Comprehensive Exercise Report

Team DIP392-FutureFrame

1.Osura Wijeweera 221ADB215
2. Lahiru Atigala 221ADB184
3. Avishka Sandun 221ADB142
4.Janith Kandambige 221ADB178
5.Isuri Nilupuli 221ADB141
6.Keyan Gisadh 221ADB140
7.Ravika mahapatabnedhige 213AEB041

Requirements/Analysis

Week 2

Journal

The following prompts are meant to aid your thought process as you complete the requirements/analysis portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- After reading the client's brief (possibly incomplete description), write one sentence that describes the project (expected software) and list the already known requirements.

  - Handle the customer (Restaurant) Kitchen inventory management application
  - Provide real-time tracking of stock levels.
  - Automate stock adjustments based on usage.
  - Offering decision support for order management.
  - Facilitate item management for adding/removing stock items.
  - Implement user access control for secure system use.
  - Add real time updates about users who updated last.

- After reading the client's brief (possibly incomplete description), what questions do you have for the client? Are there any pieces that are unclear? After you have a list of questions, raise your hand and ask the client (your instructor) the questions; make sure to document his/her answers.

  - **User Role Clarification**

    It's mentioned that employees and management can log in, but it's unclear What specific permissions each role should have Stock Update Process
    It's not clearly defined how stock is updated

  - **Unit Measurement**
    The brief does not specify how stock quantities are measured:

  - **Low Stock Alerts**
    The system's reaction to low stock isn't specified.

  - **Security and Data Access**
    There is no information on how secure the system needs to be.

  - **Reporting Requirements**
    The brief doesn't clearly define reporting features.

  - **Callability**
    The brief focuses on one restaurant but doesn't mention future expansion.

Q1) **How many users (employees/managers) will access the system simultaneously?**

Total users are 10 including one manager

Q2) **Do you require separate login roles for admin, manager, and kitchen staff?**
Yes! Manager Should has access to add or remove users and rest of users only able to use the system

Q3) **Should stock be deducted automatically after a Delivery, or only after a staff update?**
No, it should be updated only with staff update

Q4) **Do you want to track expiry dates for perishable items like dairy and meat?**
Yes! We need a system which has update about expiration dates about perishable items

Q5) **Is there a reorder level or low-stock threshold you'd like to be alerted about?**
Yes! Our main expectation is to have low stock alerts for the products that we will going to runout.

Q6) **Should the system support printing or exporting reports**
Yes, I will be good to have printable materials for accountant

Q7) **Should the system support multiple restaurants or branches in the future?**
Currently we have only one location and one main restaurant so far it won't need.

Q8) **Will employees need access on mobile devices, or just desktop systems in-store?**
We are expecting to have all device accessible system

Q9) **How system need to alert about low stocks by email or notification or text message?**
It is good to have a notification in the system.

Q10) **What unit measurements are expected in the system?**
For each item there is metrics by liters grams kilograms and pcs

- **Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.**
  - No

- **Describe the users of this software (e.g., small child, high school teacher who is taking attendance).**
  - Head Chef
  - Cooks
  - Kitchen helpers
  - Restaurant Manager
  - Restaurant owner

- **Describe how each user would interact with the software**
  - Cooks – they can insert and update the amount oof items that they are consuming for the service in the restaurant and notify the head chef about the missing products
  - Head Chef- Can update the stocks and order through the stock requirements
  - Kitchen helpers - they can also insert the product inventory in the kitchen and update the stock in the kitchen.
  - Restaurant Manager - the manager can add or remove kitchen members to the system. Also check who updates the inventory and giving updates real time check for low stocks and download report for the accountant.

- **What features must the software have? What should the users be able to do?**

    - Managers can add or remove the users and update the stock
    - The chef can add or remove food products to the systems
    - Can make the stock level alarm in the system.
    - Filter the products in the inventory
    - Manager can see the logs who update the items

- **Other notes:**

- The system should be scalable to support additional restaurant branches in the future.
- Data such as passwords must be stored securely using encryption.
- The interface should be responsive and accessible on tablets.
- The application should support input validation to avoid incorrect stock updates.
- Users should be shown clear success/error messages for all actions.
- Regular data backup options should be considered for recovery in case of data loss.

Software Requirements

The Web-Based Inventory Management System is designed to help Manana Sia Riga, a Mexican food and beverage restaurant, efficiently track and manage its inventory. The system will provide real-time stock tracking, automatic stock adjustments, order decision support, item management, and user access control. By implementing this system, the restaurant aims to reduce food wastage, optimize ordering processes, and enhance operational efficiency.

