**NAMES:** KEITH S. REYES  **DATE:** 12/18/2023

SHERWIN T. VALLEDOR **INSTRUCTOR: Arl Kheem Rey Gallario**

**COURSE/YEAR/BLOCK:** BSIT 2A

**INVENTORY SYSTEM**

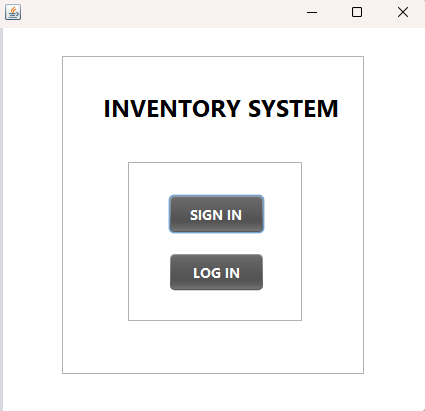
1. **Introduction**

The primary objective of the project is to design and implement a comprehensive CRUD application encompassing both user account management and product inventory control. The system involves several several jframe classesStartingTab, RegisterTab, LoginTab, and MenuTab. This application is intended to provide users with a seamless experience for registering, logging in, and efficiently managing product data through an intuitive and user-friendly graphical interface.

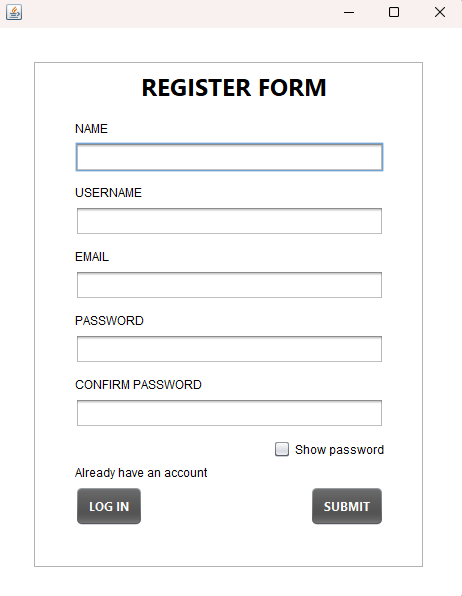
The significance of the CRUD application lies in its ability to streamline user interactions with the system, offering a centralized platform to perform essential operations on both user accounts and product records. This enhances the efficiency of data management and contributes to an organized and structured approach to user and inventory control.

1. **Project Structure**

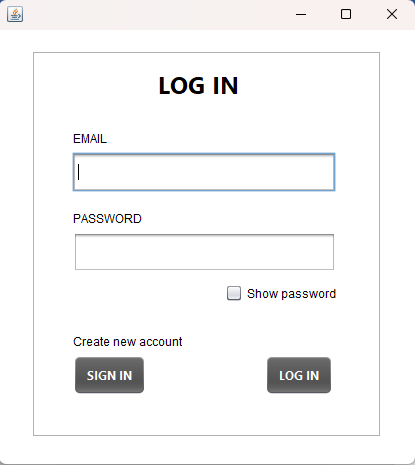
**StartingTab:** Serves as the entry point, presenting users with options to either register or log in. This class sets the initial tone for user interaction as you can see in the Figure 1.



