Inventory management system for retailers

(Report by – Aparna)

## *Abstract-* The Purpose of this project is to highlight the versatility and functionality of ﬂask along with MySQL by developing an inventory 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.

* 1. **Overview**

1. **Introduction**

Inventory, A structured, categorized 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 record like products, quantities 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 MySQL database. The front end has been developed using bootstrap and connected to the back-end using ﬂask. The system will be login enabled. The managers will be able to add, update, delete, retrieve it and display it on the front-end.

## Purpose

The Aim of this project is to showcase the versatility & pragmatics of utilizing Flask along with Remote MySQL to deliver a robust inventory management 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.

# Literature Survey

## Existing Problem

Businesses are quickly realising that inventory control is absolutely necessary to run an eﬃcient business and make money in the process. Especially in today’s competitive marketplace, business owners simply can’t aﬀord to have money go down the drain.

* + - High cost of inventory
    - Consistent stock outs
    - Low rate of inventory turnover
    - High amount of obsolete inventory
    - High amount of working capital
    - High cost of storage
    - Spreadsheet data-entry errors
    - Lost customers

## Proposed Solution

For this system we shall be using the following technologies:

1. *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

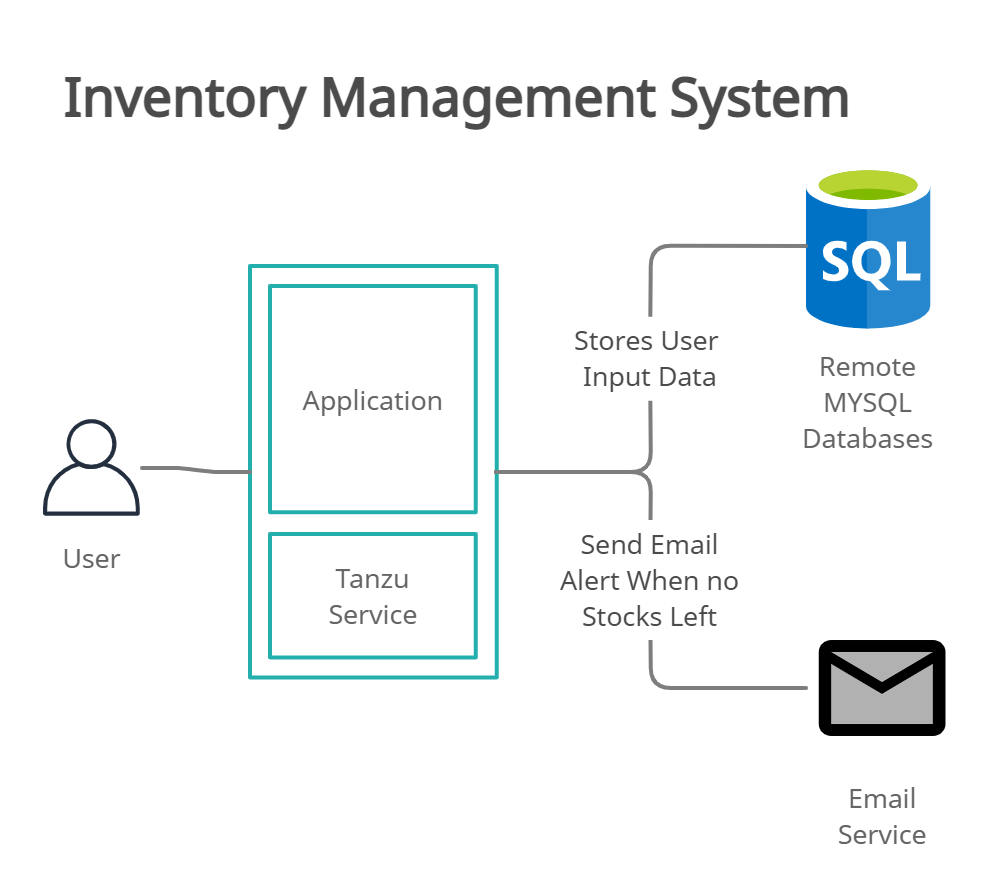
provided for database integration, form validation, upload handling, various open authentication technologies, and more. Flask may be “micro”, but it’s ready for production use on a variety of needs.

