**ORGANIC ORIGINS**

**(A RUST BASED INVENTORY MANAGEMENT SYSTEM)**

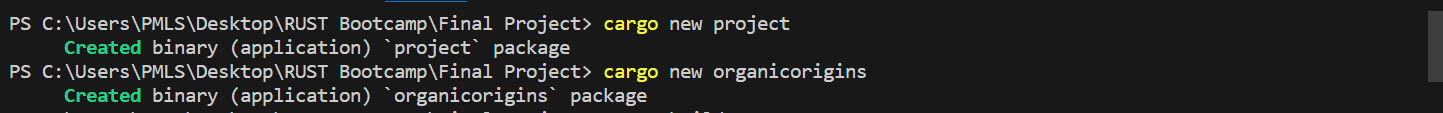
**BY**

**MUHAMMAD AZFAR IQBAL**

* Running our System Program

For our Rust Organic Origins (Inventory Management System), we need to have cargo installed. After that we create a new cargo project “organicorigins” using:

**cargo new organicorigins**

****

Then we go to the directory where our project is created using:

**cd organicorigins**

We write the code for our Organic Origins project in the main.rs file.

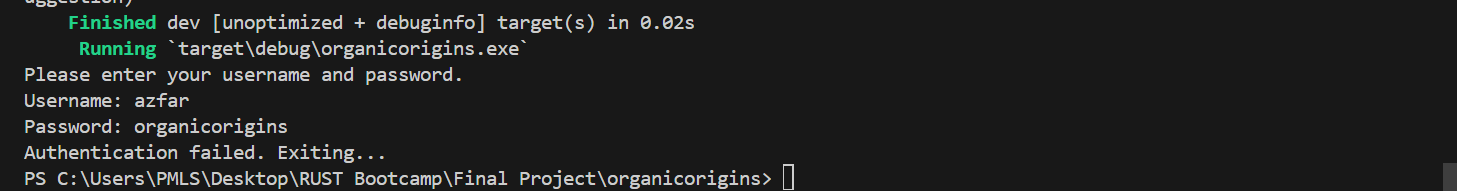
After finishing developing the project, we first compile it using:

**cargo build**

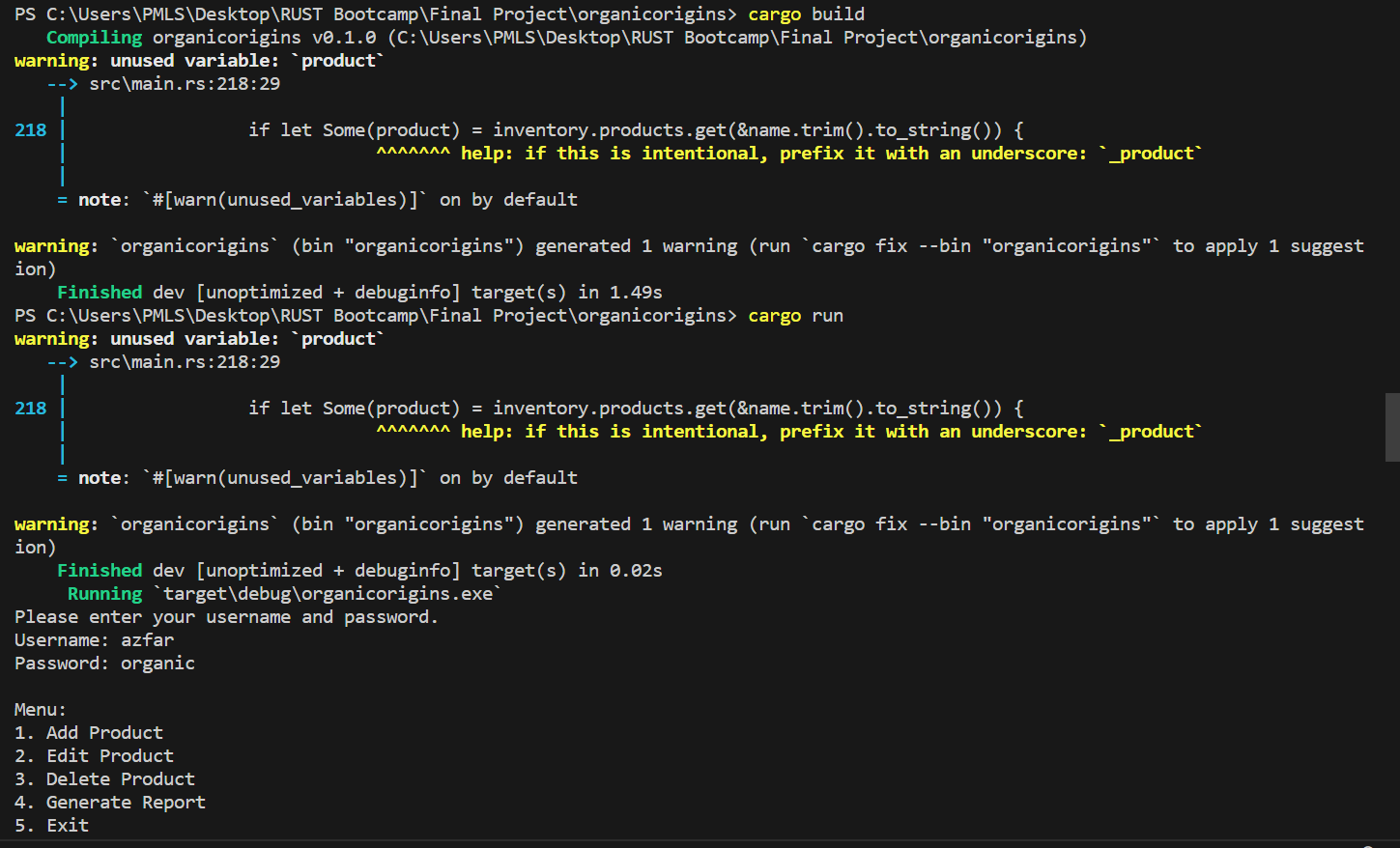
Then, we run it using:

**cargo run**

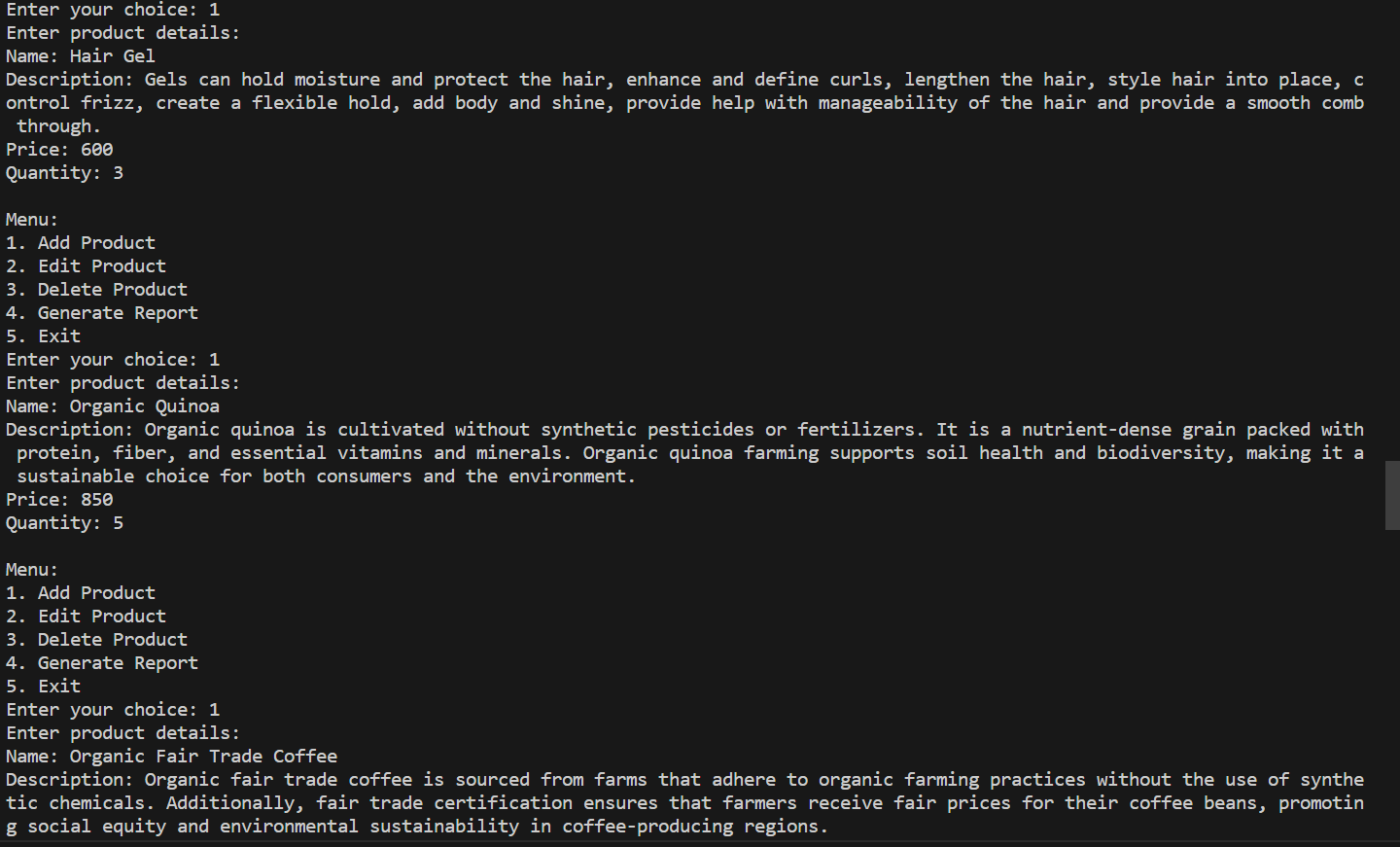
Failed authentication exits the user from the system:



The system prompts us for authentication. We enter the username and password as per login credentials then proceed. Our project menu is displayed.



* Add Product (1)



A computer screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

* Edit & Delete Product (2 & 3)

A computer screen with white text

Description automatically generated

* **Generate Report (4)**

A screenshot of a computer screen

Description automatically generated

* **Exit (5)**

A black screen with white text

Description automatically generated

This code includes the implementation of a well-structured table for generating the inventory report. It utilizes string formatting to align the columns properly, making the report more readable and organized. The remaining parts of the code need to be implemented for adding, editing, and deleting products, as well as authentication logic and file I/O operations.

Authentication: The authenticate function prompts the user for a username and password. It checks whether the entered credentials match a predefined username and password (in this case, "admin" and "password"). If authentication fails, the program exits.

Persistence: The program loads inventory data from a file named "inventory.txt" at startup and saves the inventory to the same file before exiting. If the file does not exist, the program creates a new inventory.

Expanded Error Handling: The program handles errors related to file operations, parsing data from the file, and invalid input during product editing.

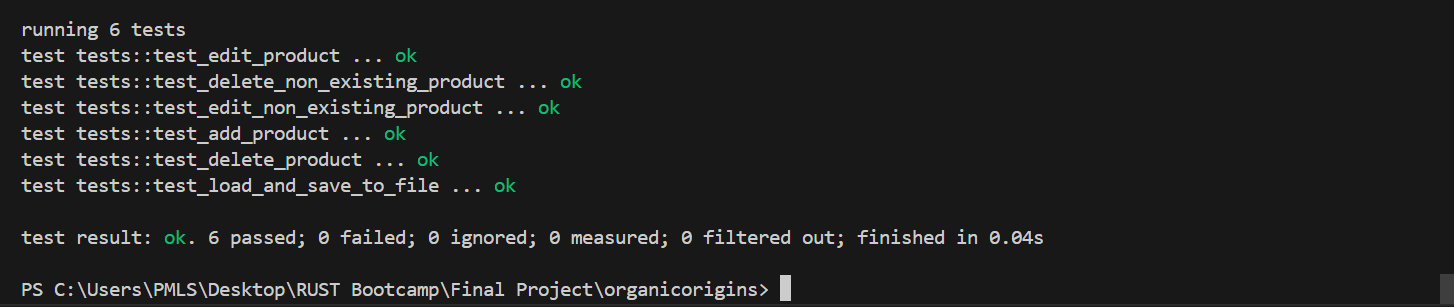
The project consists of major features such as Inventory Management, Reporting, Error Handling, Security, and intuitive text-based user interface.

We use Rust's basic data types, functions, and control flow structures, as well as concepts such as ownership, borrowing, structs, enums, traits, and error handling.

**Testing:**

We have developed unit test cases related to adding, editing, and deleting products. Moreover, editing and deleting non-existent products has also been tested. Loading and saving to files has also been tested successfully. After building and compiling, we use:

**cargo test**

****