**Figure 1: StartingTab**

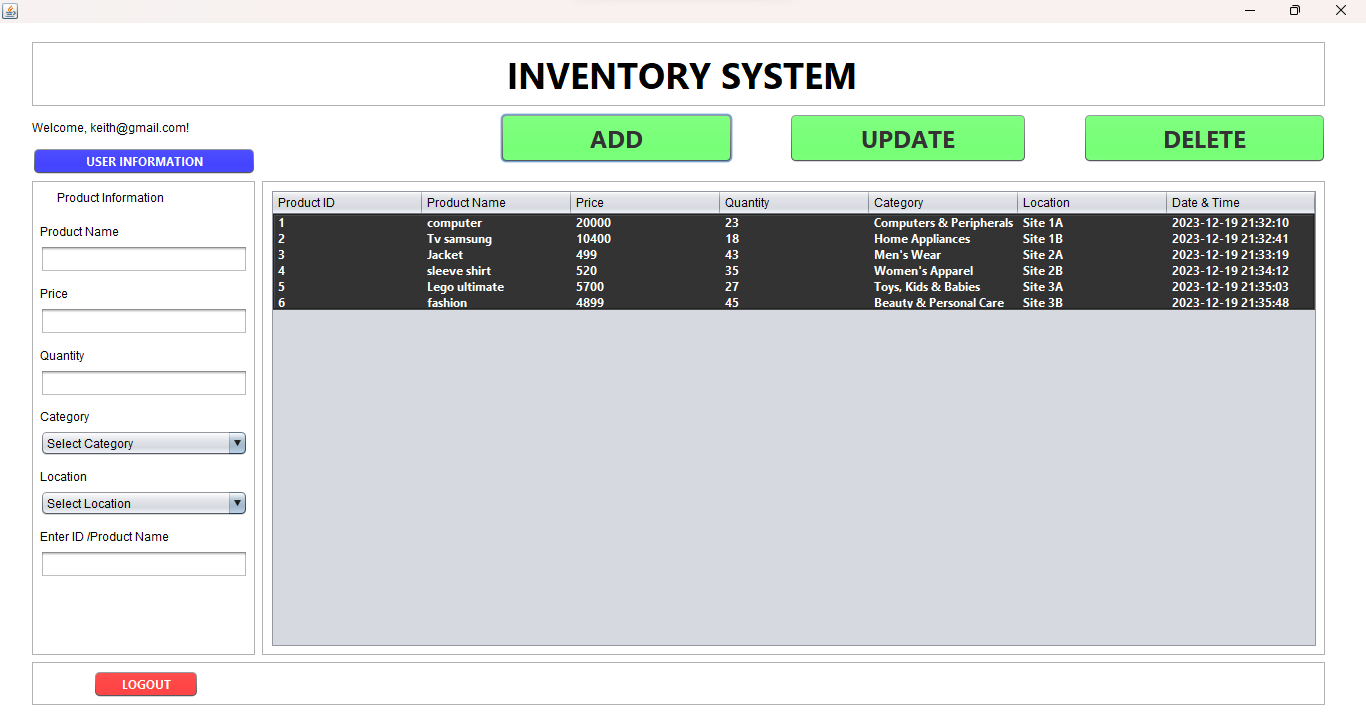
**RegisterTab:** Facilitates the user registration process, collecting essential information and persistently storing it. This class is responsible for maintaining user data as you can see in figure 2.

**Figure 2: RegisterTab:**

**LoginTab:** Manages the login functionality, ensuring secure authentication by validating user credentials against stored information as you can see in figure 3.

**Figure 3:** **LoginTab**

**MenuTab:** Forms the core of the application, providing a versatile interface for users to perform CRUD operations on product information. This class plays a pivotal role in controlling and presenting product data as you can see in figure 4.

