammad Sajjad Khan/2000)	Aims to improve efficiency of database management and uses proper software development techniques	Uses FoxPro language. Foxpro is suitable for database projects.	Foxpro language is less frequent used in the current list of languages.
Design And Development Of An Application For Database Maintenance In Inventory Management System Using Tkinter And Sqlite Platform	Uses Python language Uses Sqlite Platform which is suitable for database management	Tkinter is GUI toolkit made for desktop app. Desktop app will only limit to desktops only	There may be difficulty converting desktop app to web app. Desktop add only available for desktop users. A cross platform method is needed to implement it to

(K Yuvaraj, G M Oorappan, K K Megavarthini, M C Pravin, R Adharsh and M Ashwath Kumaran/2020**)**			convert desktop to mobile platform.
The Financial Impact of Manual Inventory Record Errors **(**Dr. Shamia Wynn Liberty University & Dr. John R. Kuhn, Jr. October 2021)	Emphasized the importance of automated inventory system	Discusses the societal issues of manual inventory system	Technical aspects of development are less discussed due to it being a social science paper.

E-Inventory management system using android mobile application at Faculty of Engineering Technology laboratory stores(Rohana Abdullah, Kek Zi Xiang, Muhammad Ilman Hakimi Chua Abdullah/ May 2018)	Includes software engineering techniques such as Software Development Life Cycle	Focuses on Android mobile app Development and aimed at laboratory equipment	UI/UX improvements is needed as it helps to ease user usability.
Web Based Inventory Management System in Lottemart Solo Baru Adonis Pallas Sutanto(February 2019)	Uses web-based concept Uses python language framework for backend	Django is a different framework compared to Flask. Django has more built in features while Flask is more lightweight for rapid development.	Javascript can be implemented to improve user experience

Development of Warehouse Inventory Management System (Muhammad Aniq Mohd Aris1, Mohd Zaki Mohd Salikon/ 30 July 2021**)**	Includes database management system concepts such as the Entity Relationship diagram (ERD)	Uses xampp and mysql. Based on the flowchart, it uses SMS . Currently, SMS is less used and more social media chat applications are frequently used.	Adding notifications through the web app and allows communication between user and management
Stock Management System (Muhammad Farid Afif Bin Mohamat Johari**)**	Have simlar objectives as the current project. Uses Software Engineering methodologies in project planning. Allows CRUD processes in app.	Focuses on mobile application but not web development	Mobile application development will make it more difficlt for users from desktop to access as it requires a cross platform technique to create a web app or dekstop app.

Performance improvement of inventory management system processes (Anas M. Atieh, Hazem Kaylani/2015)	Uses mySQL as the backend database	ASP.Net is used as their web framework	Software tools are less explained.
Automated Inventory (Devendra Kumar, Aditi Audichya, Ambuj Verma, Bhavesh Kumar Sharma, Amit Bohra/ June 2021)	Has a login page that allows user authentication	Includes QR Code as an item scanner. This helps the system to automate the inventory system	Implementation of QR Code will increase cost of production. Final year project's objectives does not include QR implementation

CHAPTER 3

METHODOLOGY

3.1 Project Development

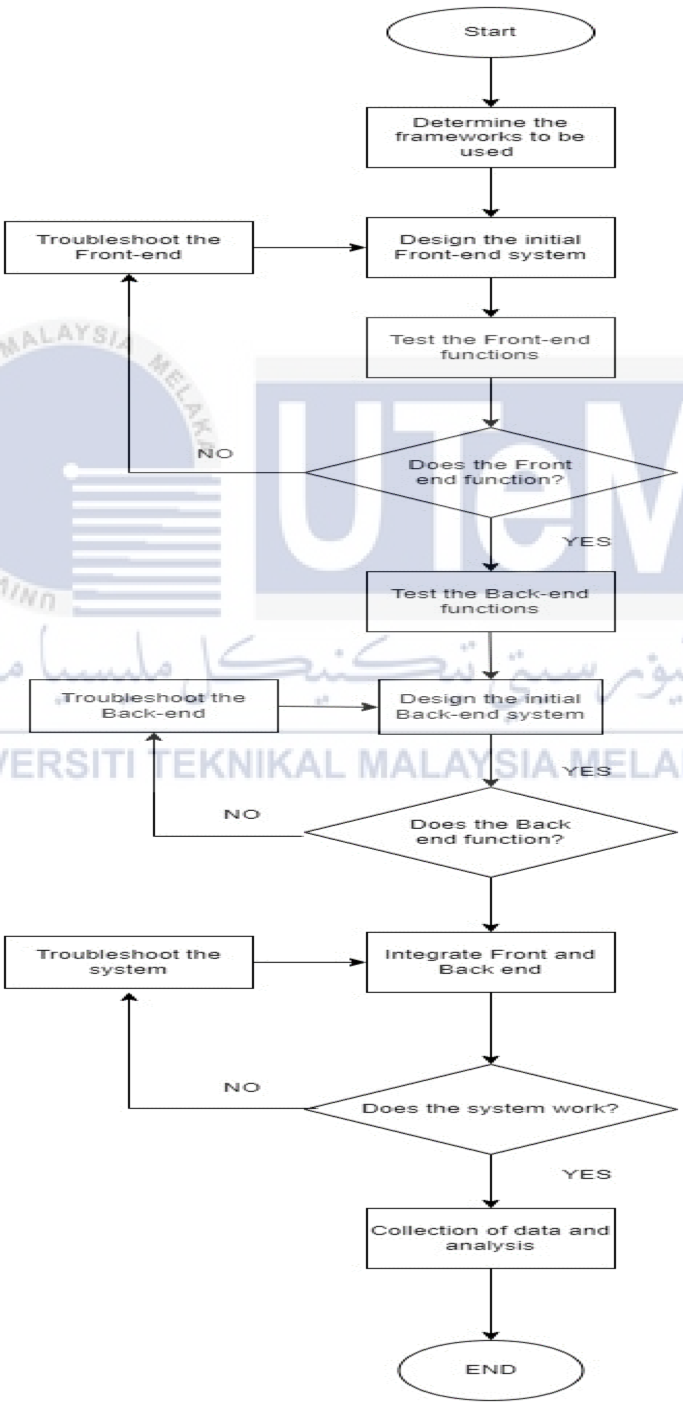

Start
Determine the frameworks to be used
Troubleshoot the Front-end
Design the initial Front-end system
Test the Front-end functions
NO
Does the Front end function?
YES
Test the Back-end functions
Troubleshoot the Back-end
Design the initial Back-end system
YES
NO
Does the Back end function?
Troubleshoot the system
Integrate Front and Back end
NO
Does the system work?
YES
Collection of data and analysis
END

Figure 3.1 Project Development flow chart

Methodology is the techniques, procedure and process needed to carry out a project. Based on figure 3.1, it shows the flow chart of the project development process. Theproject starts with the selection of language and suitable frameworks for this web application. The research was done from a few journals and internet sources provide a lot of programming language tutorials for web application development.

The design of user interface, also known as the front-end will be developed using the languages chosen in the previous step, which is using HTML and Bootstrap. The front end requires multiple languages to provide a good user experience. After designing the interface, a user input test is needed to ensure the web pages can accept and send input to the back-end side. If successful, we will proceed to design the back end, if unsuccessful, troubleshooting will be carried out for the user interface.

The success of front-end design will be followed up by the development of the back-end side of the system. Back-end will include the Python framework FLASK, and it will communicate with the database used, MySQL. The integration of both ends will be tested to make sure the system functions as it is required. The results obtained will be collected and analyzed for accuracy purposes.

3.2 Project Planning

Table 3.1 Gantt Chart for Bachelor Degree Project 1

SEM 1 2023/2024														
PSM 1 Activities	WEEK													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
First meeting with supervisor (title proposal)	🟦													
PSM Briefing by JK PSM	🟦													
Begin Chapter 1 (Introduction)		🟦												
Understanding objectives and problem statement		🟦	🟦											
Research on project scope and background writing				🟦										
Begin Chapter 2 (Literature Review)				🟦										
Review the main title of the project					🟦									

Submission of Progress 1					🟦									
Review related societal issues of project						🟦	🟦							
Comparison of literatures in table form						🟦	🟦							
Begin Chapter 3 (Methodology)								🟦						
Studying on programming language needed for frontend								🟦						
Studying on programming framework needed for backend									🟦					
Research on database needed									🟦					
Test on frontend design										🟦	🟦			
Submission of Progress 2												🟦		
Submission of report to supervisor and panel													🟦	
Presentation of PSM 1														🟦

Table 3.2 Gantt Chart for Bachelor Degree Project 2

SEM 2 2023/2024														
PSM 2 Activities	WEEK													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
First meeting with supervisor for project implementation	🟦													
PSM2 Briefing by JK PSM	🟦													
Begin Database design using MySQL Workbench		🟦												
Testing the database with SQL statements		🟦	🟦											
Integration between MySQL and flask				🟦										
Displaying tables on web frontend				🟦	🟦	🟦								
Meeting with supervisor on frontend development					🟦									
Submission of Progress 1 Report					🟦									
Tested editing product information in project						🟦	🟦	🟦	🟦					

Tested the login system with different username and passwords						🟦	🟦							
Begin using the app from different problems								🟦						
Studying on techniques to enable realtime notification								🟦	🟦	🟦	🟦			
Research on various testing methods									🟦					
Recheck code for any potential bugs									🟦	🟦	🟦	🟦		
Meeting with supervisor for objective discussion										🟦	🟦			
Submission of Progress 2 Report												🟦		
Finalise the results and discussion of report													🟦	
Presentation of PSM 2														🟦

3.3 Software Process Model

In every software development project, a software process model is often used to meet the demands of the clients. It is a method used to design and develop software products. Sometimes, it is often referred to as Software Development Life Cycle (SDLC). This model includes waterfall process model, incremental development and integration and configuration. There are many different software products in the current market, but all of the software follows the same fundamental rules of development. The following processes must be involved in a software development are

i. Software specifications: Functions of software features must be defined and document the expected outcomes of the system to be built.

ii. Software Development: Designing and implementing the system based on the specifications that have been defined.

iii. Software Validation: Ensuring the system meets the demands of the client by testing its functionalities.

iv. Software Evolution: Software product must be modified so that is able to meet the demands of the client.

3.3.1 Incremental Development

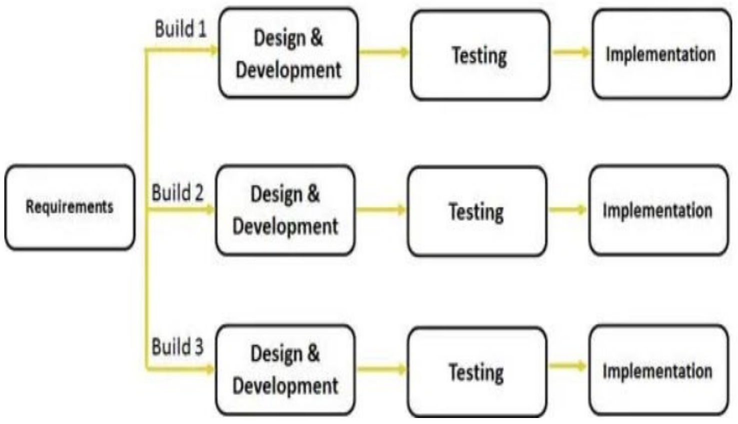


Figure 3.2 Incremental Development

In this project, incremental development of the project is used. Incremental development allows the developer to evolve the product over time. This process allows the developer to break the project into smaller parts known as increments. Each increment will be built based on the previous versions so that the system can be improved. For example, if this project is taking more time to develop than the expected time, the initial progress or system can be first released to the user to be tested. This also allows users to be able to provide feedback for future evolution.

3.4 Web Application Tools

To develop a web application, it is required to involve many different programming languages to build front-end and back-end. In full stack developments, usually the front-end language is determined by HTML and CSS with JavaScript being optional. For backend development, there are a variety of languages to choose from JavaScript, PHP, or python. Over the years, software engineers tried to develop web applications with different languages, making web applications not having a specific type of language to use.

Figure 3.3 Web Development

3.4.1 Microsoft Visual Studio

Microsoft visual studio is an Integrated Development Environment used mainly for developing web apps, web services, mobile apps and so on. For this project, the programming syntax will be written in Microsoft visual studio editor. This IDE allows writing multiple different programming languages and integrating it in the same environment. We can create folders and have multiple programming files under the same workspace. Using python for example requires us to download python extensions from its library so that we are able to use its latest features.

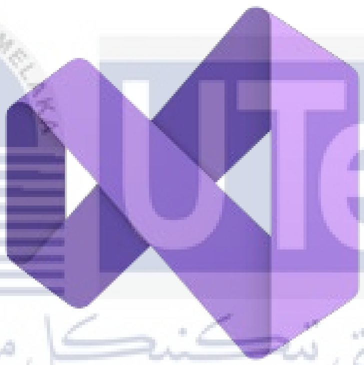


Figure 3.4 Microsoft Visual Studio Logo

3.4.2 Javascript

Figure 3.5 JavaScript Logo

JavaScript is a flexible, multi-paradigm, object-oriented, lightweight, and interpreted programming language. JavaScript was initially developed to add interactivity to online pages. Both HTML and CSS can be modified and updated using JavaScript. Data can be calculated, manipulated, and validated using JavaScript. Scripts are what this language refers to as programs. Scripts are simple text files that don't require any setup to run.