1. *Remote MYSQL*

Overview. This feature allows **remote** hosts (servers) to access **MySQL**® databases on your account. This is useful, for example, if you wish to allow shopping cart or guestbook applications on other servers to access your databases.

## Block Diagram

1. **THEORITICAL ANALYSIS**



* 1. **Hardware / Software Designing**

Hardware Requirements:

* + - Minimum 1GB Ram
    - Minimum Intel i3 Core Processor
    - Minimum 10GB Hard disk space

Software Requirements:

* + - Operating System: Windows 10 or later
    - Browser: Chrome latest version
    - Internet Connection: 1Mbps

# Experimental Investigation

Investigation made while working on the solution:

Modules included in webapp:

**Registration**: The user can register with their details in order to login in the webapp.

**Login**: The users can Login in the webapp using their details.

**Add Product**: User/Retailer can Add their products by entering Product name, Description and QTY.

**Report**: User can Export the list of products and QTY by clicking Export report button.

**Stock**: User can view their available products in the stock.

**Location**: User can add their location.

**Product Movement**: User can manage their product movements

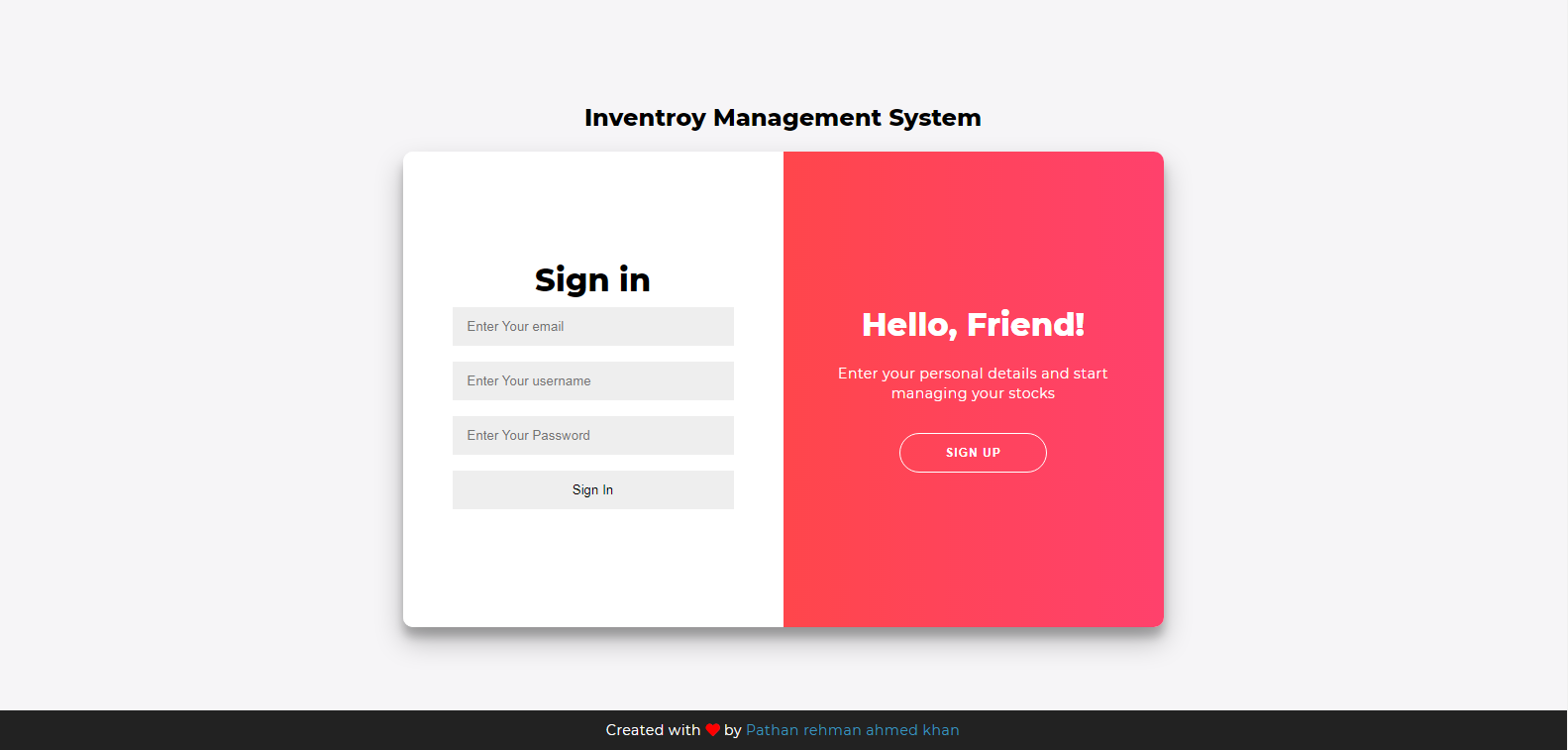
**Logout:** The user can logout from the webapp