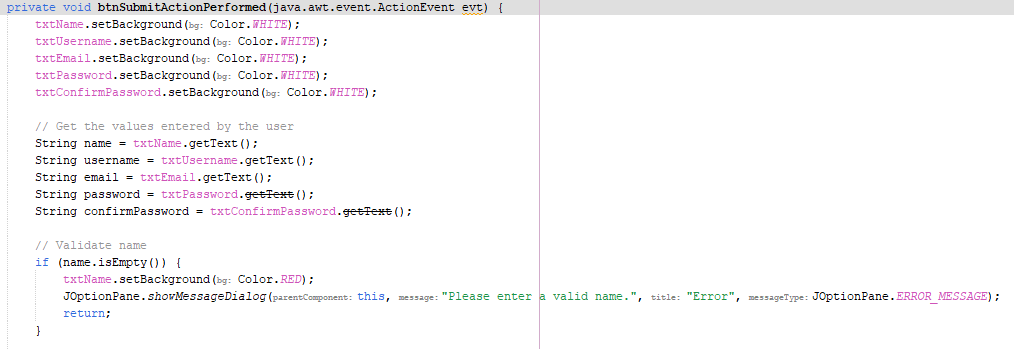


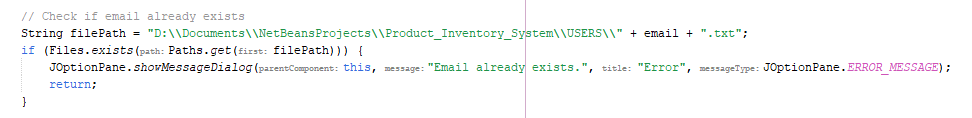
**Figure 4:** **MenuTab**

1. **Code Snippets**

**StartingTab:**

**RegisterTab:**

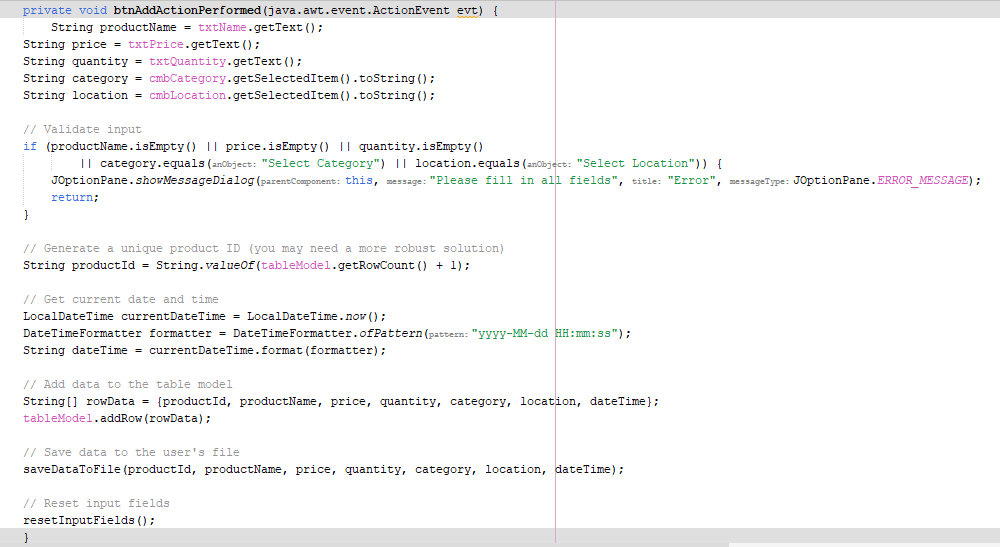


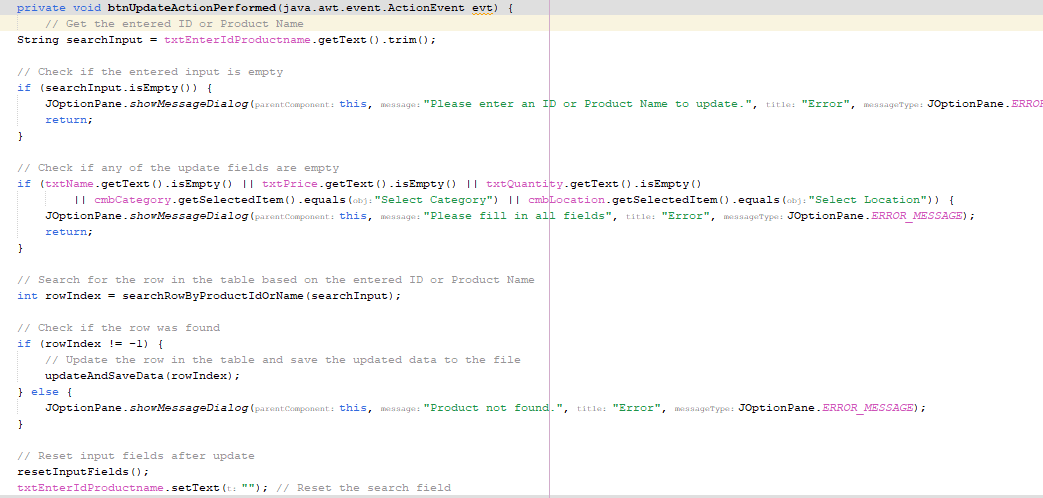


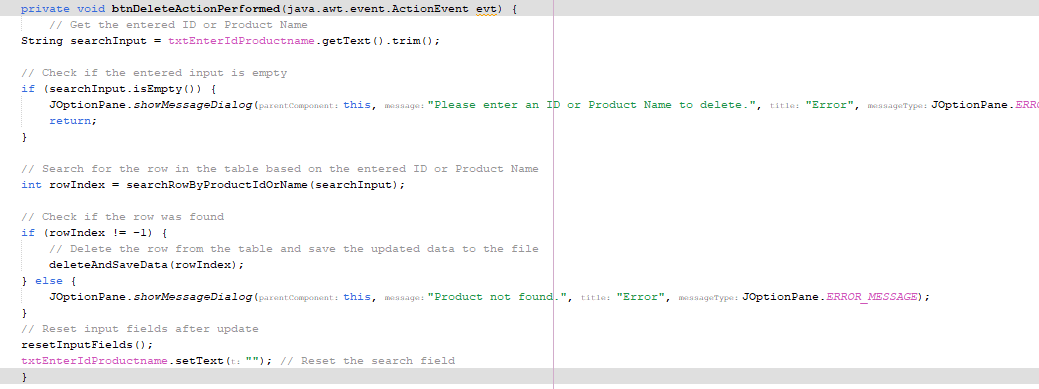


**LoginTab:**

**MenuTab:**





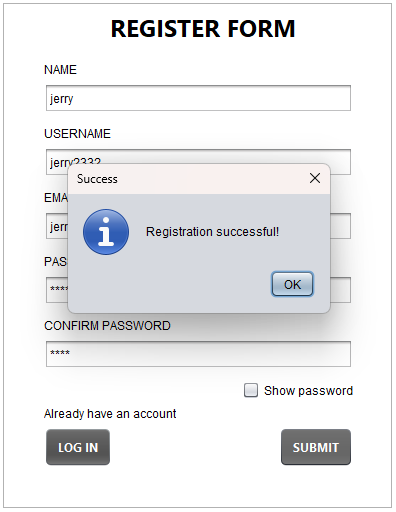
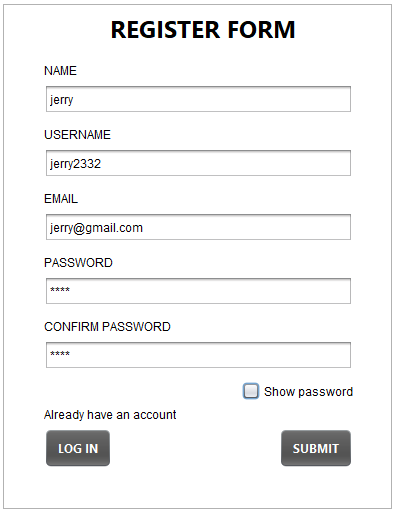


1. **Functionality**

**User Registration:**

**Description:** Users can register by providing necessary information such as email, password, and any other required details.

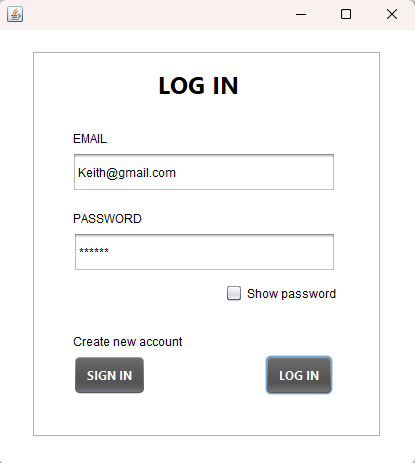
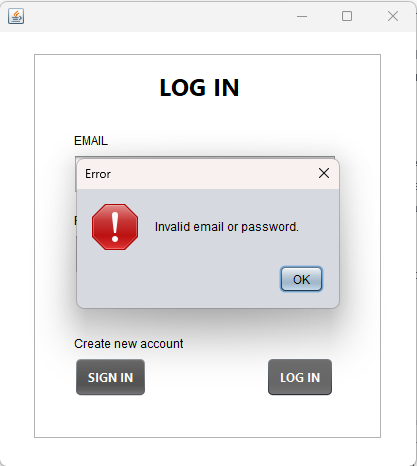
**Implementation:** The RegisterTab class handles user registration. The application ensures that mandatory fields are filled, and it securely stores user data, possibly by hashing passwords.