3.4.3 Flask

Figure 3.6 Flask Logo

Flask is a backend framework written in Python used to develop web applications. It is considered light weight by many developers and it's suitable for beginners who want to learn a web application framework. Flask is written in Python; therefore, the syntax is a lot simpler than programming languages such as C++ and Java. We need to install flask if we want to run the flask app. Flask also has a built-in template known as Jinja which allows rendering of html files for front-end development.

3.4.4 MySQL

MySQL will be used for this project in constructing the database of the backend server side. The “SQL” part of “MySQL” stands for “Structured Query Language.” SQL is the most common standardized language used to access databases. By using MySQL, we can use the relational database function. In the relational database, each table has a distinct ID or a special key. This allows the associations between table of data, as the column tables will carry the primary key of the respective rows. A primary key is important to allow database to allow create, read update, and delete operations.

Figure 3.7 MySQL logo

3.5 General Flowchart of Inventory Management System

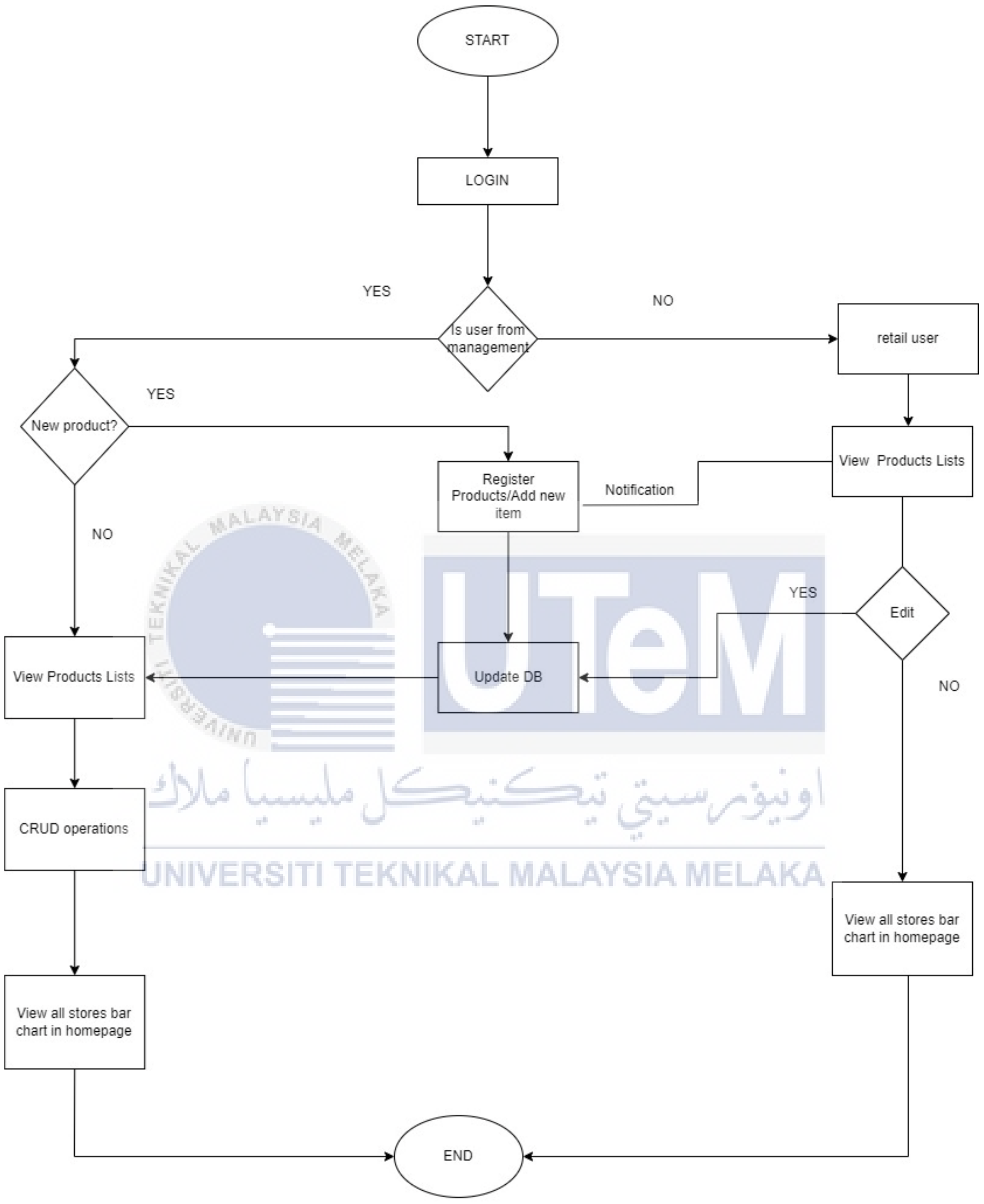


Figure 3.8 General Flowchart of Inventory Management System

Based on figure 3.8, it shows the general flowchart of the Inventory Management System. It shows the start of the process with login page. This allows users to login as management staff or retail staff. If the user from management, it will go to next decision. If there is a new product arrival, it allows management to register it or add it as a new item in the list of product tables.

If a new item is registered it will send a notification to the retail user. Thisprovides a communication link through the web app between management and retail. After this process, it will automatically update the database. As this happens, management can once again view the list once it is being updated. For the admin dashboard, only admins can view the graph of both tables while store user can view their table graphs only in their respective homepage.

The alternative scenario of this flowchart would be retail user to be logged in. Retail user will be directed to the home page to view the table list. Assuming the management and retail user are online, Retail user will receive a notification if there is a registration or update on a new product. If a retail user decides to edit the product numbers, it will proceed using edit operations. If not, the process will end.

3.6 Entity Relationship Diagram

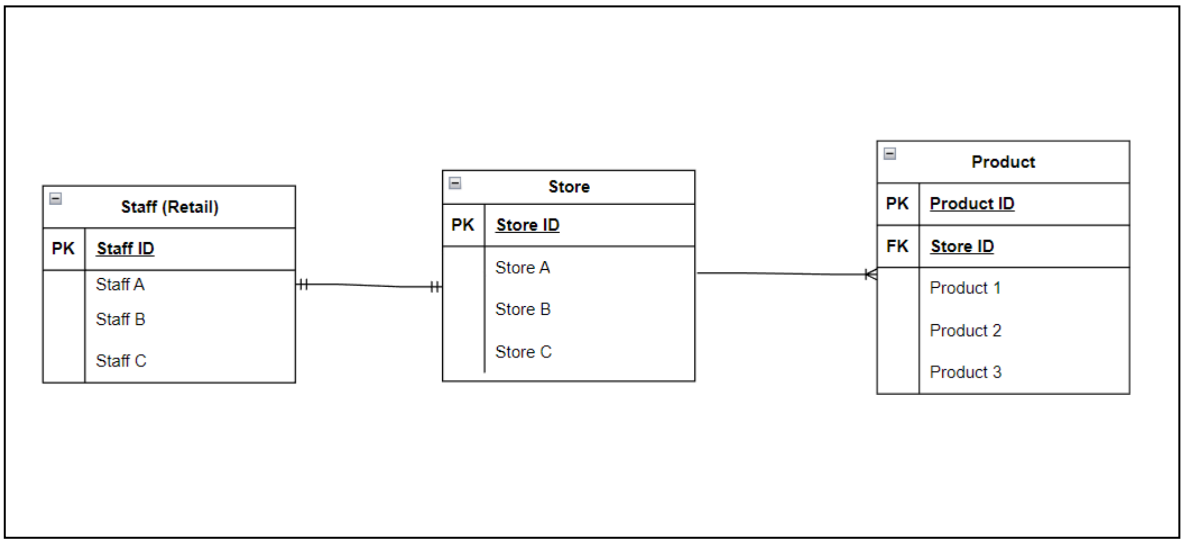


Figure 3.9 Entity Relationship Diagram of staff, store and products

Figure 3.9 above shows the Entity Relationship Diagram for staff, store and product. In this ERD, it can be observed that the staff and store have a one-to-one relation. This means that one retail staff will represent one store (acting as store manager) in this web application. So for example, staff A can manage the system of either store A, B or C and same applies to staff B and staff C.

The store has a one-to-many relationship with product. This means that one store can have many products. For example, store A can have product 1, product 1 and 2 or product 1,2,3. This applies Store B and Store C as well.

3.7 Use Case Diagram

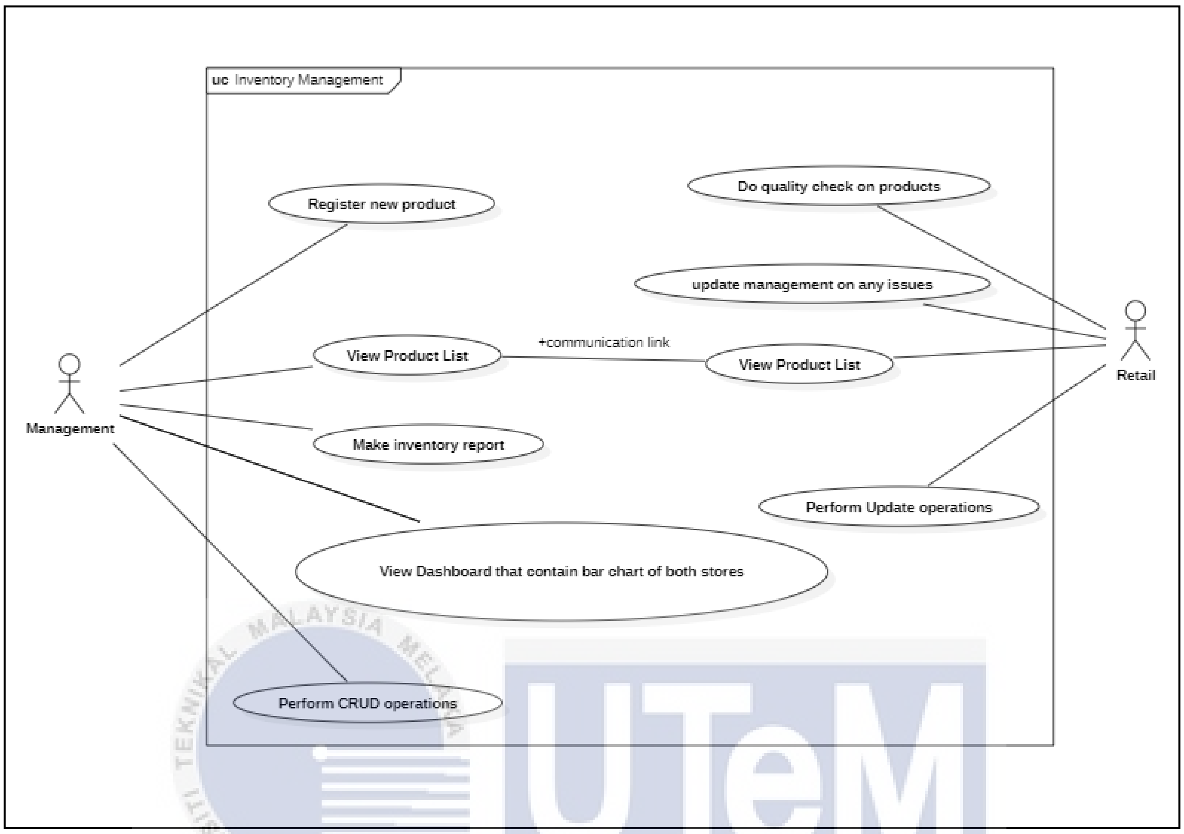


Figure 3.10 Use Case Diagram of staff, store, and products

Use-case diagrams help in capturing system requirements and depict a system's behavior in UML. The scope and high-level functions of a system are described in use-case diagrams. The interactions between the system and its actors are also depicted in these diagrams. Use case diagrams show what the system does and how the actors utilize it.

- Management entity – Register new product, View product list, make inventory report, and view dashboard for both store's bar chart and CRUD operations.

- Retail entity- Do quality check on products, update management on any issues and perform edit operations.

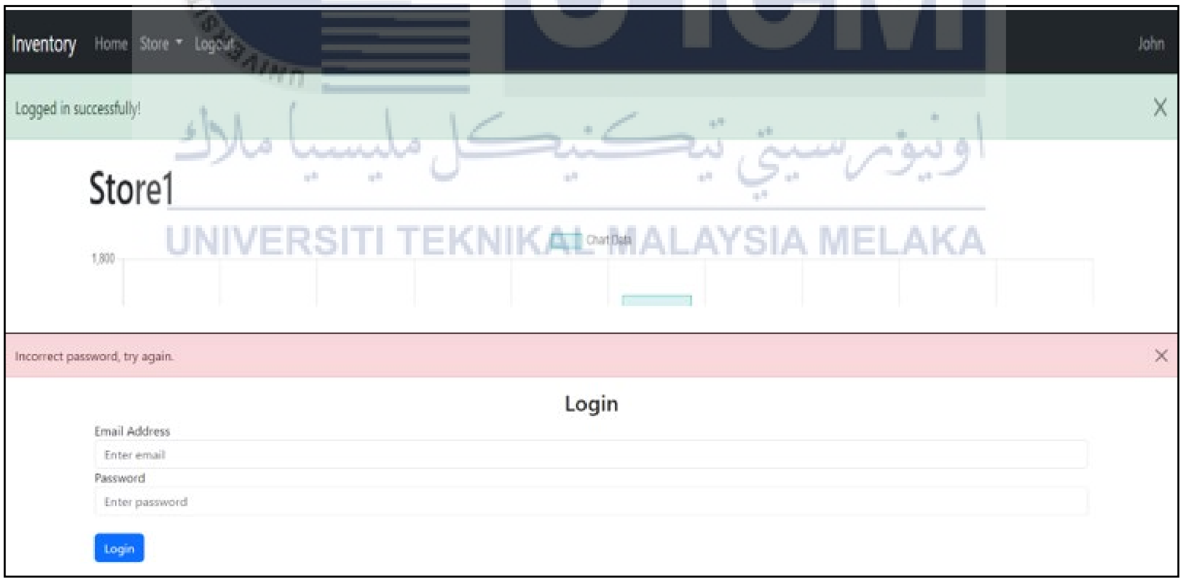
CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter presents the results and analysis of development of web application-based inventory system using JavaScript and python for stock management efficiency in businesses. Result and Analysis of databases will be shown throughout this chapter. Moreover, the design of the web application and the features are included in this chapter.