Black-Box Testing

Instructions: Week 4

Journal

Remember: Black box tests should only be based on your requirements and should work independent of design.

The following prompts are meant to aid your thought process as you complete the black box testing portion of this exercise. Please review your list of requirements and respond to each of the prompts below. Feel free to add additional notes.

- What does the input for the software look like (e.g., what type of data, how many pieces of data)?
  - **Login credentials**: Email (string), Password (string)
  - **Product data**: Add item(String), Quantity (Float), Unit(Int), Category(Dropdown)
  - **Add User**: Full Name (string), Email (string), Password (selection: manager/chef), Role (Selection)
  - **Search/filter**: Keywords (string), Category (optional dropdown)
  - **Stock alert thresholds**: Quantity level (integer)

- What does the output for the software look like (e.g., what type of data, how many pieces of data)?
  - **Login success/failure messages** (text)
  - **Inventory lists**: Table of items with columns for name, category, quantity, Threshold, Updated By etc.
  - **Confirmation messages**: For adding, updating, or deleting stock/users
  - **Filtered lists**: Matching products only
  - **Stock alarms**: Visual indicators or alert messages (e.g., "Low stock!")

- What equivalence classes can the input be broken into?

Input Type	Valid Equivalence Class	Invalid Class
Login Email	Existing email	Blank or unregistered email
Password	Correct password	Blank or incorrect password
Quantity	Positive integer	Negative or non-numeric value
Category	Selected from list	Not selected / invalid option
Search Keyword	Valid keyword	Empty / special characters

- What boundary values exist for the input?

Input Field	Boundary Values
Quantity	0, 1 (minimum allowed stock), maximum capacity
Product name length	0 chars (invalid), 1 char (min valid), 255 chars (max length)
Price	0.00 (free), max decimal value (e.g., 9999.99)
Email	Empty (invalid), valid format, overly long address (invalid)

- Are there other cases that must be tested to test all requirements?

  - Try to log in with invalid credentials
  - Use a user with a similar email

- Testing after removing the users and log in
- Adding a product without any quantity
- Filtering by category with no results
- Update the products and make the changers
- Logged as manager and manage users

- Other notes:

- Testing must be possible to track adding, removing and updating items in the inventory
- Low Stock values must be verified

Black-box Test Cases

Use your notes from above to complete the black-box test plan section of the formal documentation by writing black box test cases (other than actual results since no program currently exists). Remember to test each equivalence class, boundary value, and requirement.

Test ID	Description	Input	Expected Result	Actual Result
TC01	Valid login credentials	Email: admin@test.com, Password: admin123	Login successful	Login successful
TC02	Invalid password	Email: admin@test.com, Password: wrongpass	Error: Invalid credentials	Error shown
TC03	Add new product with valid data	Name: Rice, Category: Food, Qty: 100, Price: 1.50	Product added successfully	Product added successfully
TC04	Add product with zero quantity	Qty: 0	Stock alert triggered	Alert shown
TC05	Search with keyword returning results	Keyword: Rice	Product list filtered correctly	Filtered list shown
TC06	Search with keyword returning no results	Keyword: InvalidItem	Message: No results found	No results found
TC07	Add user with duplicate email	Email: admin@test.com	Error: Email already exists	Error shown
TC08	Delete user and attempt login	Deleted user credentials	Error: User not found	Login blocked
TC09	Enter negative quantity	Qty: -5	Error: Invalid quantity	Error shown
TC10	Filter by category only	Category: Beverage	Product list filtered by category	Filtered list shown
TC11	Password left blank at login	Email: user@test.com, Password: (empty)	Error: Password required	Error shown

Design

Instructions: Week 6

Journal

Remember: You still will not be writing code at this point in the process.

The following prompts are meant to aid your thought process as you complete the design portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- List the nouns from your requirements/analysis documentation.

  - Inventory
  - Product
  - Stock
  - Category
  - User
  - Manager
  - Chef
  - Supplier
  - Report
  - Log
  - Threshold
  - Alert
  - Login
  - Dashboard

- Which nouns potentially represent a class in your design?

  - Product
  - User
  - Inventory
  - Category
  - Supplier
  - Log
  - Alert

- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.

Attribute	Class
name	Product, User, Supplier
quantity	Product
price	Product
category	Product
role	User
email	User
password	User
date_added	Product, Log
action	Log
threshold	Alert

- Now that you have a list of possible classes, consider different design options (***lists of classes and attributes***) along with the pros and cons of each. We often do not come up with the best design on our first attempt. Also consider whether any needed classes are missing. These two design options should not be GUI vs. non-GUI; instead you need to include the classes and attributes for each design. Reminder: Each design must include at least two classes that define object types.