** **

**Example:** A new user navigates to the registration tab, enters their email, password, and etc. clicks the "Sign Up" button. After successful registration, the user's data is stored securely for future logins.

**User Login:**

**Description:** Registered users can securely log in with their credentials.

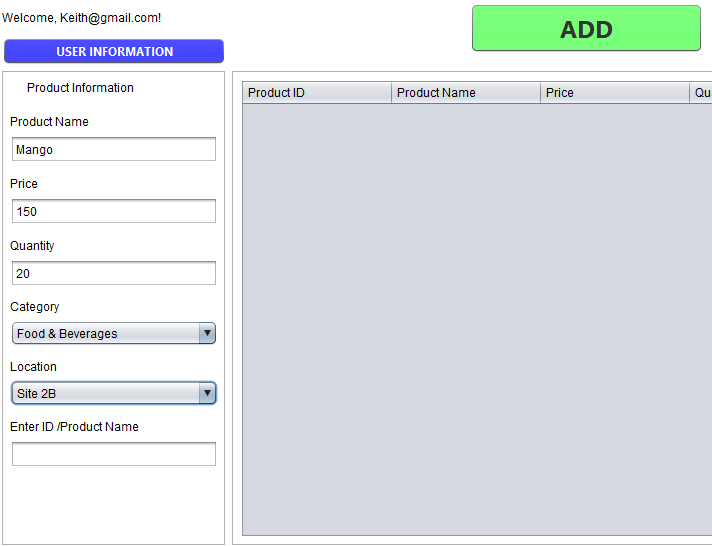
**Implementation:** The LoginTab class handles user login. The application validates the entered credentials against stored information, providing access only if the credentials match.

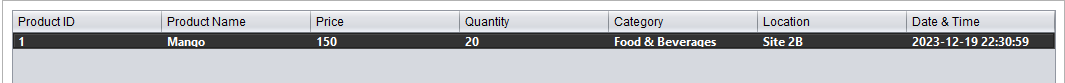
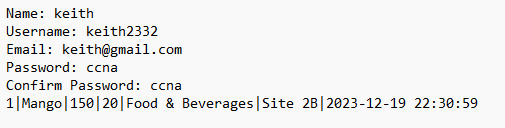
**Example:** An existing user navigates to the login tab, enters their email and password, and clicks the "Log In" button. If the credentials are correct, the user gains access to the system.