4.2 User Authentication Test

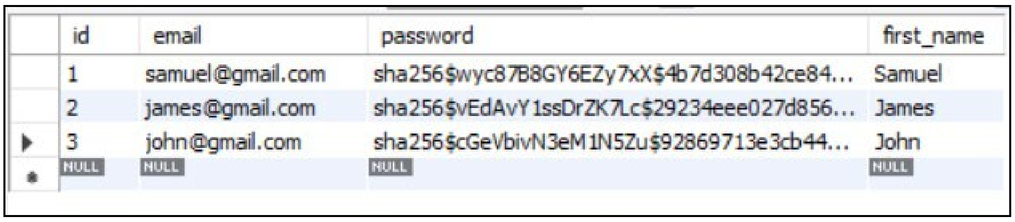


Logged in successfully!

Incorrect password, try again.

Figure 4.1 User login system

Based figure 4.1, it can be seen that the authentication message when login is successful shows a green pop up message “Logged In Successfully” while if failed it shows a red message “Incorrect password. Try again.” This part of the system checks with the database that stores the user email and checks whether the password submitted matches with the hashed password by flask backend.



	id	email	password	first_name
	1	samuel@gmail.com	sha256$wyc87B8GY6EZy7xX$4b7d308b42ce84...	Samuel
	2	james@gmail.com	sha256$vEdAvY1ssDrZK7Lc$29234eee027d856...	James
▶	3	john@gmail.com	sha256$cGeVbivN3eM1N5Zu$92869713e3cb44...	John
●	NULL	NULL	NULL	NULL

Figure 4.2 User Credentials in MySQL

Based on Figure 4.2.2, the password column has the password hashed in sha256 format. During login, the server recalculates the hash value and compares it to the stored hash in the database for authentication. In the following database, Samuel oversees store 1, James is in charge of store 2 and John is in charge of both of the stores making it as the manager. In this system, user id = 3 has the most access to the features of this web application and being the manager of the system.

4.3 Access Level Between Management and Staff

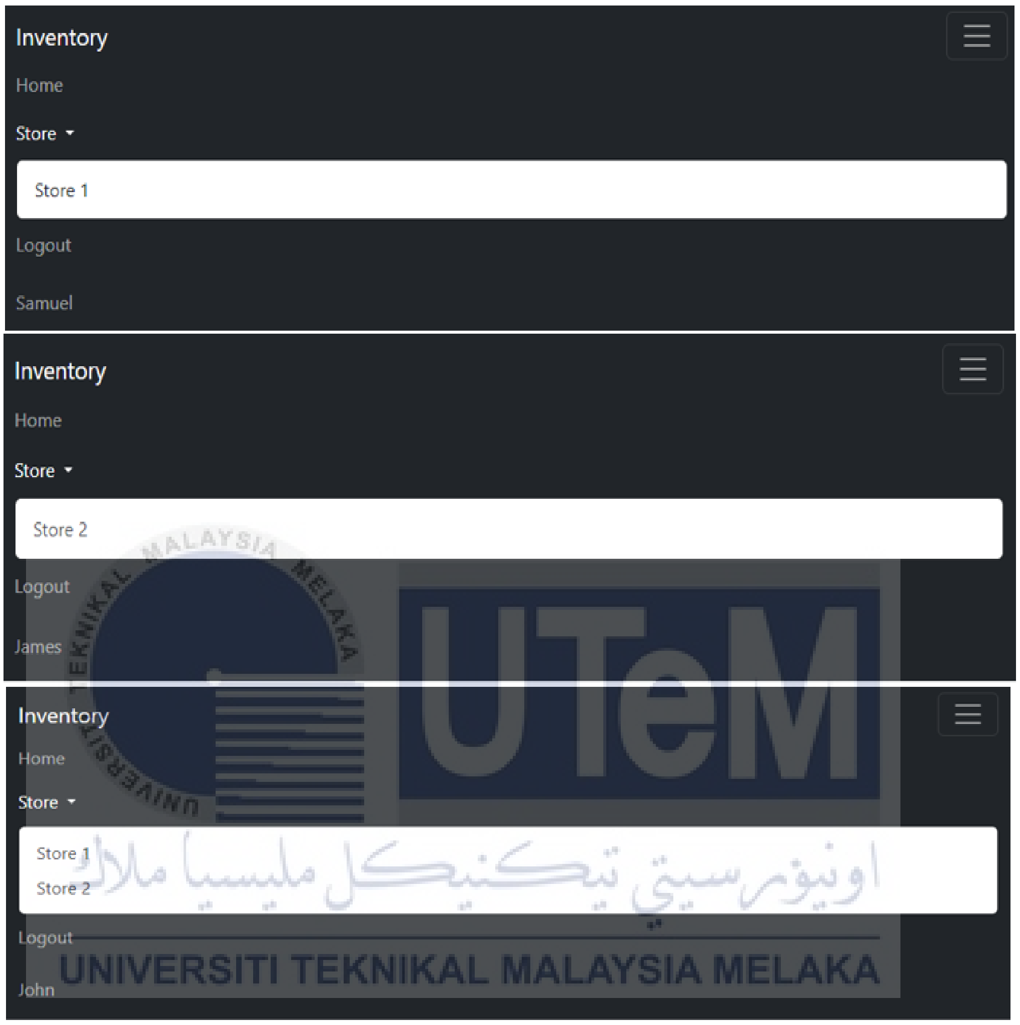
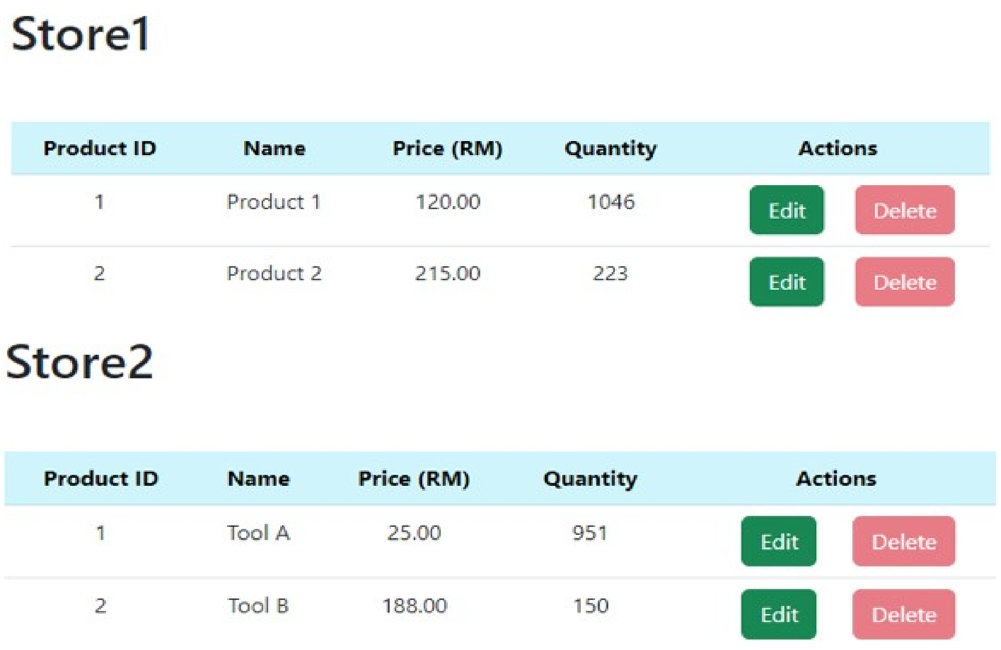


Figure 4.3 Dropdown of Store button when different users log in

In the navigation bar, the user named Samuel with user id = 1 has access to store 1 only while user named James with user id = 2 has access to store 2 only. Lastly, the manager named John at the bottom of the navigation bar with ID = 3 has the access to both store 1 and store 2.



Store1

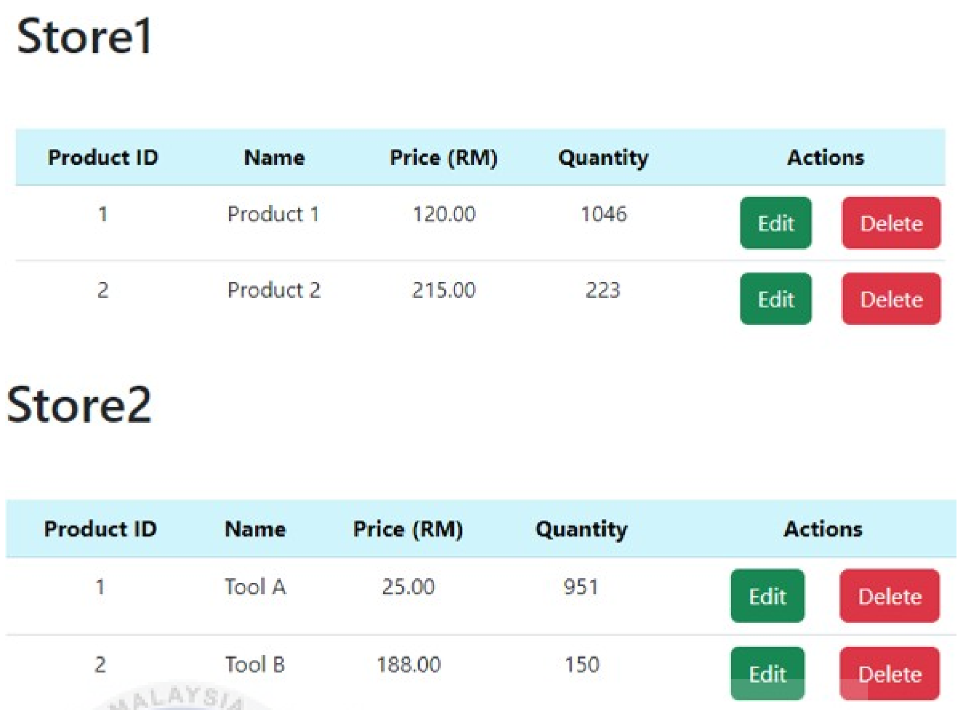
Product ID	Name	Price (RM)	Quantity	Actions
1	Product 1	120.00	1046	Edit Delete
2	Product 2	215.00	223	Edit Delete

Store2

Product ID	Name	Price (RM)	Quantity	Actions
1	Tool A	25.00	951	Edit Delete
2	Tool B	188.00	150	Edit Delete

Figure 4.4 Delete button when user id = 1 and user id = 2 logged in

It can be seen that delete button has a faded red color. This is the disabled featured by html which prohibits the user of id = 1 and id = 2 to press the delete button. The delete button is disabled to prevent unauthorized removal of products without the management's permission.



Store1

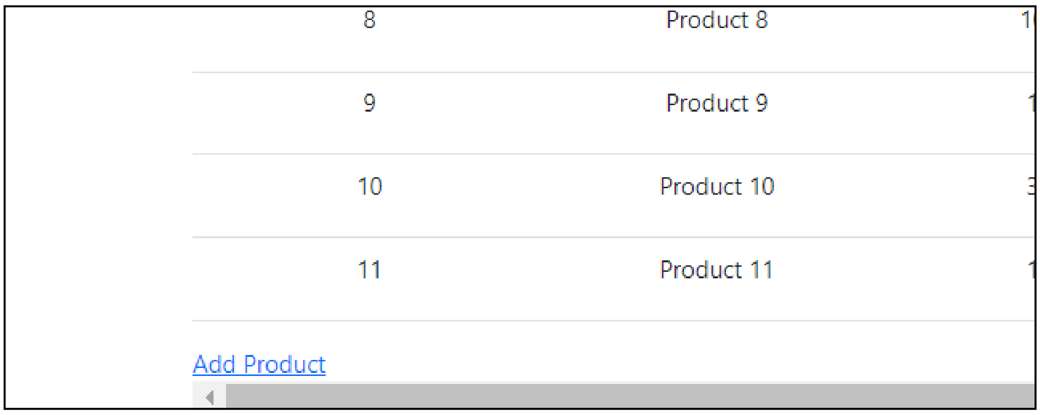
Product ID	Name	Price (RM)	Quantity	Actions
1	Product 1	120.00	1046	Edit Delete
2	Product 2	215.00	223	Edit Delete

Store2

Product ID	Name	Price (RM)	Quantity	Actions
1	Tool A	25.00	951	Edit Delete
2	Tool B	188.00	150	Edit Delete

Figure 4.5 Delete button when user id = 3 logged in

Delete button has a red color without faded attributes. The delete button enables management to delete product row. This is to allow management to have total control of the database.