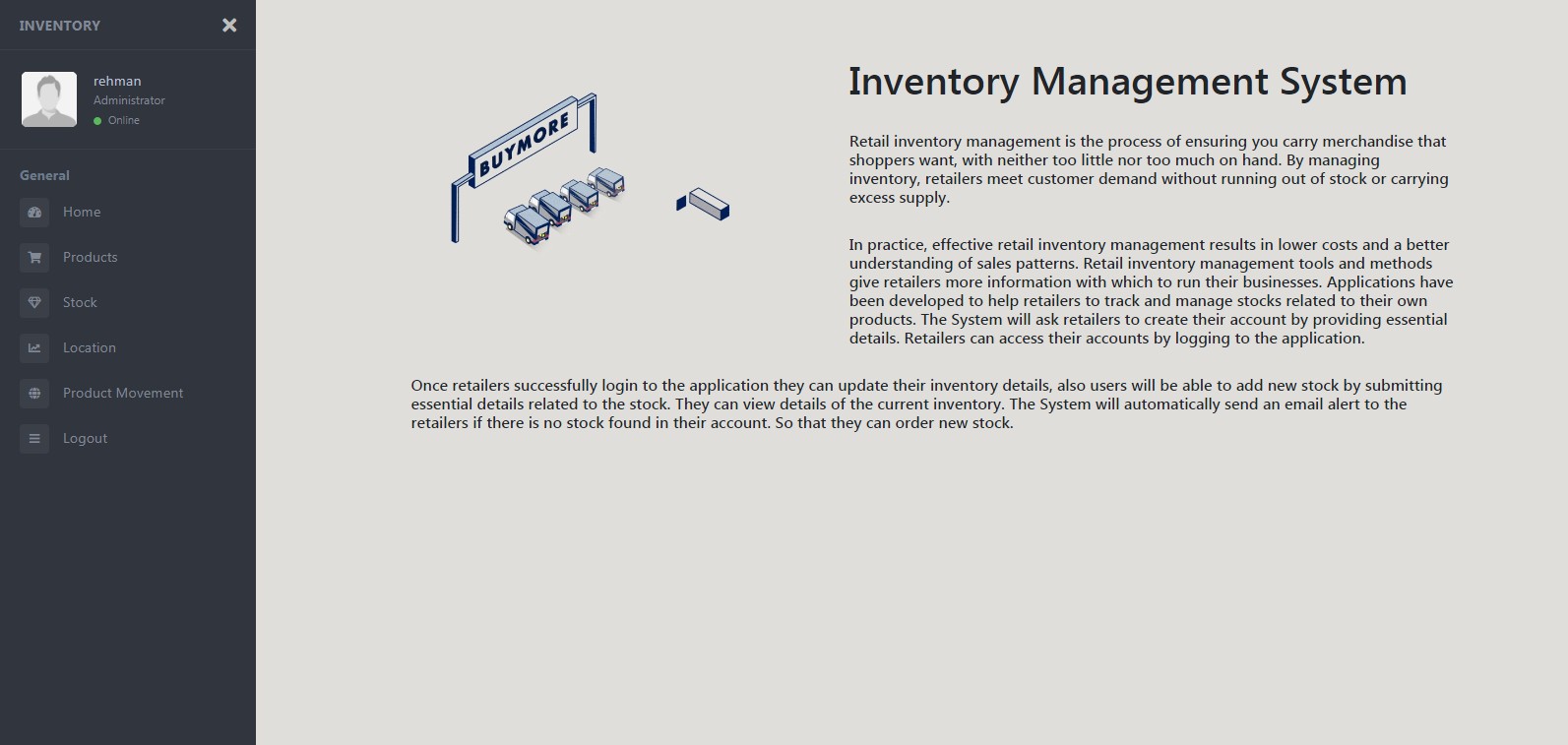
# Flowchart

1. **Result**

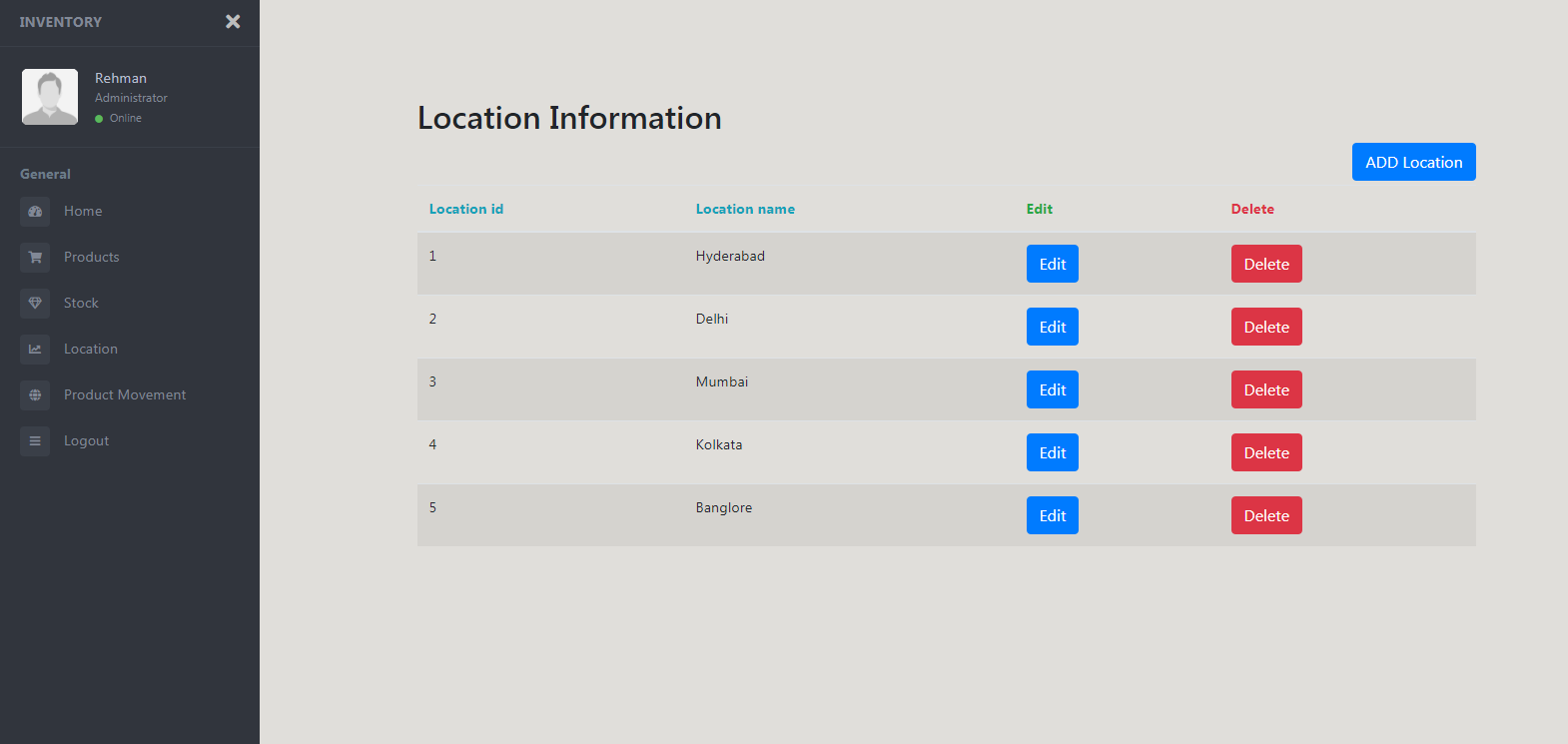
## Sign-Up Page

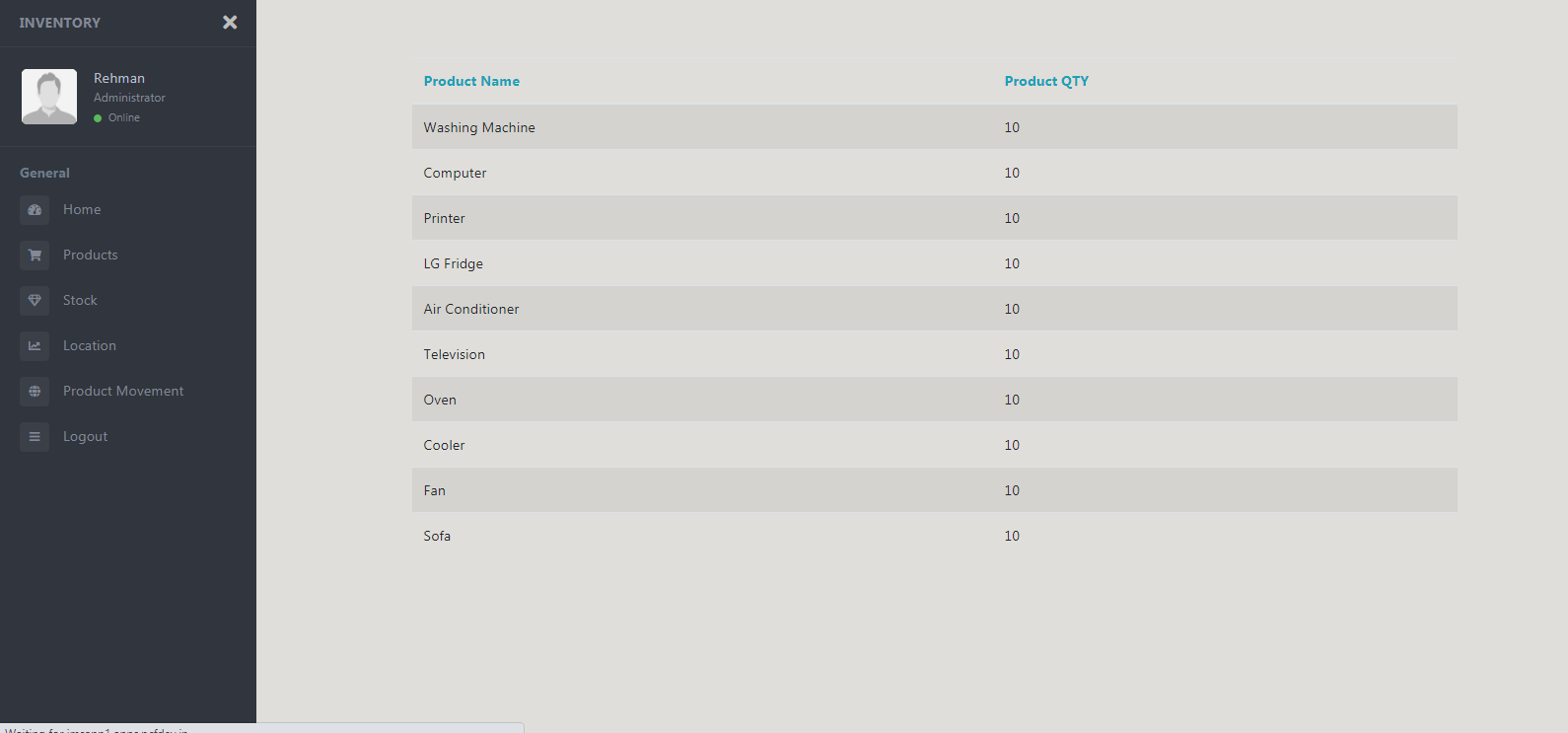
**Sign-In Page**



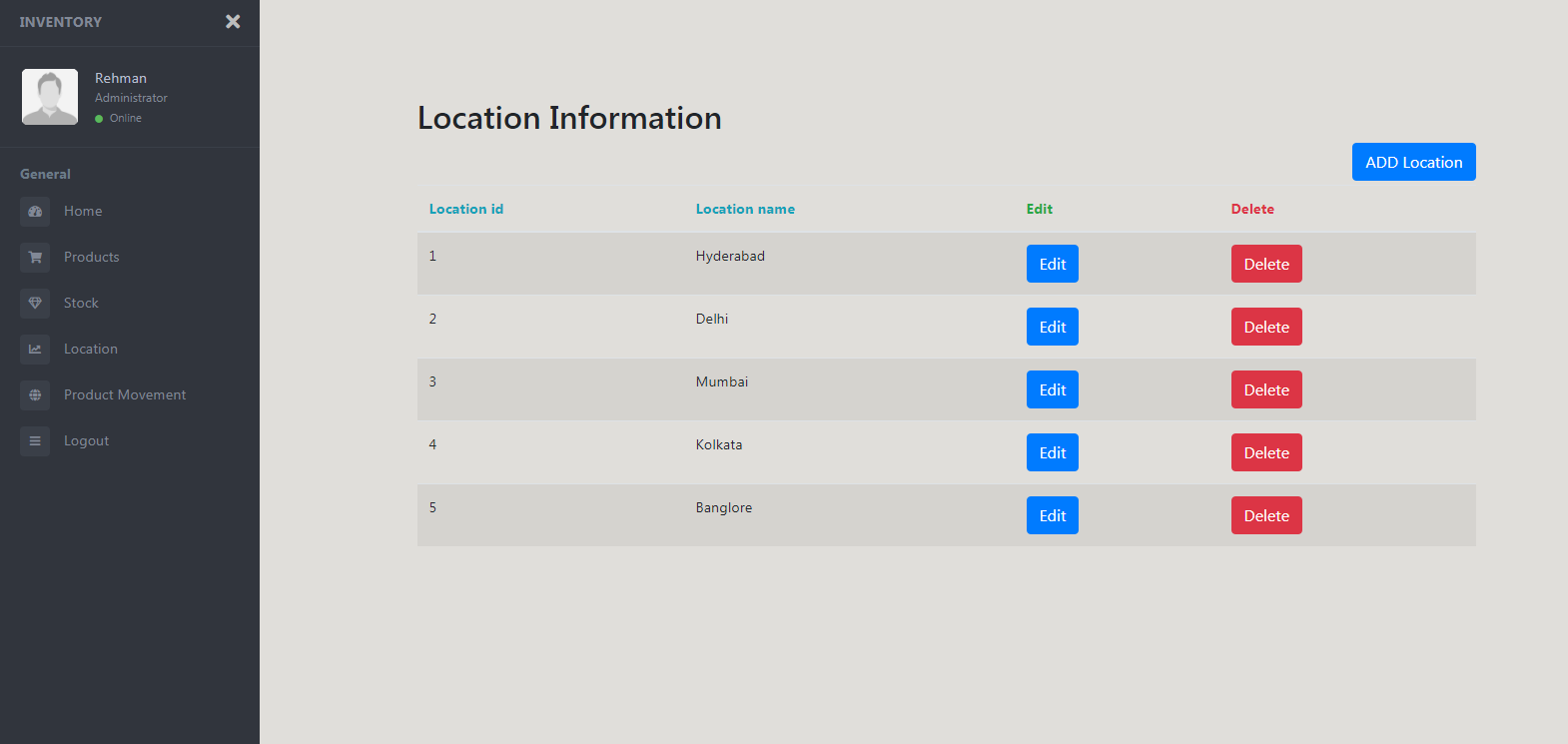
**Homepage**

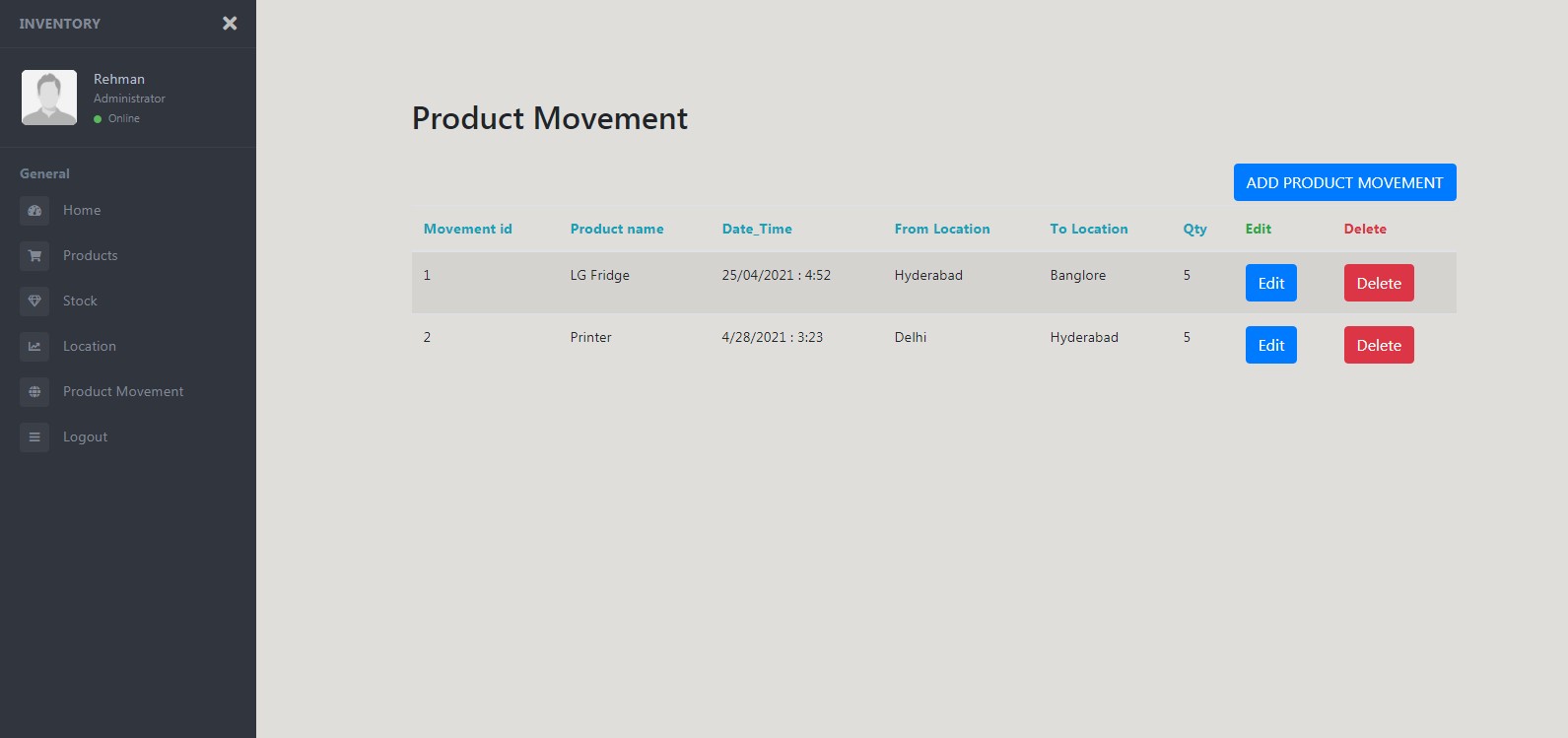
**Products Page**



**Stocks Page**

**Location Page**



**Product Movement Page**

1. **Advantages and Disadvantages**

**Advantages:**

1. Better Inventory Accuracy.
2. Reduced risk of overselling.
3. Cost Savings.
4. Avoiding stockouts and Excess stock.
5. More Productivity.
6. Increased Proﬁts.