**Product CRUD Operations (in MenuTab class):**

**Create:**

**Description:** Users can add new products to the inventory.

**Implementation:** The application provides input fields for product details, and the user clicks a "Add" button to add a new product to the inventory.

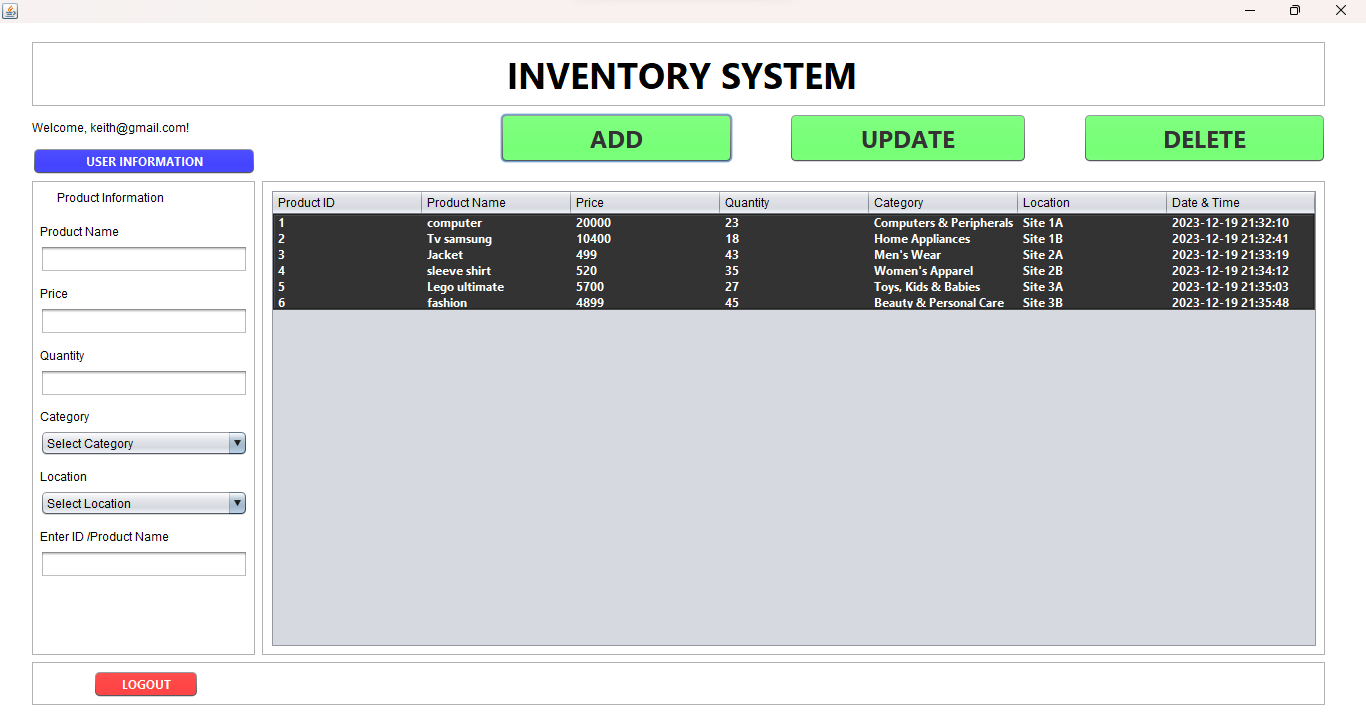


Email.txt

**Read:**

**Description:** Users can view a table displaying existing products in the inventory.

**Implementation:** The MenuTab class initializes a table model and loads existing product data to display in a table.

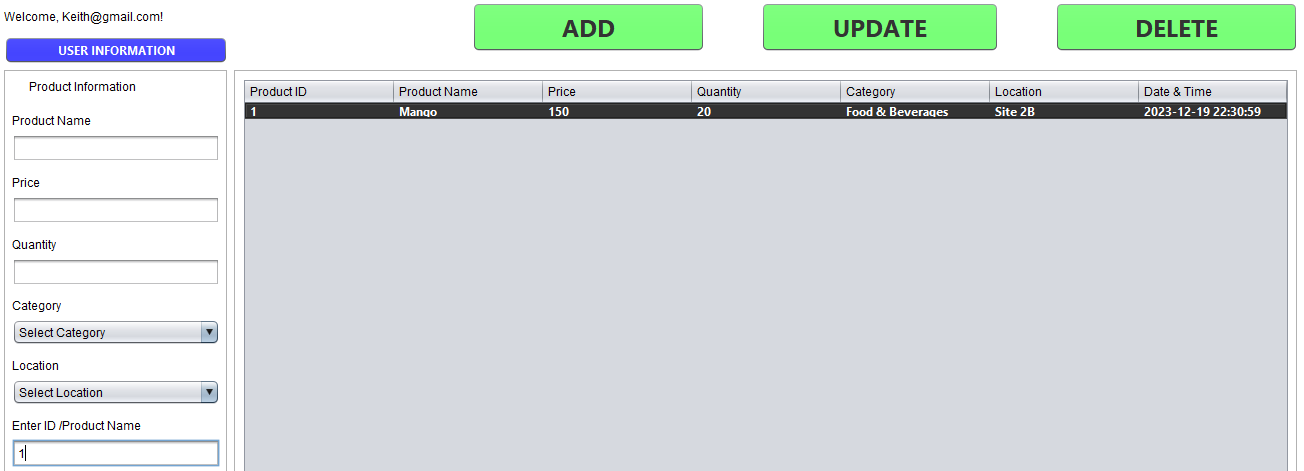


**Example:** A user navigates to the menu tab and sees a table listing all available products with their details.

**Update:**

**Description:** Users can modify information for existing products.

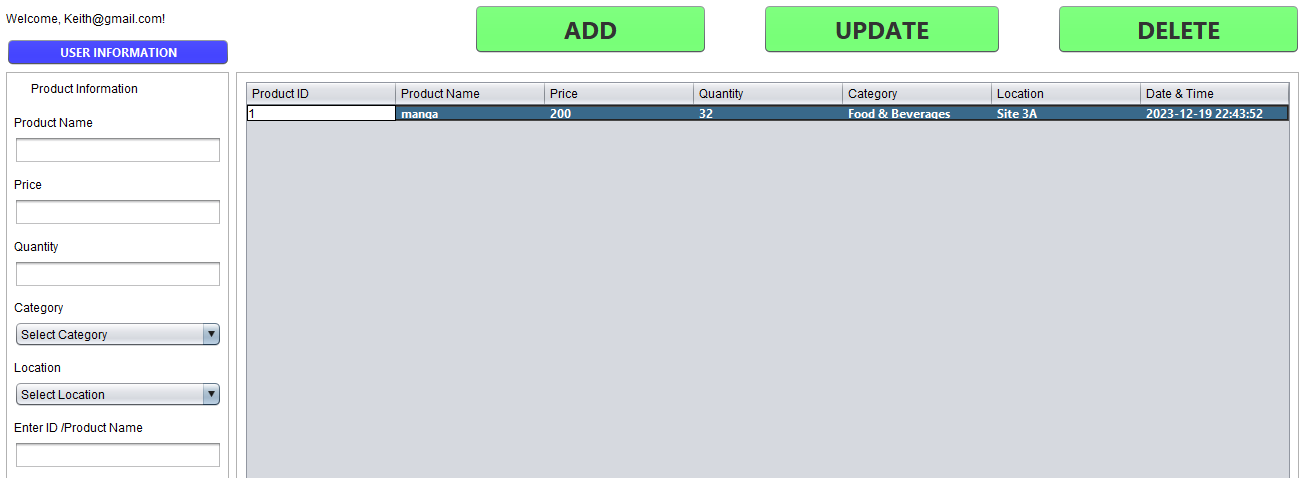
**Implementation:** The application allows users to select a row in the table, modify the product details in input fields, and click an "Update" button.



**Example:** A store employee wants to update the price of a product. They select the product in the table, enter the new price, and click "Update."