8	Product 8	1
9	Product 9	1
10	Product 10	3
11	Product 11	1

Add Product

Figure 4.6 Bottom of the web page of store when user id = 3 logged in

There is an "Add Product" link that will direct users to a modal page that will be explained in the features section. Only user id = 3 has access to add new type of product row. In many business settings, a product can be added after being approved by the management. User id = 1 and user id = 2 are not allowed to add product due to their role.

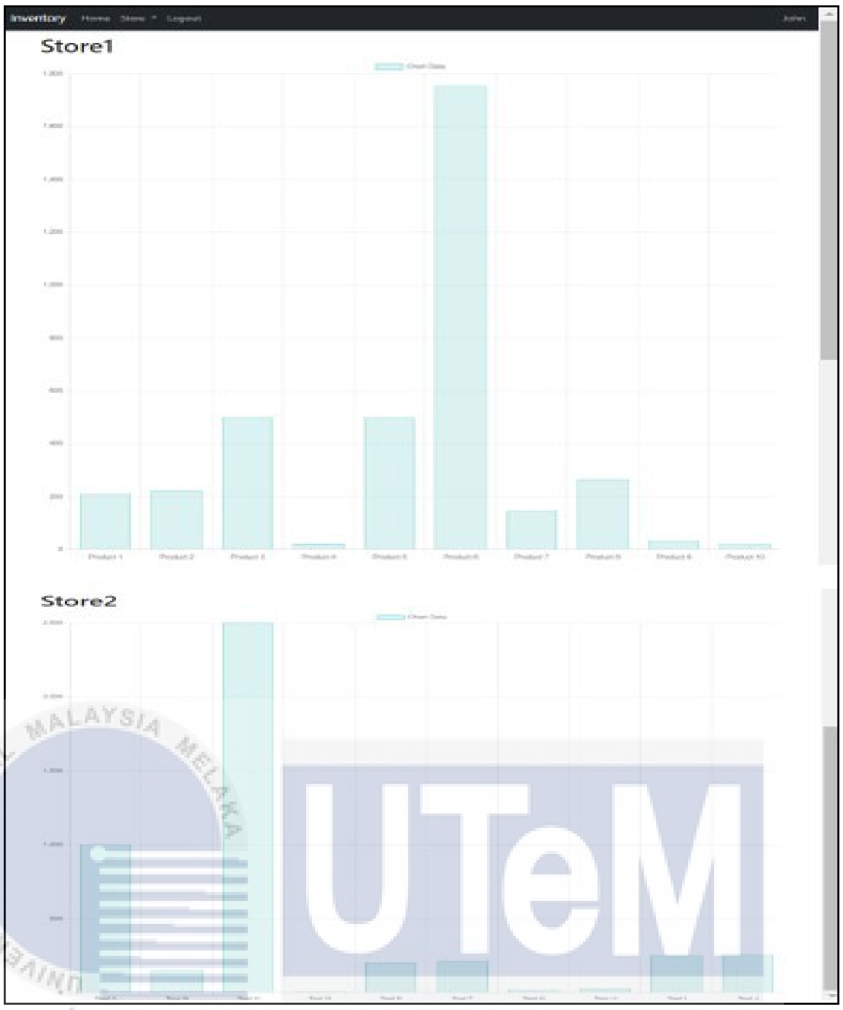


Figure 4.7 Home Page of User Manager named John

For user manager, the manager with user id = 3 will be able to view the bar chart of both of the stores while user id = 1 and user id = 2 are able to view store 1 and store 2 only respectively. This is to provide an overview of the bar chart where management can analyze and breakdown strategies to improve their business.

4.4 Features of the system

4.4.1 Modal Form

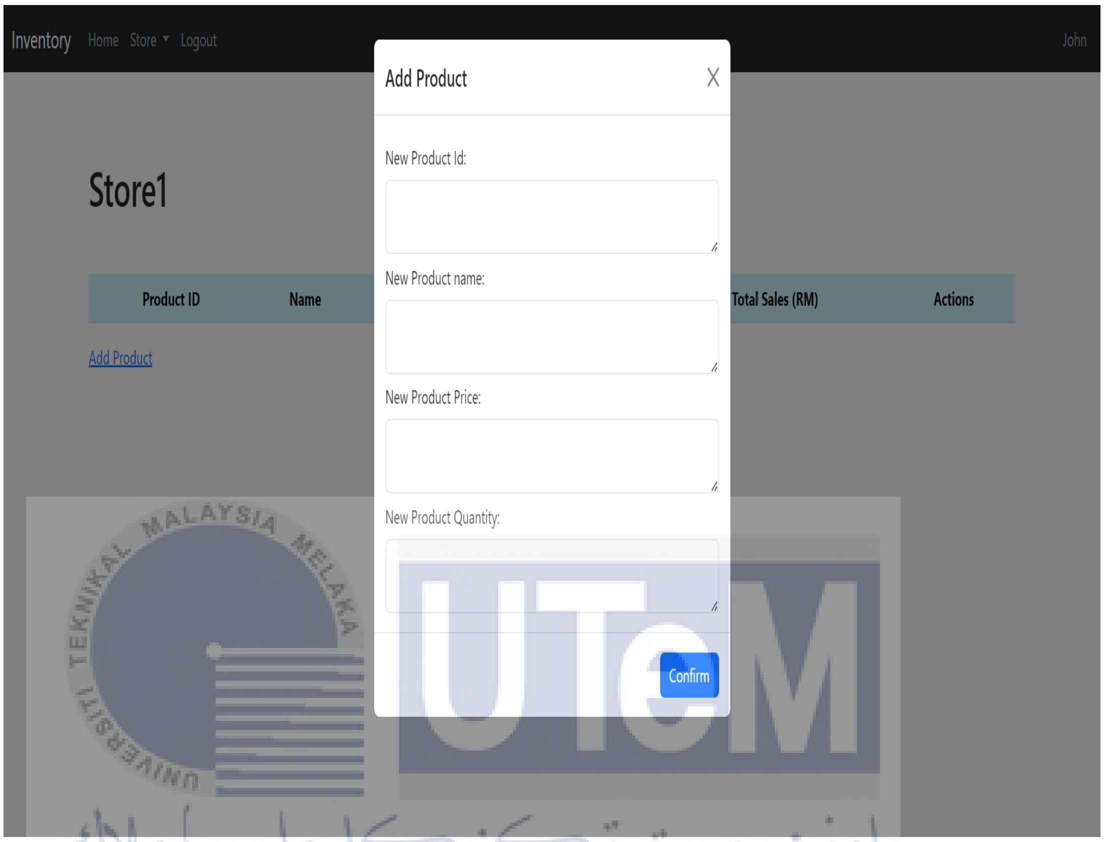
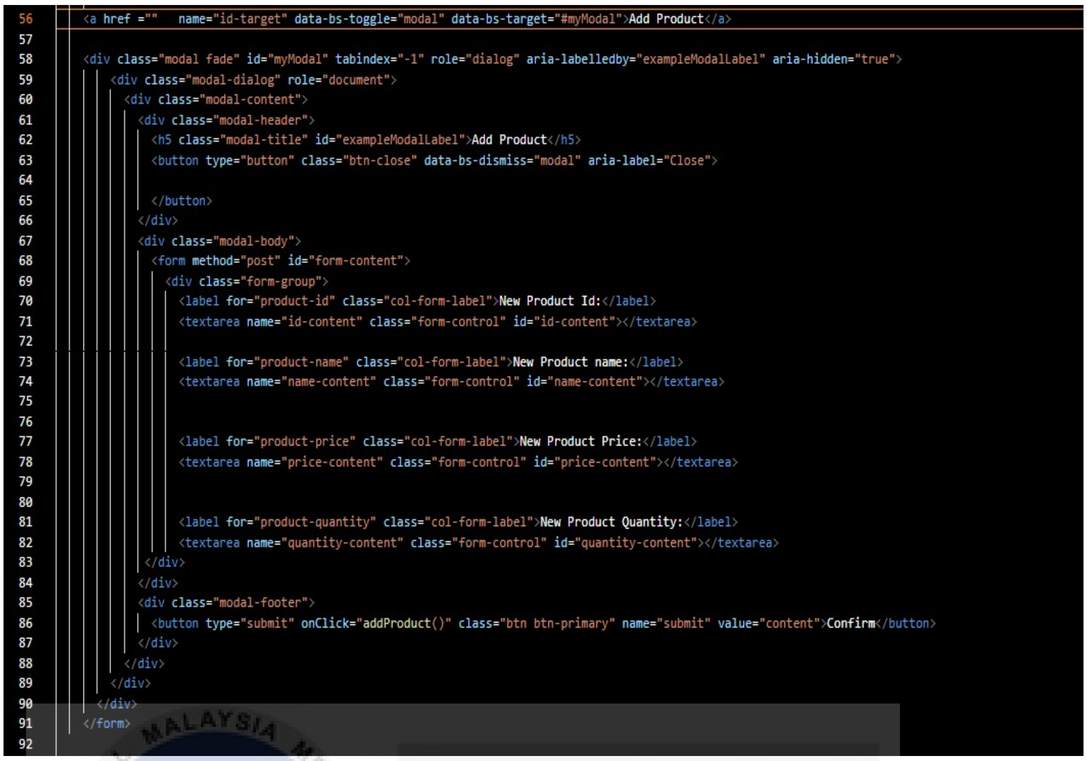


Figure 4.8 Modal Form

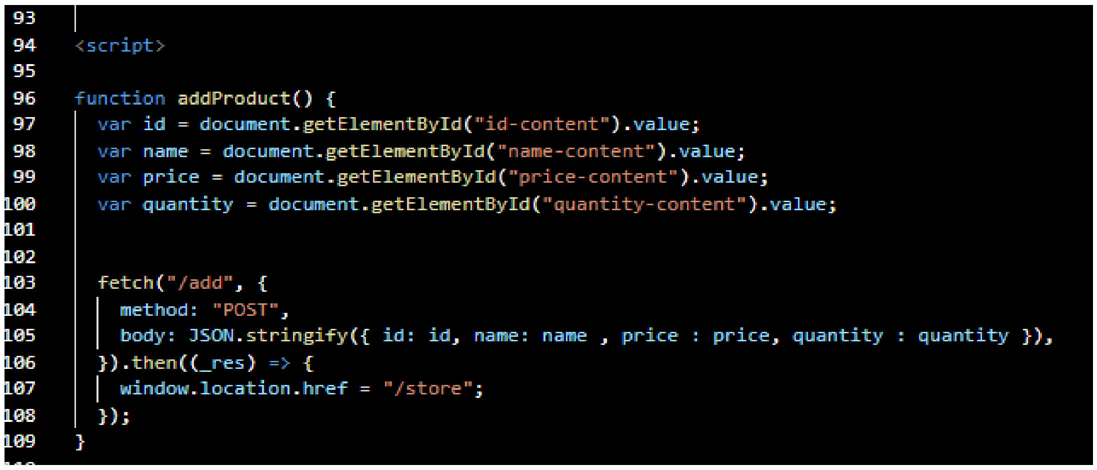
After clicking the add product link highlighted in blue, a modal form is popped out. This is with the help of JavaScript and bootstrap classes. Users are asked to input in the blanks given on the modal form.



```
56  <a href =""   name="id-target" data-bs-toggle="modal" data-bs-target="#myModal">Add Product</a>
57
58  <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
59    <div class="modal-dialog" role="document">
60      <div class="modal-content">
61        <div class="modal-header">
62          <h5 class="modal-title" id="exampleModalLabel">Add Product</h5>
63          <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close">
64
65          </button>
66        </div>
67        <div class="modal-body">
68          <form method="post" id="form-content">
69            <div class="form-group">
70              <label for="product-id" class="col-form-label">New Product Id:</label>
71              <textarea name="id-content" class="form-control" id="id-content"></textarea>
72
73              <label for="product-name" class="col-form-label">New Product name:</label>
74              <textarea name="name-content" class="form-control" id="name-content"></textarea>
75
76
77              <label for="product-price" class="col-form-label">New Product Price:</label>
78              <textarea name="price-content" class="form-control" id="price-content"></textarea>
79
80
81              <label for="product-quantity" class="col-form-label">New Product Quantity:</label>
82              <textarea name="quantity-content" class="form-control" id="quantity-content"></textarea>
83            </div>
84          </div>
85          <div class="modal-footer">
86            <button type="submit" onClick="addProduct()" class="btn btn-primary" name="submit" value="content">Confirm</button>
87          </div>
88        </div>
89      </div>
90    </div>
91  </form>
92
```

Figure 4.9 Code Modal Form

Based on the figure 4.6, it can be in line 56, as the user Manager (id = 3) clicks the Add Product link, it launches the modal form. a href =” ” means that it will not send the user to another page but triggering another type of response such as the popup. In the same line, data-bs-toggle="modal which is a bootstrap class will help trigger the appearance of the modal form.