7. Better Customer Experience.

## Disadvantages:

1. Expensive for small businesses.
2. Risk of system crashes.
3. Malicious Hacks.
4. Works for only single company.

## 8.Applications

1. Wholesale.
2. Distributors.
3. Ecommerce.
4. Retailers.
5. Manufaturing companies.
6. Warehouse

## Conclusion

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.

## Future Scope

Future work can be done to add a sales page and payments page to manage stocks and product sells simultaneously. The customers can buy products from sales page with secured payment gateway in order to avoid fradulents and scams.

## Bibilography

1. **[Flask.*ﬂask.* ]** [**https://ﬂask.palletsprojects.com/en/1.1.x/foreword/**](https://flask.palletsprojects.com/en/1.1.x/foreword/)
2. **[Flask install]**

[**h ttps://www.twilio.com/docs/usage/tutorials/how-to-set-up-your-python-and-ﬂask-development-envi**](https://www.twilio.com/docs/usage/tutorials/how-to-set-up-your-python-and-flask-development-environment)

[**ronment**](https://www.twilio.com/docs/usage/tutorials/how-to-set-up-your-python-and-flask-development-environment)

1. **[My SQL database creation]** [**h ttps://remotemysql.com/**](https://remotemysql.com/)
2. **[Creating a Flask Project]**

[**h ttps://code.tutsplus.com/tutorials/creating-a-web-app-from-scratch-using-python-ﬂask-and-mysql--cms-22972**](https://code.tutsplus.com/tutorials/creating-a-web-app-from-scratch-using-python-flask-and-mysql--cms-22972)

1. **[Creating REST API]**

[**h ttps://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-ﬂask**](https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask)

1. **[Designing the UI] h** [**ttps://spring.io/guides/tutorials/react-and-spring-data-rest/**](https://spring.io/guides/tutorials/react-and-spring-data-rest/)