**Delete:**

**Description:** Users can remove products from the inventory.

**Implementation:** The application allows users to select a row in the table and click a "Delete" button to remove the corresponding product. 

**Example:** A store manager identifies a product that is no longer available, selects it in the table, and clicks "Delete" to remove it from the inventory.

This comprehensive functionality ensures that the application provides a complete set of features for managing both user accounts and the product inventory system.

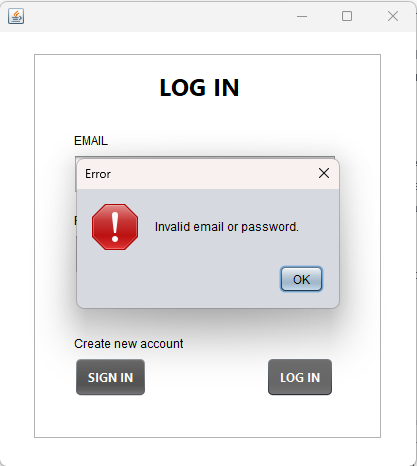
1. **Error Handling and Input Validation**

**Error Handling:**

**File Operations:**

**Description:** The application includes error handling mechanisms to manage potential exceptions related to file operations, such as file not found or I/O errors.

**Implementation:** In methods where file operations are performed (reading/writing), the application uses try-catch blocks to catch IOExceptions. This ensures that if, for example, a user file is not found or there is an issue with reading/writing, the application gracefully handles the error and may display an appropriate message to the user.



