PRACTICE 2: PickPollo
Date: 2nd of June of 2025

Professor: Cesar Eduardo Inda Ceniceros

Students:

- Hugo Misael Caro Jacobo
- Jessie Alain Peña Ramirez
- Jesus Eduardo Romero Isiordia
- Angel Michel Vela Flores
- Elvis Vergara Hernandez

Team: La Gestapo

INTRODUCTION

Nowadays, automating processes in small businesses—such as a poultry shop—represents a great opportunity to optimize time, improve customer service, and reduce human errors. This project aims to develop a comprehensive software system to efficiently manage the main operations of a poultry shop, including customer registration, sales control, product management, and report generation.

The system will be designed with a modern architecture and implemented through a user-friendly, accessible, and functional interface. By using technologies such as a REST API and an interactive graphical interface, we will ensure a smooth experience for both staff and customers.

REQUIREMENTS

Defined Roles

- **Who is the CEO:**
- **Who is the Software Architect:**
- **Who is the SD.FrontEnd:**
- **Who are the SD.BackEnd:**
- **Project Milestones (3 Months of Development – 12 Weeks):** Simulating 3 months of development (4 weeks × 3 months), 12 milestones (checkpoints) are defined.

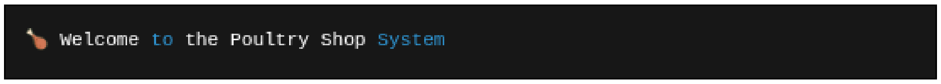
DEVELOPMENT

(This section is included in the MD document)

<u>RESULTS</u>

1. Welcome Message

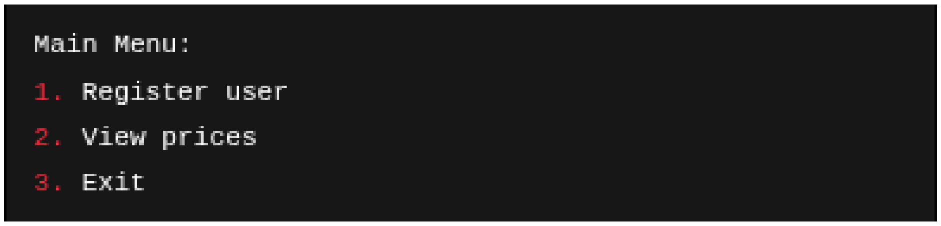
When the system starts, it shows:



```
🍗 Welcome to the Poultry Shop System
```

2. Main Menu

It displays three options:



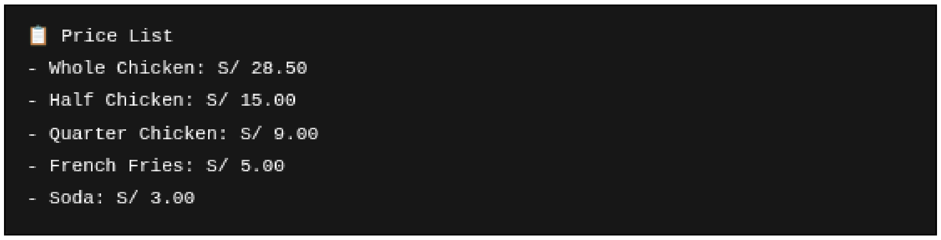
```
Main Menu:
1. Register user
2. View prices
3. Exit
```

3. User Registration

- Prompts the user to enter their name.
- Validates that the name contains only letters.
- Prompts for a username (alphanumeric, no symbols).
- Checks if the username already exists or confirms successful registration.

4. Price List

Displays something like:



```
📋 Price List
- Whole Chicken: S/ 28.50
- Half Chicken: S/ 15.00
- Quarter Chicken: S/ 9.00
- French Fries: S/ 5.00
- Soda: S/ 3.00
```

ISSUES

In this practice we have some troubles in this part because.

1. Lack of Persistent Storage

- Current Issue: User data and product data are stored in memory only (dictionaries).
- Future Problem: Once the app is closed, all data is lost. No login system or user tracking is possible.
- Solution: Implement a database (e.g., SQLite or PostgreSQL) to store users, products, and orders persistently.

2. Scalability Limitations

- Current Issue: The code is simple and linear.
- Future Problem: As more features (orders, reports, inventory, authentication) are added, the structure will become hard to maintain.
- Solution: Refactor into modules (e.g., models, controllers, views) or use a framework like Flask or Django.

3. User Interface

- Current Issue: Terminal-based input/output is not user-friendly.
- Future Problem: Not suitable for staff with no technical background; lacks accessibility.
- Solution: Develop a graphical interface (GUI) or web-based UI.

4. Security and Validation

- Current Issue: No password system, and minimal validation.
- Future Problem: Any user can register freely, data could be easily corrupted or spoofed.
- Solution: Add password handling, hash storage, session management, and better input sanitization.

CONCLUSIONS

Through this practice, we learned several key programming concepts, especially in data handling and input validation in a user registration system.

Main Learnings:

- **Data Validation**
  - We used nombre.isalpha() to ensure the name only contained letters.
  - We applied regular expressions (re.match()) to validate that the username was alphanumeric without symbols.
- **Data Structure Management**
  - We used dictionaries (productos and usuarios_registrados) to store information in a structured way.
- **Flow Control and Loops**
  - We implemented an interactive menu with while True so the user could choose options until deciding to exit (break).
  - We used conditionals (if-elif-else) to handle the different menu options.
- **Clear User Messages**
  - We provided feedback with emojis and descriptive text (✅, ❌, ⚠️) to guide the user in case of errors or successful actions.
- **Modular Functions**
  - We separated the logic into functions (validar_nombre, validar_usuario, registrar_usuario, etc.) to keep the code organized and reusable.

GITHUB REPOSITORY

https://github.com/elvis-v7/ISOFT5_SA.git