2. **[Flask and MYSQL Database Connection]**

[**h ttps://www.askpython.com/python-modules/ﬂask/ﬂask-mysql-database**](https://www.askpython.com/python-modules/flask/flask-mysql-database)

1. **[Docker Image creation]** [**h ttps://www.geeksforgeeks.org/dockerize-your-ﬂask-app/**](https://www.geeksforgeeks.org/dockerize-your-flask-app/)
2. **[Depolying APP to TAS]**

[**h ttps://drive.google.com/ﬁle/d/1Y7V9XBm1iCsC37Fjv2UO6fnyGPZJcE6r/view?usp=sharing**](https://drive.google.com/file/d/1Y7V9XBm1iCsC37Fjv2UO6fnyGPZJcE6r/view?usp=sharing)

1. **[installing cli foundary] h** [**ttps://docs.cloudfoundry.org/cf-cli/install-go-cli.html**](https://docs.cloudfoundry.org/cf-cli/install-go-cli.html)

**APPENDIX:**

**Source code:**

[**h ttps://github.com/smartinternz02/SPS-9714-Inventory-Managment-System-for-Retailers/tree/main/fINAL\_**](https://github.com/smartinternz02/SPS-9714-Inventory-Managment-System-for-Retailers/tree/main/fINAL_IMS_PROJECT)

[**IMS\_PROJECT**](https://github.com/smartinternz02/SPS-9714-Inventory-Managment-System-for-Retailers/tree/main/fINAL_IMS_PROJECT)