- Product (id, name, category, quantity, price, supplier_id)
- User (id, name, email, password, role)
- Log (id, user_id, action, timestamp)

Pros:

- Simple and easy to implement
- Suitable for small-scale systems
- Direct mapping to relational database tables

Cons:

- Lacks abstraction
- Harder to maintain when scaling
- Logic for role-based access needs to be implemented separately

- Which design do you plan to use? Explain why you have chosen this design.

As a learning team we used an innovative design for our Inventory Management System, we used main PHP files like login, inventory updates, and user management. Instead of using object-oriented classes, we arranged the system functionally, with clear separation between features.

We included role-based access control, allowing managers and chefs to perform different actions, and used a MySQL database to manage users, products, and logs. We used this method as student we all learn about these in our degree, and it helps implement and allowed us to focus on building a functional system within our time frame

- List the verbs from your requirements/analysis documentation.

  - Login
  - Add
  - Remove
  - Update
  - Filter
  - Alert
  - View
  - Manage
  - Track
  - Assign

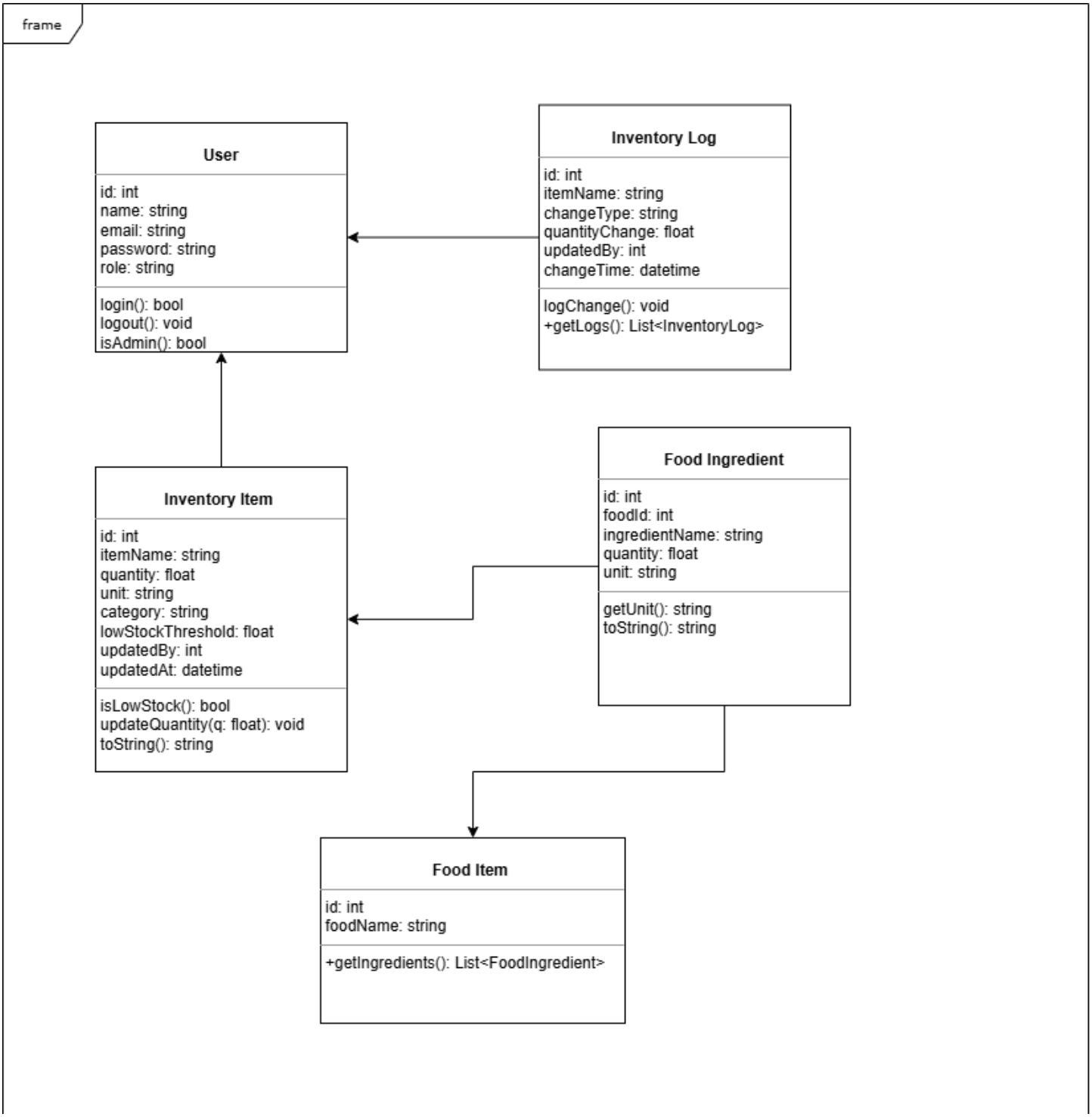
- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.