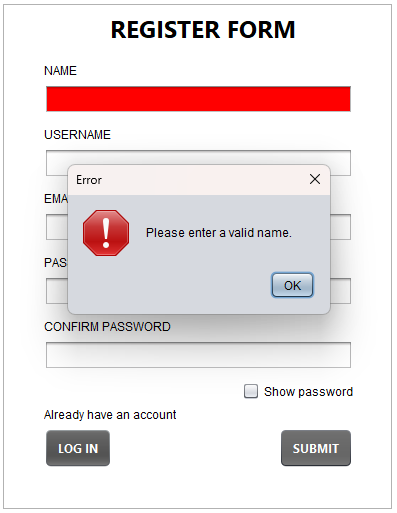
**Example:** When attempting to read user data from a file during login, if the file is not found or an I/O error occurs, the application displays an error message to the user, indicating a problem with the login process.

Input Validation:

**User Registration:**

**Description:** The application validates user inputs during the registration process to guarantee that essential information is provided and follows the required format.

**Implementation:** Before processing the registration, the application checks that mandatory fields (e.g., email, password) are not empty. Additionally, it may perform more specific validation, such as ensuring a valid email format.

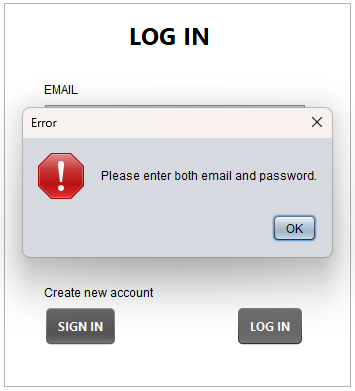


**Example:** During registration, if a user attempts to sign up without entering an email or password, the application displays an error message, prompting the user to provide the required information.

**User Login:**

**Description:** Input validation is applied to the login process to ensure that users enter valid credentials.

**Implementation:** The application checks that both email and password fields are not empty before attempting to validate the login. It may also perform additional checks, such as verifying the email format.

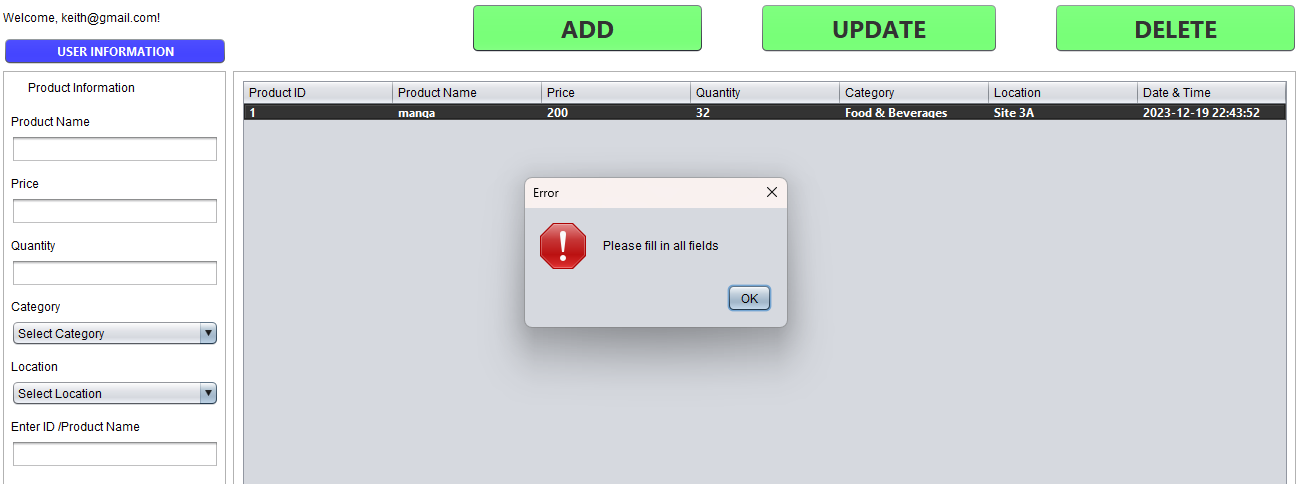


**Example:** If a user tries to log in without entering any credentials, the application displays an error message, prompting the user to provide valid login information.

**Product CRUD Operations:**

**Description:** Input validation is enforced when users add or update product information to maintain data integrity.

**Implementation:** Before adding or updating product information, the application validates input fields, ensuring that essential details are provided (e.g., product name, price). Numeric fields may be checked for valid numbers.



**Example:** If a user tries to add a new product without entering a product name or specifies a non-numeric value for the price, the application displays an error message, preventing invalid data entry.

These error handling and input validation strategies contribute to the overall reliability, security, and user-friendliness of the application by preventing unintended behaviors and guiding users toward correct usage.