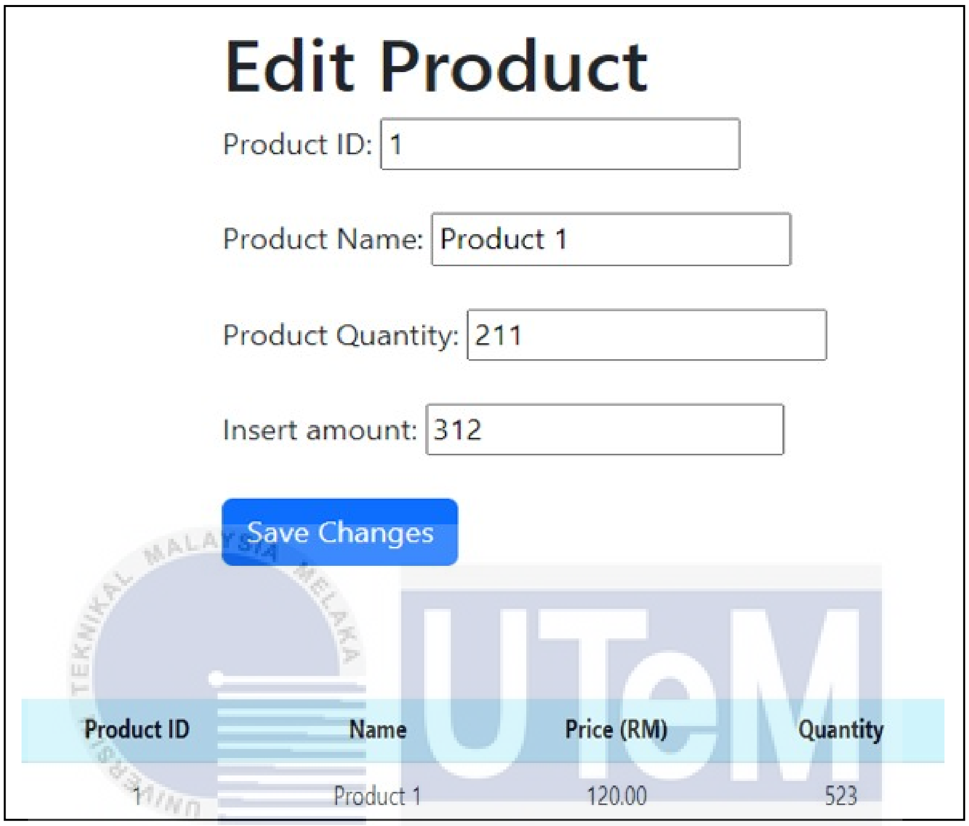


```
<script>

function addProduct() {
  var id = document.getElementById("id-content").value;
  var name = document.getElementById("name-content").value;
  var price = document.getElementById("price-content").value;
  var quantity = document.getElementById("quantity-content").value;


  fetch("/add", {
    method: "POST",
    body: JSON.stringify({ id: id, name: name , price : price, quantity : quantity }),
  }).then((_res) => {
    window.location.href = "/store";
  });
}
```

Figure 4.10 Javascript addProduct function

Figure 4.7 shows a block of JavaScript code responsible for product addition, which can be seen in the function addProduct(). This will receive the inputs in the modal form and assign each of the input to each of the variables such as id, name, price, and quantity. Then it fetches to the end point "/add", the flask backend route. With JavaScript, this enables the user to add the product on the same page and there is no need to redirect the user to another page.

4.4.2 Edit Page


Edit Product

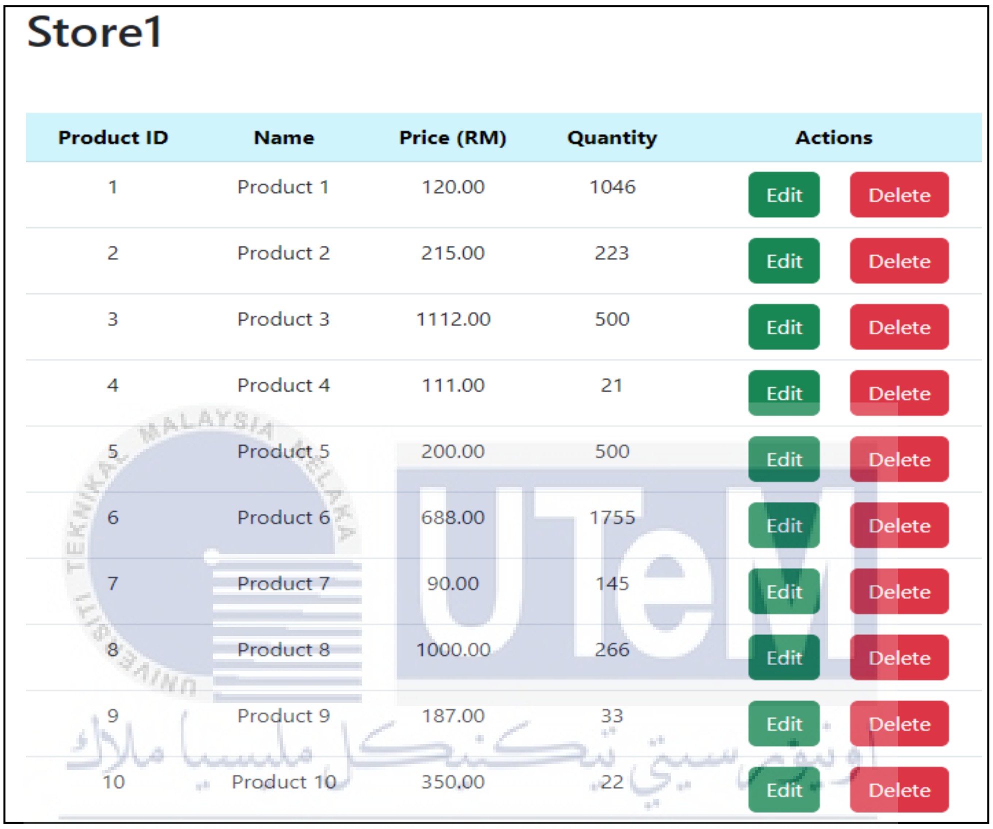
Product ID:	1
Product Name:	Product 1
Product Quantity:	211
Insert amount:	312

Product ID	Name	Price (RM)	Quantity
1	Product 1	120.00	523

Figure 4.11 Edit Product page

If the user presses the edit button in the table, it will return the original quantity 211 to the edit page. It will only allow the user to update the amount and other fields will only be set to read only. This is to prevent unauthorized changes to the product ID and name. As the user inserts the amount 312, it adds up to the existing amount of 211 and results in 523. After pressing save changes the product quantity becomes 523. This indicates there are 523 sold.

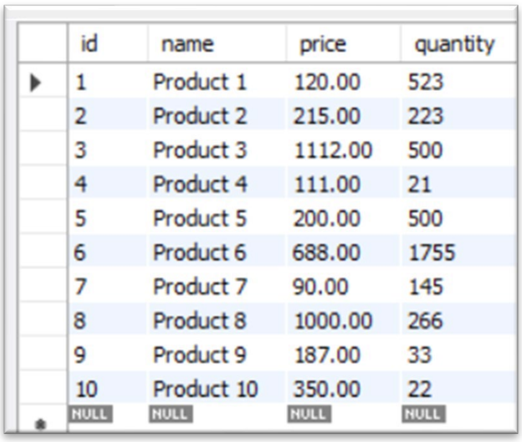
4.5 Accuracy of the database



Store1

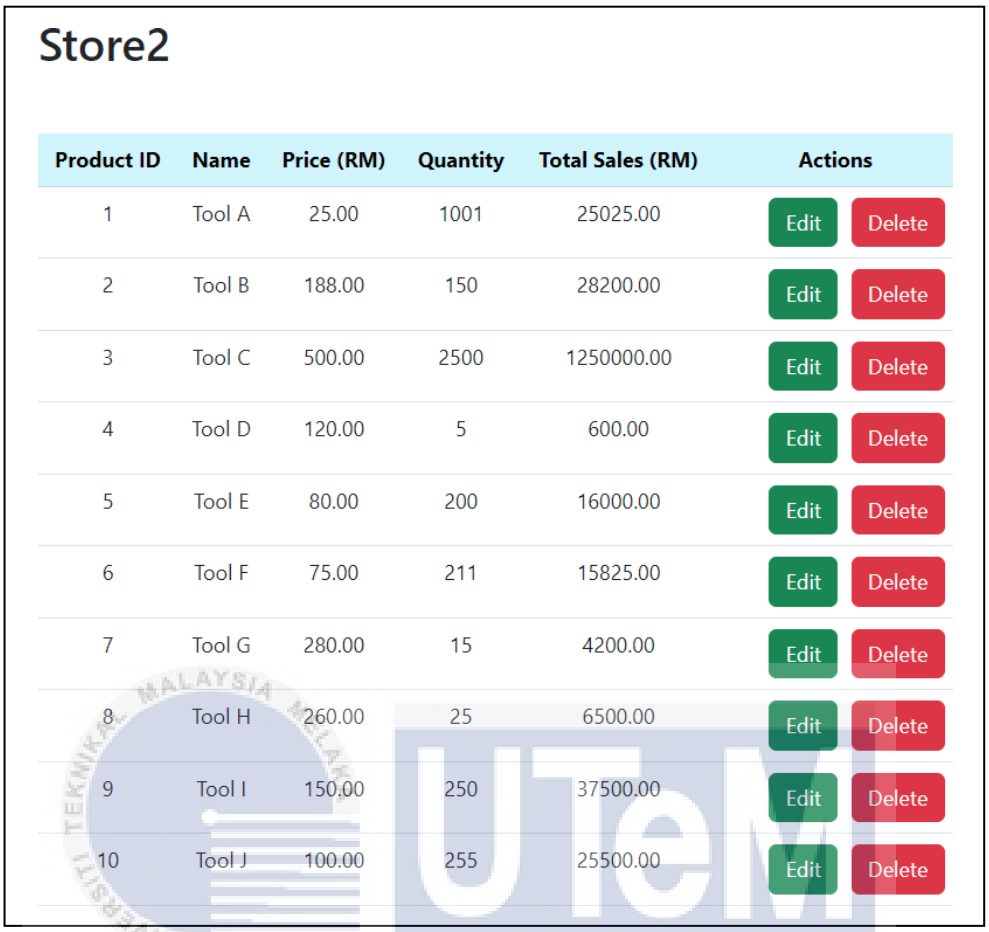
Product ID	Name	Price (RM)	Quantity	Actions
1	Product 1	120.00	1046	Edit Delete
2	Product 2	215.00	223	Edit Delete
3	Product 3	1112.00	500	Edit Delete
4	Product 4	111.00	21	Edit Delete
5	Product 5	200.00	500	Edit Delete
6	Product 6	688.00	1755	Edit Delete
7	Product 7	90.00	145	Edit Delete
8	Product 8	1000.00	266	Edit Delete
9	Product 9	187.00	33	Edit Delete
10	Product 10	350.00	22	Edit Delete

Figure 4.12 Table of Store 1 from webpage



id	name	price	quantity
1	Product 1	120.00	523
2	Product 2	215.00	223
3	Product 3	1112.00	500
4	Product 4	111.00	21
5	Product 5	200.00	500
6	Product 6	688.00	1755
7	Product 7	90.00	145
8	Product 8	1000.00	266
9	Product 9	187.00	33
10	Product 10	350.00	22
NULL	NULL	NULL	NULL

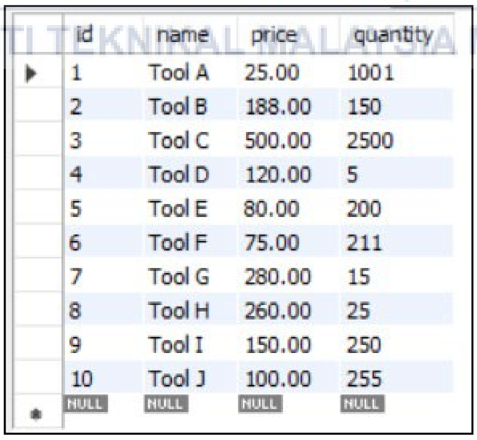
Figure 4.13 Table of Store 1 from MySQL Workbench



Store2

Product ID	Name	Price (RM)	Quantity	Total Sales (RM)	Actions
1	Tool A	25.00	1001	25025.00	Edit Delete
2	Tool B	188.00	150	28200.00	Edit Delete
3	Tool C	500.00	2500	1250000.00	Edit Delete
4	Tool D	120.00	5	600.00	Edit Delete
5	Tool E	80.00	200	16000.00	Edit Delete
6	Tool F	75.00	211	15825.00	Edit Delete
7	Tool G	280.00	15	4200.00	Edit Delete
8	Tool H	260.00	25	6500.00	Edit Delete
9	Tool I	150.00	250	37500.00	Edit Delete
10	Tool J	100.00	255	25500.00	Edit Delete

Figure 4.14 Table of Store 2 from webpage



id	name	price	quantity
1	Tool A	25.00	1001
2	Tool B	188.00	150
3	Tool C	500.00	2500
4	Tool D	120.00	5
5	Tool E	80.00	200
6	Tool F	75.00	211
7	Tool G	280.00	15
8	Tool H	260.00	25
9	Tool I	150.00	250
10	Tool J	100.00	255
NULL	NULL	NULL	NULL

Figure 4.15 Table of Store 2 from MySQL Workbench

A comparison has been made to compare the SQL table from MySQL workbench and the table in the webpage. To test whether both tables provide the desired results, insertion of data through MySQL workbench has been carried out and observed in the webpage. After that process, the insertion of data through the webpage is carried out and observed in the workbench to verify whether the desired output is obtained. All operations of CRUD were carried out and both webpage display and MySQL workbench are observed.