Method	Class
login()	User
addProduct()	Manager, Chef
removeProduct()	Manager, Chef
updateStock()	Inventory
filterProducts()	Inventory
triggerAlert()	Alert
viewLogs()	Manager
addUser()	Manager
removeUser()	Manager

- Other notes:

  - The system should enforce role-based permissions on methods as example only Manager can call `addUser`.
  - Consider using inheritance for shared methods

Software Design



Implementation

Instructions: Week 8

Journal

The following prompts are meant to aid your thought process as you complete the implementation portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- What programming concepts from the course will you need to implement your design? Briefly explain how each will be used during implementation.
    - **Modular Programming**-Each feature is divided into separate modules, and it will help with code reuse, testing, and maintainability.
    - **Object-Oriented Programming (OOP)**-Implemented as classes to encapsulate data and functionality, enabling inheritance and polymorphism are the core system components.
    - **UI Design Principles**-We create a clear, consistent, and user-friendly design using user interface slides. Menu navigation, confirmation dialogs, and input fields will follow usability rules.
    - **Data Validation and Error Handling**-Include client- and server-side validation will be every input form. Descriptive, non-judgmental, and easy to correct error messages will follow best practices.
    - **Testing (Unit & Integration)**-Calculations and data processing critical functions used to unit test according to the guidelines of SDLC. interaction between modules will validate by Integration test.
    - **Version Control (GitHub)**-Per the lecture slides we tag versions as Alpha (incomplete), Beta (user-tested), and Production (final). And Implementations will be committed with meaningful commit message regularly.
    - **Deployment Strategy**-To Minimize and simplify feedback collection we follow phased Deployment to roll one module at a time.

- OtherNotes
    - We used wireframes and mockups early to improve interface consistency and ease to use.
    - For the functions written only we used agile build to test concept when already defined
    - Training and possibly some help to document embedded into the UI and it will include in the deployment.

Implementation Details

In our system we have use some main categories and, in our database, we used phpMyAdmin panel and also in that we used manana-inventory. In that database we have implemented some main class to the system.

We have used,
User: It handles login, logout, and arrange permission.
Inventory_Item: it will show the items in the inventory and can make action after even inserting it into the system.
Inventory_Log: It will record the history of the inventory and save who updated the system items.
Food_Items: This will store the main menu items of the system.
Food_Ingredients: This will show the ingredients that are used for food items.

Used features:

Automatically indicate the low stock as a warning.
User authentication based on role that assigned
Automatically selecting the foods according to the category that is assigned.

README: Guided instructions

Login: In this interface we can use our registered Email and Password, adding user is fully responsible by the restaurant manager. Users can define into 2 parts as Admin and Staff

View Inventory: this is the main dashboard that we used to show up the items that are listed and it will show from name to low stock feature as we mentioned earlier.

Update Inventory: In this we can update the current inventory and add or remove new quantity in the system.

View Logs: In this feature we can track who updates the system and what are the stock that we really needed

Manage Users: This function mainly applies for the Manager/Admin of the system; they can add or remove a staff member from the system.

Testing

Instructions: Week 10

Journal

The following prompts are meant to aid your thought process as you complete the testing portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- Have you changed any requirements since you completed the black box test plan? If so, list changes below and update your black-box test plan appropriately.
    - Yes after testing we added the following
        - Role-based feature restrictions
        - Customizable stock level thresholds
- List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
    - User
        - Login success/Faliure
        - Role validation
        - Logging in and out triggers appropriate state
    - InventoryItem
        - Check lowstock at below and above threshold
        - Item creation with full and partial data
        - Testing update quantity with negative and positive values
    - Inventorylog
        - Log entry creation
        - Createing correct log entries
        - Retrieval of logs for an item
    - Foodingredient
        - Calling getUnit() and toString() outputs expected format
        - Ingredient added to food item with valid link
    - Fooditem
        - Handle food items with and without ingredients
- Other notes:
    - Test linking across classes (FoodItem to FoodIngredient to InventoryItem)
    - Validate Inventory Log triggers on item change
    - Confirm role-based actions (e.g., only managers can update inventory)
    - Validate time/date fields are captured correctly on updates
    - Verify system behavior under concurrent logins and inventory updates