1. **User Interface**

**Design and Layout:**

**Java Swing Framework:**

**Description:** The user interface is crafted using the Java Swing framework, providing a platform-independent and visually appealing GUI (Graphical User Interface) for the CRUD application.

**Implementation:** Swing components such as JFrame, JPanel, JLabel, JTextField, JPasswordField, JCheckBox, JButton, and JTable are utilized to create a cohesive and interactive user interface.

**Intuitive Form Elements:**

**Input Forms:**

**Description:** Registration and login screens include intuitive input forms with labeled fields for email, password, and other relevant information.

**Implementation:** JLabels are used to display field names, and corresponding JTextFields or JPasswordFields allow users to input data securely. The use of checkboxes (e.g., Show Password) enhances user interaction.

**Visual Feedback and Alerts:**

**Error Messages:**

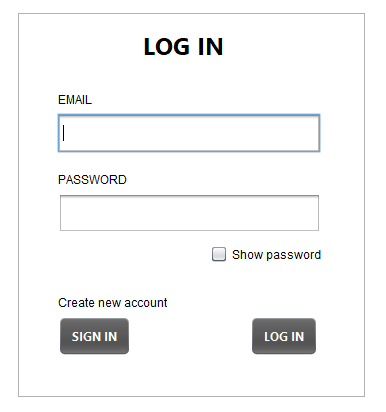
**Description:** The UI provides visual feedback through error messages in case of invalid inputs or operation failures.

**Implementation:** JOptionPane is utilized to display pop-up dialog boxes with descriptive error messages when issues arise, guiding users on corrective actions.

**Inclusion of Screenshots:**

**Description:** Visual aids, such as screenshots or diagrams, can be incorporated to illustrate the actual appearance of the application's UI.

**Implementation:** Screenshots can be captured during different stages of interaction, showcasing the login screen, registration form, and product management interface.



**Example/Scenario:** An accompanying diagram can visually represent the layout and design of the login screen, providing a concrete reference for users.

1. **Challenges Faced**

Reyes:

**File I/O Operations:**

**Challenge:** Implementing file I/O operations for reading and writing user and product data files posed challenges, especially handling file paths and potential exceptions.

**Solution:** Robust exception handling mechanisms were implemented to address potential issues like file not found or IO exceptions. Collaborative debugging sessions helped identify and rectify specific issues in file operations.

**User Interface Design:**

**Challenge:** Designing an intuitive and visually appealing user interface using Java Swing required a balance between aesthetics and functionality.

**Solution:** Regular design discussions within the buddy and seeking feedback from my buddy helped refine the UI. Leveraging the Java Swing framework's documentation and examples aided in overcoming specific design challenges.

**Communication Between UI Components:**

**Challenge:** Ensuring effective communication between different UI components, especially during transitions between tabs, posed coordination challenges.

**Solution:** A standardized approach to event listeners and method calls between UI components was established. The we implemented clear protocols for data transfer and UI updates to maintain cohesion during transitions.

**Valledor:**

**Secure Password Storage:**

**Challenge:** Implementing secure password storage practices to protect user credentials required careful consideration of encryption or hashing techniques.

**Solution:** We researched and implemented secure password hashing using industry-standard algorithms. Regular code reviews and consultation with security best practices helped fortify the authentication system.

**Handling Tab Transitions:**

**Challenge:** Managing the transition between different tabs (JFrames) without losing data or causing inconsistencies in the application state.

**Solution:** A well-defined protocol for cleaning up resources and saving data before tab switches was established. We utilized methods like dispose() to close frames gracefully and transfer relevant data between tabs.

**Collaborative Debugging:**

**Challenge:** Identifying and resolving bugs in a collaborative coding environment required effective communication and debugging strategies.

**Solution:** Regular collaborative debugging sessions were conducted using version control tools. We utilized code comments, logging, and debugging tools to pinpoint and resolve issues efficiently.

1. **Individual Reflection**

**Reyes:**

**Contributions:**

I primarily focused on the implementation of file I/O operations and ensuring a robust data storage mechanism.

Actively participated in collaborative debugging sessions and provided insights into optimizing file-related code.

**Lessons Learned:**

Improved my understanding of Java Swing and file handling in GUI applications.

Developed skills in handling exceptions and debugging complex I/O issues.

**Valledor:**

**Contributions:**

Specialized in implementing secure password storage practices and enhancing authentication processes.

Contributed to the design discussions for the user interface.

**Lessons Learned:**

Gained expertise in secure password hashing techniques.

Enhanced skills in UI design considerations and Java Swing.

**Conclusion:**

The challenges faced during the implementation provided valuable learning opportunities, and the solutions adopted contributed to the overall success of the project. The experience highlighted the importance of effective communication, collaborative problem-solving, and continuous learning in a team-based development environment.