4.6 Testing on multiple devices

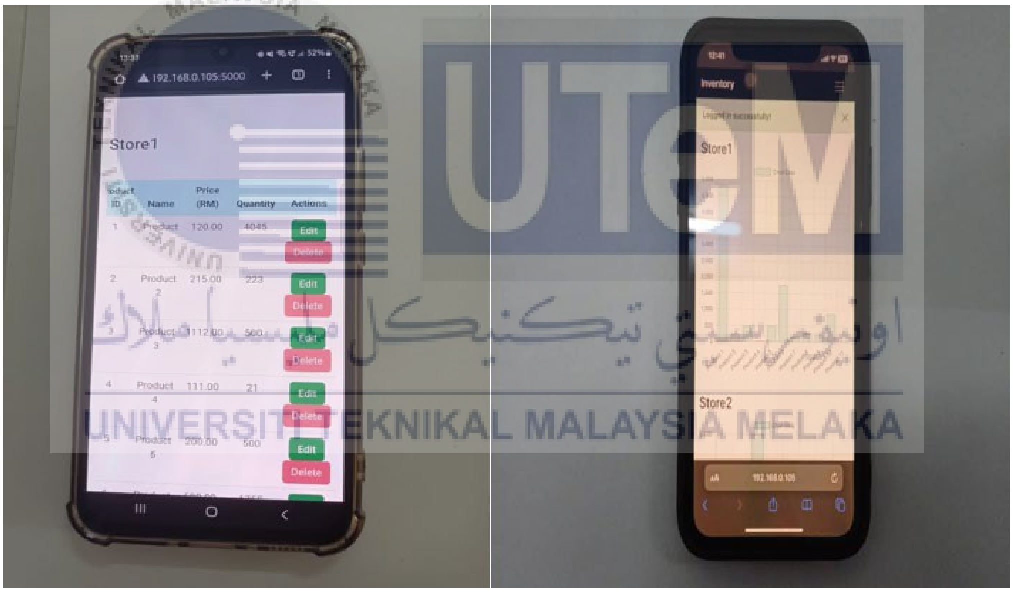
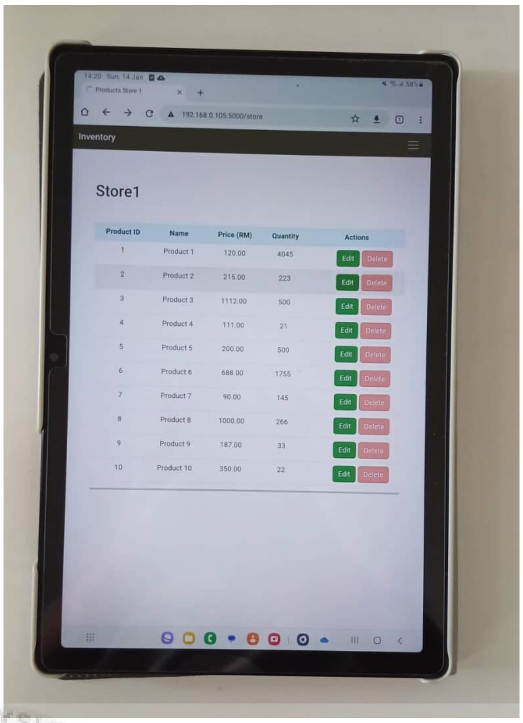


Figure 4.16 Using the application through Android and IOS mobile phones



Store1

Product ID	Name	Price (RM)	Quantity	Actions
1	Product 1	120.00	4045	Edit Delete
2	Product 2	215.00	223	Edit Delete
3	Product 3	1112.00	500	Edit Delete
4	Product 4	111.00	21	Edit Delete
5	Product 5	200.00	500	Edit Delete
6	Product 6	688.00	1755	Edit Delete
7	Product 7	90.00	145	Edit Delete
8	Product 8	1000.00	266	Edit Delete
9	Product 9	187.00	33	Edit Delete
10	Product 10	350.00	22	Edit Delete

Figure 4.17 Using the application through tablet

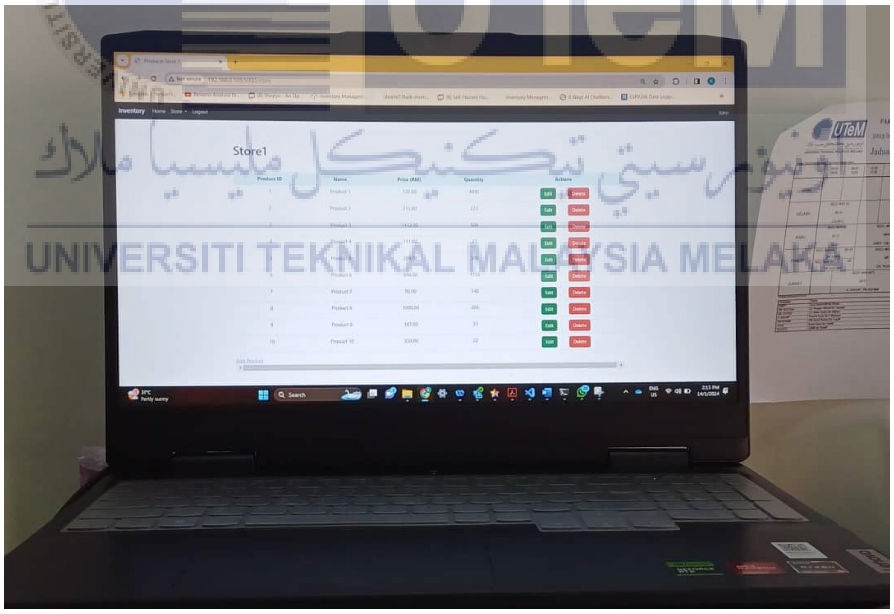
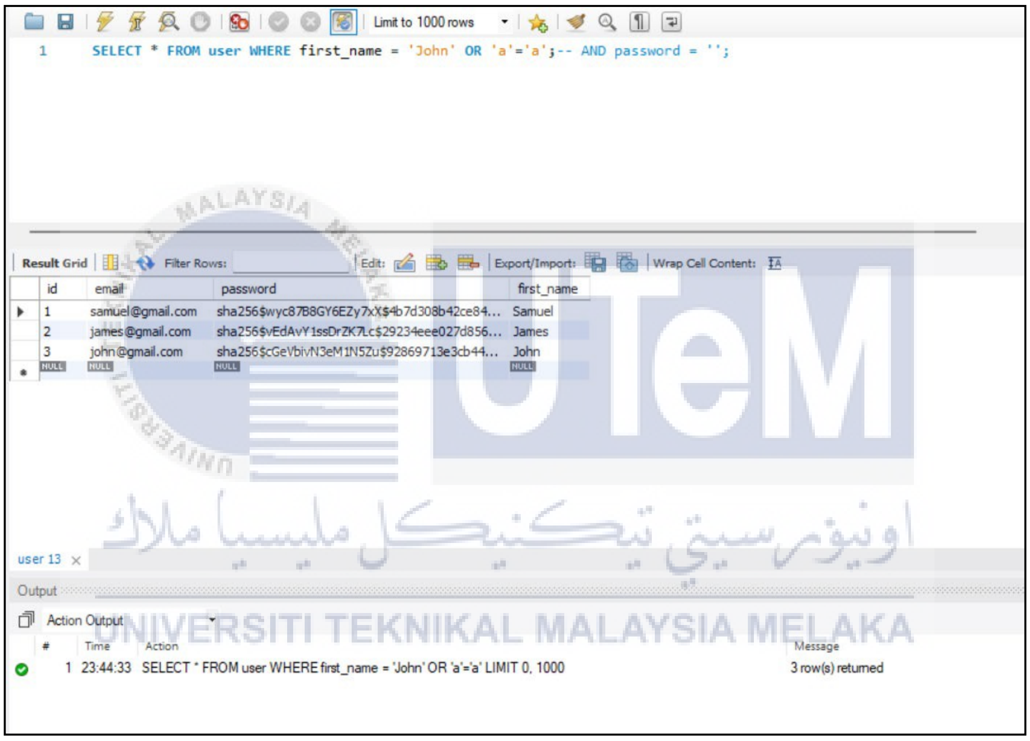


Figure 4.18 Using the application through laptop

This configuration causes the web server to bind to the given IP address local network when you run your Flask application. This indicates that devices linked to the same Wi-Fi network can access the Flask app. On the local network, every device—your laptop, phone, and tablet—has a distinct IP address.

4.7 Defense against SQL Injection



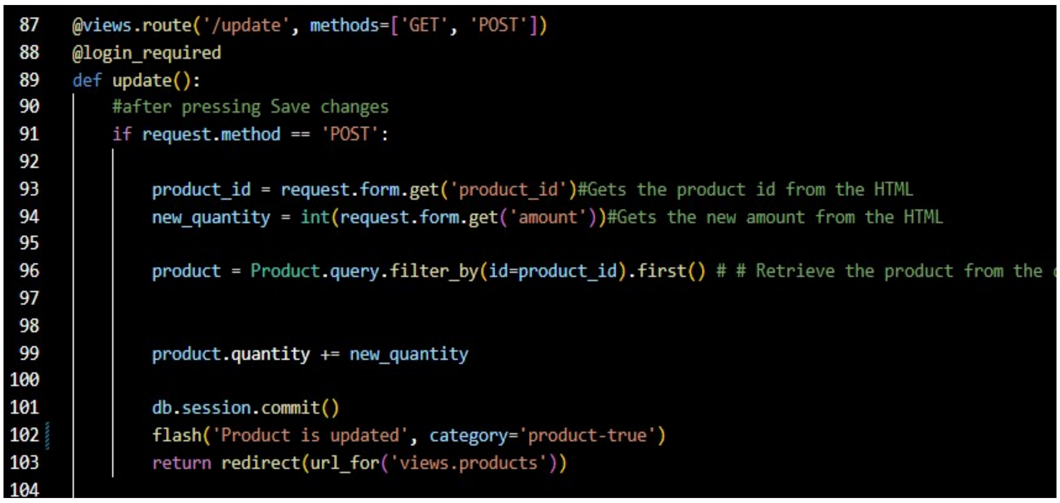
```
SELECT * FROM user WHERE first_name = 'John' OR 'a'='a';-- AND password = '';
```

id	email	password	first_name
1	samuel@gmail.com	sha256$wyc87B8GY6EZy7xX$4b7d308b42ce84...	Samuel
2	james@gmail.com	sha256$vEdAvY1ssDrZK7Lc$29234eee027d856...	James
3	john@gmail.com	sha256$cGeVbivN3eM1N5Zu$92869713e3cb44...	John
NULL	NULL	NULL	NULL

```
#   Time      Action                                                            Message
1   23:44:33  SELECT * FROM user WHERE first_name = 'John' OR 'a'='a' LIMIT 0, 1000   3 row(s) returned
```

Figure 4.19 SQL Injection Syntax

Above is an example of an SQL query SELECT * FROM user WHERE first_name = 'John' OR 'a'='a';-- AND password = ''; that is vulnerable to an SQL injection. When ‘a’ = ‘a’ it is a tautology, it means any row with a non-empty string will be equal to itself. This allows all rows to be returned. The double hyphen ‘--’ means that anything after the hyphen is commented on. This means the password is not taken into consideration.



```
87   @views.route('/update', methods=['GET', 'POST'])
88   @login_required
89   def update():
90       #after pressing Save changes
91       if request.method == 'POST':
92
93           product_id = request.form.get('product_id')#Gets the product id from the HTML
94           new_quantity = int(request.form.get('amount'))#Gets the new amount from the HTML
95
96           product = Product.query.filter_by(id=product_id).first() # # Retrieve the product from the
97
98
99           product.quantity += new_quantity
100
101          db.session.commit()
102          flash('Product is updated', category='product-true')
103          return redirect(url_for('views.products'))
104
```

Figure 4.20 Block of code executing update function

The code in figure 4.20 in line 95 uses SQL Alchemy technique which helps to sanitize input from user. It will parameterize the query using filter_by method. This will enable the product_id to be represented by a parameter rather than directly inserting to raw SQL string. These methods help to separate raw SQL from user input in the webpage to prevent SQL injection.

4.8 Limitations

An issue encountered during the project would be real time notifications when updating in laptop can't be observed from a user in the phone. A communication protocol called WebSocket allows for full-duplex communication channels over a single, persistent connection. In contrast to the request-response model of traditional HTTP, WebSocket enables real-time bidirectional communication between the server and clients. Applications that need to receive notifications and updates instantly will find this especially helpful. WebSocket will be best to be incorporated into this project to improve real-time communication between the server and linked clients. This is particularly helpful in situations where users need to be informed immediately, like when there are updates to a product, data changes, or other urgent information.

4.9 Summary

This chapter presented development results to demonstrate applicability of the proposed system using flask backend framework. Initial designs and tests are important to ensure the system's expected outcome will be produced as expected. It also allows the developer to detect any initial problems of the project. This chapter has shown ways the system was tested initially so that the first stages are executed without errors and the next steps can be implemented. For a web application, it is important to be used in a multi-platform environment.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This thesis presents a method for developing a web application-based inventory system. The proposed methodology is effective and robust in order to obtain good result by using a web-based app known as flask. Through this final year project, we have researched the tools needed to develop an inventory system that will be able to help them manage stock items in their warehouse. The scope and features of the web application will be defined with the assistance of this research. Then, a database schema will be created, considering elements like product specifics and stock levels, to ensure proper data storage and retrieval. Based on the objectives, formulating an accurate database is achieved due to obtaining the desired result after every operation. Moreover, the proposed objective to develop a notification message between management and staff user interface and dashboard for businesses to manage their inventory levels was not fully achieved due to not using a more suitable protocol such as WebSocket or APIs such as Firebase. In addition, the final proposed objective would be to verify whether the developed web application able to run effectively and smoothly for tablet, laptop and mobile devices and that is achieved because the functionality that was intended for the system does what it should at all devices.