Testing Details

Test Script (PHP File)	Description
test_login.php	Tests the login system with valid and invalid email/password combinations. Validates session start and role redirection.
test_add_inventory_item.php	Adds new inventory items using POST data. Tests handling of valid, zero, and negative quantities.
test_user_management.php	Tests adding users with unique and duplicate emails, role assignments, and login access.
test_inventory_log.php	Verifies logs are generated on item addition, update, or deletion by checking database log table.
test_search_filter.php	Tests the product search bar and category filter with matching and non-matching inputs.
test_food_ingredients.php	Links food items to ingredients and checks if ingredient data is correctly stored and retrieved.
test_role_permissions.php	Simulates manager and chef actions (e.g., deleting a user) to validate access control logic.

Test ID	Description	Input	Expected Result	Actual Result (via PHP)	Status
TC01	Valid login	email=admin@test.com , pass=admin123	Redirect to dashboard	Redirected with session set	Pass
TC02	Invalid login	Wrong password	Error message shown	"Invalid credentials" shown	Pass
TC03	Add item: valid	Rice, 100 qty, Food	Added to DB, success alert	Insert successful, alert shown	Pass
TC04	Add item: qty 0	Qty = 0	Warning: Low stock alert	Alert triggered post-insert	Pass
TC05	Search item	Keyword: “Rice”	Show relevant items	Table updated with matching items	Pass
TC06	Search no result	Keyword: “Xyz”	Show “No items found”	Message displayed correctly	Pass
TC07	Add duplicate user	Email already exists	Error: Email taken	Duplicate email warning	Pass
TC08	Delete user test	Delete & try login	Login fails	"User not found" message	Pass
TC09	Negative qty	Qty: -5	Reject input	Form validation error	Pass
TC10	Category filter	Category = “Drinks”	Filtered list shown	Matching items listed	Pass
TC11	Blank password login	Only email entered	Prompt password required	Validation error shown	Pass
TC12	Inventory log on update	Update qty → check DB	Log record created	Entry in inventory_log table	Pass
TC13	Manager deletes user	Logged in as manager	User removed	Deleted from DB	Pass
TC14	Chef deletes user	Logged in as chef	Access denied	"Unauthorized" shown	Pass
TC15	Get ingredients	Get Pizza’s ingredients	List retrieved	Correct list shown from DB	Pass

Presentation

Instructions:Week 12

Preparation

- Give a brief description of your final project
    - Our software is a inventory management system that is designed to a restaurant which is designed using php and mysql. The system mainly focuses on managers (Admin) and chefs to login to the system to manage, track qty and receive low-stock alerts for the food items.
- Describe your requirement assumptions/additions.
    - We assumed two user roles manager who has full access and chef with restricted access
    - We aslo assumed that the system will be accessed through a web browser on a local network
- Describe your design options and decision. How did you weigh the pros and cons of the different designs to make your decision?
    - We chose the relational design because it made the system more scalable and normalized
    - We selected Mysql due to its reliabilty and query support.
- How did the extension affect your design?
    - Adding support for inventory logs we had to create new tables and update our code to support joins
- Describe your tests (e.g., what you tested, equivalence classes).
    - We used black-box testing
        - Equivalence Classes- valid/invalid, positive/zero/negative quantities valid/invalid category filters, duplicate users, low-stock conditions.
        - We also tested- Login, Adding/ editing inventory items, role-based access, Filtering
- What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
    - We learned to work as a team using Github for version and code review.
    - Learned the importance of requirment clarity.
    - We gaoned experience in database normalization
- What functionalities are you going to demo?
    - Login as manager and chef.
    - Add/edit inventory item
    - Trigger and view low stock alert
    - Filter an search inventory
    - Show log entries after item updates
- Who is going to speak about each portion of your presentation? (Recall: Each group will have ten minutes to present their work; minimum length of group presentation is seven minutes. Each student must present for at least two minutes of the presentation.)

    - Avishka Sandun(221ADB142) - Title Slide, Project Overview & Motivation
    - Osura Wijeweera (221ADB215)- SDLC Methodology, Conclusion and Future Work
    - Lahiru Atigala (221ADB184)- Requirements Gathering, Software Requirements Specification
    - Isuri Nilupuli (221ADB141- System & Software Design, Implementation Phase
    - Keyan Gisadh (221ADB140)- Testing Strategy, Test Results & Validation
    - Ravika Mahapatabnedhige (213AEB041)- Process Reflection, Demo Overview