5.2 Potential for Commercialization

Web application Inventory management systems have significant potential for commercialization, particularly as demand for database accuracy and stock management efficiency continue to grow. There are a range of applications for inventory management systems that use databases in various sectors, such as food & beverage, retail, school management system, and so on. For example, a student management system that also uses inventory management concept can help educators to keep track of student grades, scores, and details. Retail sectors can keep track of their stock to improve accuracy and study sales trends by their stock count. However, it is important to balance the potential for commercialization with the need to ensure that inventory management is user friendly for diverse stakeholders, particularly in the context of addressing the global challenge of inventory management issues. By creating a user-friendly and intuitive inventory management system, this project can differentiate itself from competitors who offer outdated or complex solutions. The modern web interface, robust functionalities, and streamlined workflows provided by a Flask-based system can give the proposed project a competitive edge in the market.

5.3 Future Works

For future improvements, the functionality of web-based inventory management system could be added and enhanced as follows:

i) Remove data input redundancy such as product id input.

ii) Display prompts when user input wrong data or malicious data.

iii) Building a machine learning model to help business to improve based on the dataset and that was obtained from the database tables.

iv) Creating a machine learning model to help business to improve based on the dataset and that was obtained from the database tables.

v) Include an admin dashboard to track item numbers that are sold every day and sales trends.

vi) Integrating real-time database with backend framework flask for better accuracy.

vii)Implementing AI chatbot to help can assist users in querying product information, checking inventory levels, and facilitating seamless communication.

REFERENCES

[1] “Inventory Management System,” 2022. [Online]. Available: www.ijcrt.org

[2] L. Akande Salahudeen and A. O. A, “Effect of Inventory Management System on Operational Performance in Manufacturing Firms: Study of May and Baker Manufacturing *Industry* Nig Ltd, Lagos,” 2018.

[3] Serbia and Montenegro IEEE Section. CAS-SP Chapter, Serbia and Montenegro IEEE Section, Univerzitet u Beogradu. School of Electrical Engineering., Univerzitet u Beogradu. Innovation Center., Telecommunications Society (Serbia), and Institute of Electrical and Electronics Engineers, NEUREL 2018 : 2018 14th Symposium on Neural Networks and Applications (NEUREL) : November 20-21, 2018, SAVA Center, Milentija Popovića 9, Belgrade, Serbia.

[4] M. Chila and L. C. Susi, “Implementing a Web-Based Inventory Tracking System: A Quality Improvement Initiative,” Journal of Radiology Nursing, vol. 38, no. 4. Elsevier Inc., pp. 277–280, Dec. 01, 2019. doi: 10.1016/j.jradnu.2019.09.009.

[5] D. R. A. Shirley, R. B. Amruthavarshni, A. Durainathan, and M. P. Karthika, “QR-Based inventory management system (QR-IMS) of passenger luggage using website,” in Proceedings - 5th International Conference on Intelligent Computing and Control Systems, ICICCS 2021, Institute of Electrical and Electronics Engineers Inc., May 2021, pp. 1180–1185. doi: 10.1109/ICICCS51141.2021.9432384.

[6] I. H. Q. D. Mohd Mahzan and K. L. Lee, “Elimination of Misconduct in Manual Counting Process as an Improvement of Inventory Accuracy in A Manufacturing Company,” International Journal of Industrial Management, vol. 10, pp. 140–150, Mar. 2021, doi: 10.15282/ijim.10.1.2021.6051.

[7] K. A. Shafie and M. Zabri, ‘Inventory management practices among Malaysian micro retailing enterprises’, ABRM, 2016. [Online]. Available: www.jbrmr.com

[8] V. G. S and A. S. Shivaleela, ‘This work is licensed under a Creative Commons Attribution 4.0 International License A Review of Inventory Management System’, IJARCCE International Journal of Advanced Research in Computer and Communication Engineering, vol. 10, 2021, doi: 10.17148/IJARCCE.2021.10689.

[9] Challenges in Inventory Management and a Proposed Framework. [Online]. Available: www.gecekitapligi.com

[10] S. Wynn and J. R. Kuhn, ‘The Financial Impact of Manual Inventory Record Errors’, International Journal of Business and Social Science, vol. 12, no. 10, 2021, doi: 10.30845/ijbss.v12n10p2.

[11] A. Susanto, ‘Database Management System’, INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH, vol. 8, no. 06, 2019, [Online]. Available: www.ijstr.org

[12] O. Mamuyovwi Helen, ‘THE ROLE OF DATABASE MANAGEMENT SYSTEM (DBMS) IN INSTITUTION/ORGANIZATION’, 2021. [Online]. Available: https://www.bmc.com/blogs/dbms-database-management-systems/

[13] S. Murugesan, ‘Chapter 2 WEB APPLICATION DEVELOPMENT: CHALLENGES AND THE ROLE’.

[14] F. Ahmad, F. Baharom, and M. Husni, ‘Current Web Application Development and Measurement Practices for Small Software Firms’.

[15] S. Banerjee, S. Seth, T. Dey, and D. Pal, ‘PYTHON PROGRAMMING LANGUAGE AND ITS SCOPE IN FUTURE’. [Online]. Available: www.irjmets.com

[16] N. Idris, C. Feresa, M. Foozy, and P. Shamala, “A Generic Review of Web Technology: DJango and Flask,” 2020.

[17] Arshad, S., & Sajjad Khan, M. (n.d.). Computerized Inventory System-Program Development and Execution. In INTERNATIONAL JOURNAL OF AGRICULTURE & BIOLOGY.

[18] Yuvaraj, K., Oorappan, G. M., Megavarthini, K. K., Pravin, M. C., Adharsh, R., & Ashwath Kumaran, M. (2020). Design and Development of An Application for Database Maintenance in Inventory Management System Using Tkinter and Sqlite Platform. IOP Conference Series: Materials Science and Engineering, 995(1). https://doi.org/10.1088/1757-899X/995/1/012012

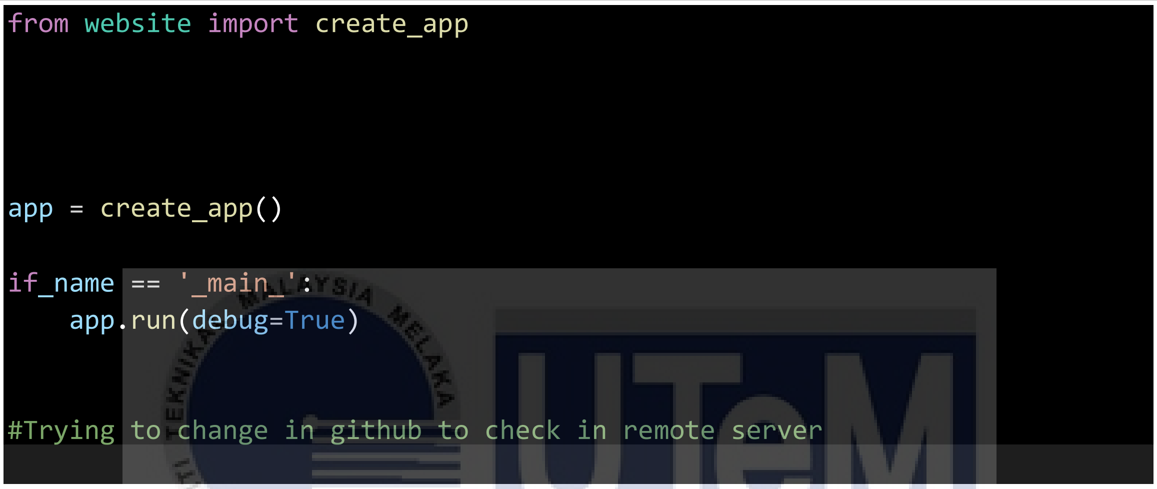
[19] Wynn, S., & Kuhn, J. R. (2021). The Financial Impact of Manual Inventory Record Errors. International Journal of Business and Social Science, 12(10). https://doi.org/10.30845/ijbss.v12n10p2

[20] Abdullah, R., Zi Xiang, K., & Ilman Hakimi Chua Abdullah, M. (2018). E-Inventory management system using android mobile application at Faculty of Engineering Technology laboratory stores. https://www.statista.com/statistics/494587/smartph

APPENDICES

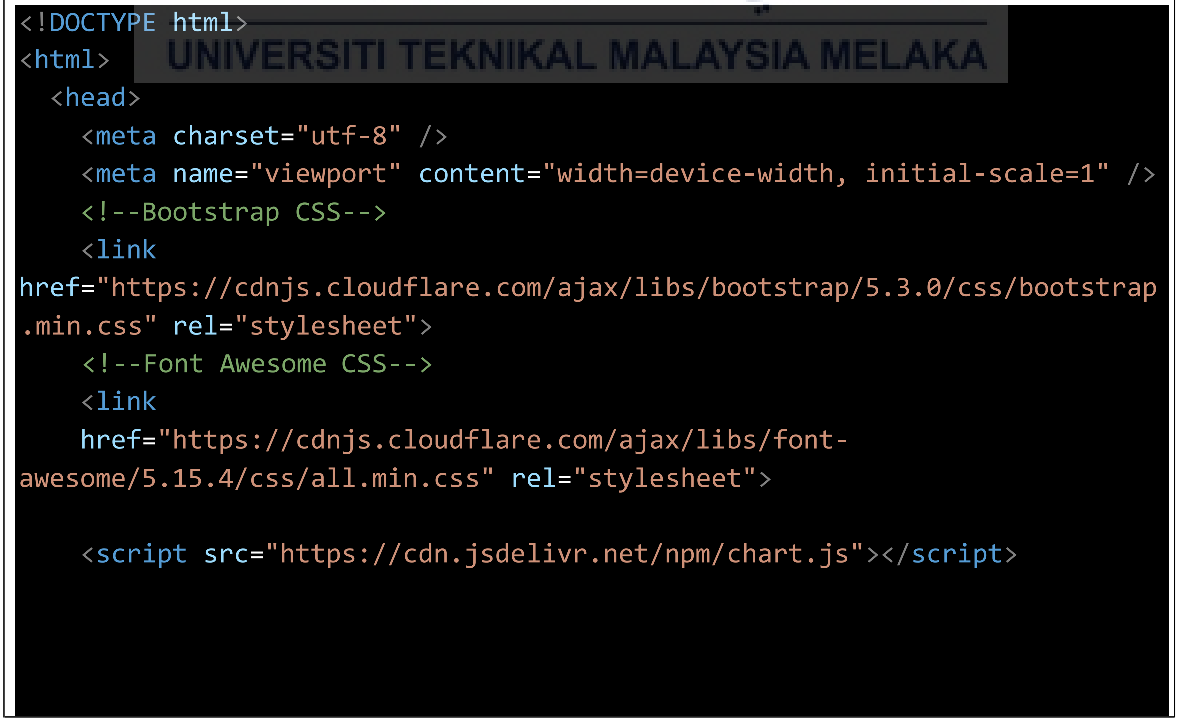
Appendix A Code For Web App

main.py



```python
from website import create_app



app = create_app()

if_name == '_main_':
    app.run(debug=True)


#Trying to change in github to check in remote server
```

Base.html



```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <!--Bootstrap CSS-->
    <link
href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.0/css/bootstrap
.min.css" rel="stylesheet">
    <!--Font Awesome CSS-->
    <link
    href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css" rel="stylesheet">

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <title>{% block title %}Home{% endblock %}</title>
  </head>
  <body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">Inventory</a>
        <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
          {% if user.is_authenticated %}
          <ul class="navbar-nav me-auto mb-2 mb-lg-0">
            <li class="nav-item">
              <a class="nav-link" aria-current="page" href="/">Home</a>
            </li>
            <li class="nav-item dropdown">
              <a class="nav-link dropdown-toggle" href="#"
id="navbarDropdownStore" role="button" data-bs-toggle="dropdown" aria-
expanded="false">
                  Store
              </a>
              {%if user.id == 3 %}
              <ul class="dropdown-menu" aria-
labelledby="navbarDropdownStore">
                  <li><a class="dropdown-item" href="/store">Store
1</a></li>
                  <!-- Add more dropdown items as needed -->

                  <li><a class="dropdown-item" href="/store2">Store
2</a></li>
                  <!-- Example of another dropdown item -->


              {%elif user.id == 1 %}
              <ul class="dropdown-menu" aria-
labelledby="navbarDropdownStore">
                  <li><a class="dropdown-item" href="/store">Store
1</a></li>
                  <!-- Add more dropdown items as needed -->



              {%elif user.id == 2 %}
              <ul class="dropdown-menu" aria-
labelledby="navbarDropdownStore">
```

```
                  <li><a class="dropdown-item" href="/store2">Store
2</a></li>
                  <!-- Add more dropdown items as needed -->

              {%endif%}


              </ul>
          </li>
            <li class="nav-item">
              <a class="nav-link" href="/logout">Logout</a>
            </li>

          </ul>
          <form class="d-flex">
            <span class="navbar-text me-2">{{user.first_name}}</span>
          </form>



          {% else %}
          <ul class="navbar-nav me-auto mb-2 mb-lg-0">
            <li class="nav-item">
              <a class="nav-link" href="/sign-up">Sign Up</a>
            </li>


            <li class="nav-item">
              <a class="nav-link" href="/login">Login</a>
            </li>




        {%endif%}
        </div>
      </div>
    </nav>
    {% with messages = get_flashed_messages(with_categories=true) %} {% if
    messages %} {% for category, message in messages %} {% if category ==
    'error' %}
    <div class="alert alert-danger alert-dismissible fade show"
role="alert">
      <div class="d-flex justify-content-between">
```

```
        <div>{{ message }}</div>
        <button type="button" class="btn-close" data-bs-dismiss="alert"
aria-label="Close"></button>
      </div>
    </div>
    {% else %}
    <div class="alert alert-success alert-dismissible fade show"
role="alert">
      <div class="d-flex justify-content-between">
        <div>{{ message }}</div>
        <button type="button" class="btn-close" data-bs-dismiss="alert"
aria-label="Close"></button>
      </div>
    </div>
    {% endif %} {% endfor %} {% endif %} {% endwith %}


    {% with message2s = get_flashed_messages(with_categories=true) %} {%
if
      message2s %} {% for category, message2 in message2s %} {% if
category ==
      'product-true' %}
      <div class="alert alert-success alert-dismissible fade show"
role="alert">
        <div class="d-flex justify-content-between">
          <div>{{ message }}</div>
          <button type="button" class="btn-close" data-bs-dismiss="alert"
aria-label="Close"></button>
        </div>
      </div>
      {% endif %} {% endfor %} {% endif %} {% endwith %}




    <div class="container">{% block content %} {% endblock %}</div>




    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min
.js"
    integrity="sha384-
IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"
    crossorigin="anonymous">
  </script>
```

```
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js
"
    integrity="sha384-
cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF"
    crossorigin="anonymous">
    </script>



  </body>
</html>


<!-- Bootstrap 5


<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Inventory</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      {% if user.is_authenticated %}
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="/">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/Store">Store</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/logout">Logout</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Dropdown
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><hr class="dropdown-divider"></li>
      <li><a class="dropdown-item" href="#">Something else
here</a></li>
     </ul>
    </li>
    <li class="nav-item">
     <a class="nav-link disabled" href="#" tabindex="-1" aria-
disabled="true">Disabled</a>
    </li>
   </ul>
   <form class="d-flex">
    <span class="navbar-text me-2">{{user.first_name}}</span>
   </form>

   {% else %}
   <li class="nav-item">
     <a class="nav-link" href="/login">Login</a>
    </li>

    <li class="nav-item">
     <a class="nav-link" href="/logout">Logout</a>
    </li>
    {% endif %}
  </div>
 </div>
</nav>
```

products.html

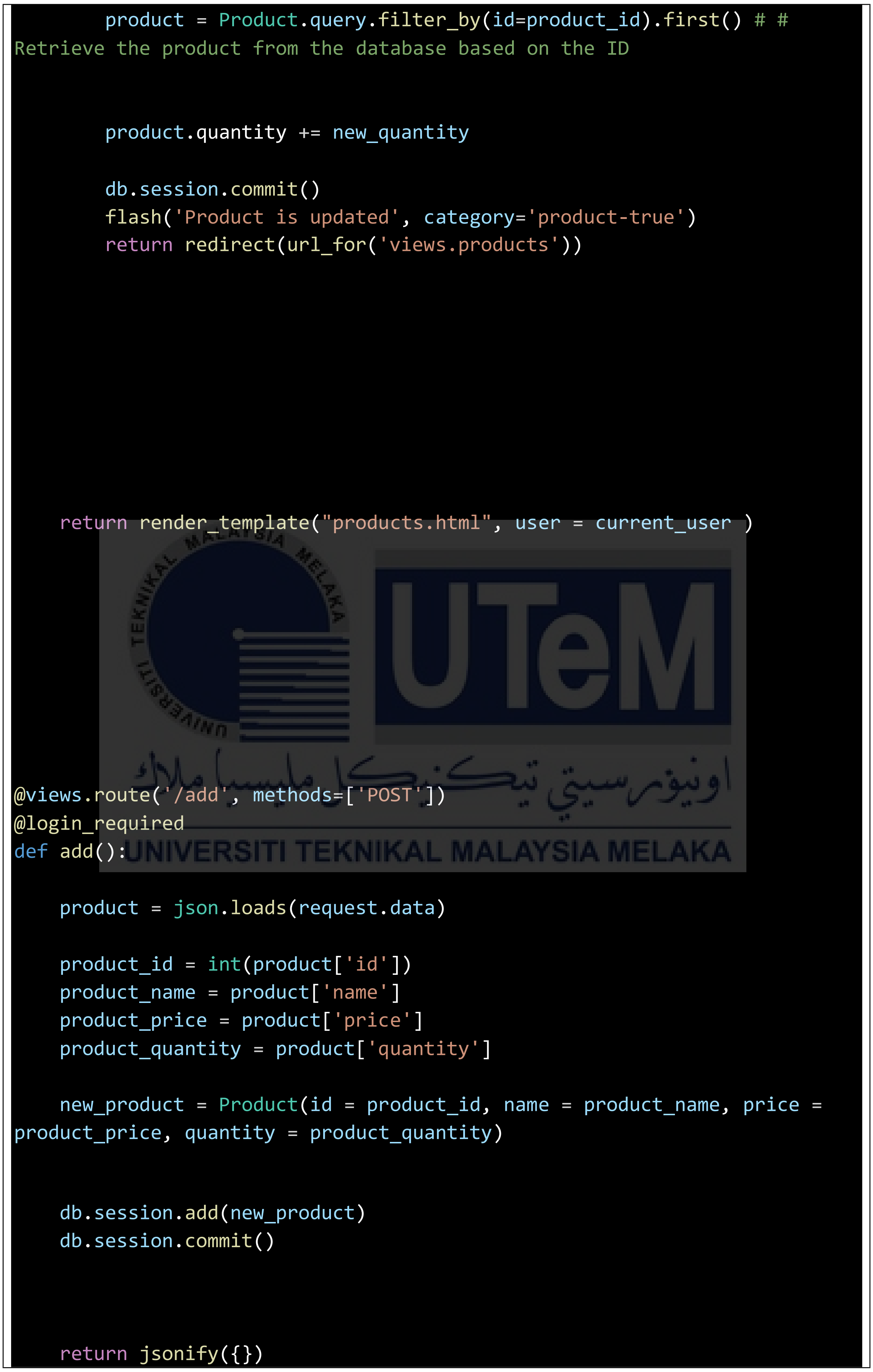
```
{% extends "base.html" %} {% block title %}Products Store 1{% endblock %}
{% block
  content %}


  <br>
  <br>
  <h1 class="mt-4 mb-4"> {{ tablename1 }}</h1>
  <br>
  <div class="table-responsive">
    <table class="table table-hover text-center">
        <thead class="table-info">
          <tr>
            <th>Product ID</th>
            <th>Name</th>
            <th>Price (RM)</th>
            <th>Quantity</th>
           <!--<th>Total Sales (RM)</th>-->
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {% for product in products %}
              <tr>
                  <td>{{ product.id if product.id is not none else "N/A"
}}</td>
                  <td>{{ product.name }}</td>
                  <td>{{ product.price }}</td>
                  <td>{{ product.quantity }}</td>
                 <!-- <td>{{ product.quantity * product.price}}</td>-->
                  <td>
                    <div class="row justify-content-center">
                      <div class="col-md-3 text-center">
                        <form method="post" action="/edit_products">
                          <input type="hidden" name="product_id"
value="{{ product.id }}">
                          <button type="submit" class="btn btn-
success">Edit</button>
                        </form>
                      </div>

                    {%if user.id == 3%}
                      <div class="col-md-3 text-center">
                        <form method="post" action="/delete">
                          <input type="hidden" name="product_id"
value="{{ product.id }}" id ="product_id">
```

```
                              <button type="submit" class = "btn btn-
danger">Delete



                            </button>
                          </form>
                        </div>
                          {%elif user.id == 1%}
                            <div class="col-md-3 text-center">
                              <form method="post" action="/delete">
                                <input type="hidden" name="product_id"
value="{{ product.id }}" id ="product_id">

                                  <button type="submit"disabled class =
"btn btn-danger">Delete
                                 </button>
                              </form>
                          </div>
                          {%endif%}
                      </div>
                    </td>
                </tr>
        {% endfor %}
        </tbody>
      </table>


    {%if user.id == 3%}
      <a href =""   name="id-target" data-bs-toggle="modal" data-bs-
target="#myModal">Add Product</a>
    {%endif%}
    <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
        <div class="modal-dialog" role="document">
          <div class="modal-content">
            <div class="modal-header">
              <h5 class="modal-title" id="exampleModalLabel">Add
Product</h5>
              <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close">

              </button>
            </div>
            <div class="modal-body">
              <form method="post" id="form-content">
                <div class="form-group">
```

```
                  <label for="product-id" class="col-form-label">New
Product Id:</label>
                  <textarea name="id-content" class="form-control" id="id-
content"></textarea>

                  <label for="product-name" class="col-form-label">New
Product name:</label>
                  <textarea name="name-content" class="form-control"
id="name-content"></textarea>


                  <label for="product-price" class="col-form-label">New
Product Price:</label>
                  <textarea name="price-content" class="form-control"
id="price-content"></textarea>


                  <label for="product-quantity" class="col-form-label">New
Product Quantity:</label>
                  <textarea name="quantity-content" class="form-control"
id="quantity-content"></textarea>
             </div>
            </div>
            <div class="modal-footer">
              <button type="submit" onClick="addProduct()" class="btn btn-
primary" name="submit" value="content">Confirm</button>
            </div>
          </div>
        </div>
      </div>
    </form>


<script>

function addProduct() {
  var id =  document.getElementById("id-content").value;
  var name = document.getElementById("name-content").value;
  var price = document.getElementById("price-content").value;
  var quantity = document.getElementById("quantity-content").value;


  fetch("/add", {
    method: "POST",
    body: JSON.stringify({ id: id, name: name , price : price, quantity :
quantity }),
  }).then((_res) => {
    window.location.href = "/store";
  });
}

function deleteProduct(){
  var id = document.getElementById("product_id").value

  fetch("/delete", {
    method: "POST",
    body: JSON.stringify({ id: id }),
  }).then((_res) => {
    window.location.href = "/store";
  });

}

</script>

{% endblock %}
```

Views.py

```python
from flask import Blueprint, redirect, render_template, request, flash, 
jsonify, url_for
from flask_login import login_required, current_user
from .models import Product,Product2,User

from . import db
import json


views = Blueprint('views',_name__)



@views.route('/', methods=['GET', 'POST','DELETE'])
@login_required
def home():

    all_users = User.query.all()

    tablename1 = Product._tablename
    tablename2 = Product2.__tablename__



    return render_template("home.html", 
user=current_user,all_users=all_users , tablename1 = tablename1 , 
tablename2 = tablename2)

@views.route('/get_chart_data')
def get_chart_data():


    data = Product.query.all()

    # Processing data
    labels = [row.name for row in data]
    values = [row.quantity for row in data]

    # Prepare data to send as JSON
    chart_data = {'labels': labels, 'values': values}
    return jsonify(chart_data)



@views.route('/store', methods=['GET', 'POST'])
@login_required
def products():
    products = Product.query.all()      #MySQL Alchemy method
    tablename1 = Product.__tablename__

    return render_template("products.html", user=current_user, products =
products , tablename1 = tablename1 )




'''The route where current product information is filled'''

@views.route('/edit_products', methods=['GET', 'POST'])
@login_required
def edit_products():

    if request.method == 'POST':
        id = request.form.get('product_id')#Gets the product id from the
HTML

        product = Product.query.filter_by(id=id).first()






    return render_template("edit.html", product = product , user =
current_user)


@views.route('/update', methods=['GET', 'POST'])
@login_required
def update():
    #after pressing Save changes
    if request.method == 'POST':

        product_id = request.form.get('product_id')#Gets the product id
from the HTML
        new_quantity = int(request.form.get('amount'))#Gets the new amount
from the HTML
```



```
        product = Product.query.filter_by(id=product_id).first() # #
Retrieve the product from the database based on the ID


        product.quantity += new_quantity

        db.session.commit()
        flash('Product is updated', category='product-true')
        return redirect(url_for('views.products'))









    return render_template("products.html", user = current_user )









@views.route('/add', methods=['POST'])
@login_required
def add():

    product = json.loads(request.data)

    product_id = int(product['id'])
    product_name = product['name']
    product_price = product['price']
    product_quantity = product['quantity']

    new_product = Product(id = product_id, name = product_name, price =
product_price, quantity = product_quantity)


    db.session.add(new_product)
    db.session.commit()



    return jsonify({})
```

```
@views.route('/delete', methods=['POST'])
@login_required
def delete():

    product_id = request.form.get('product_id')#Gets the product id from
the HTML

    product = Product.query.filter_by(id=product_id).first() # # Retrieve
the product from the database based on the ID



    db.session.delete(product)
    db.session.commit()

    return redirect(url_for('views.products'))
'''

user = User.query.filter_by(email=email).first()
        if user:
            flash('Email already exists.', category='error')
        elif len(email) < 4:
            flash('Email must be greater than 3 characters.',
category='error')
        elif len(first_name) < 2:
            flash('First name must be greater than 1 character.',
category='error')
        elif password1 != password2:
            flash('Passwords don\'t match.', category='error')
        elif len(password1) < 7:
            flash('Password must be at least 7 characters.',
category='error')

'''
```