gyst

SINCE 2021

organization? you get the gyst.

★ Super Jumbo AAAA Batteries ★

Introduction

The Specification

- *gyst* is a web application that organizes your household inventory and assists you in your everyday life.
- In the beginning, the list of needed items will be initialized, and the user can add items to that list by logging items one by one.
- The user can also remove items from their inventory or shopping list one item at a time.
- The user can see their updated inventory and shopping lists.
- There is an option for the user to save their inventory and shopping list for later use.
- This website will allow the user to be efficient and organized by decluttering their life.

gyst SINCE 2021
organization? you get the gyst.

Changes from Phase 1 to Phase 2

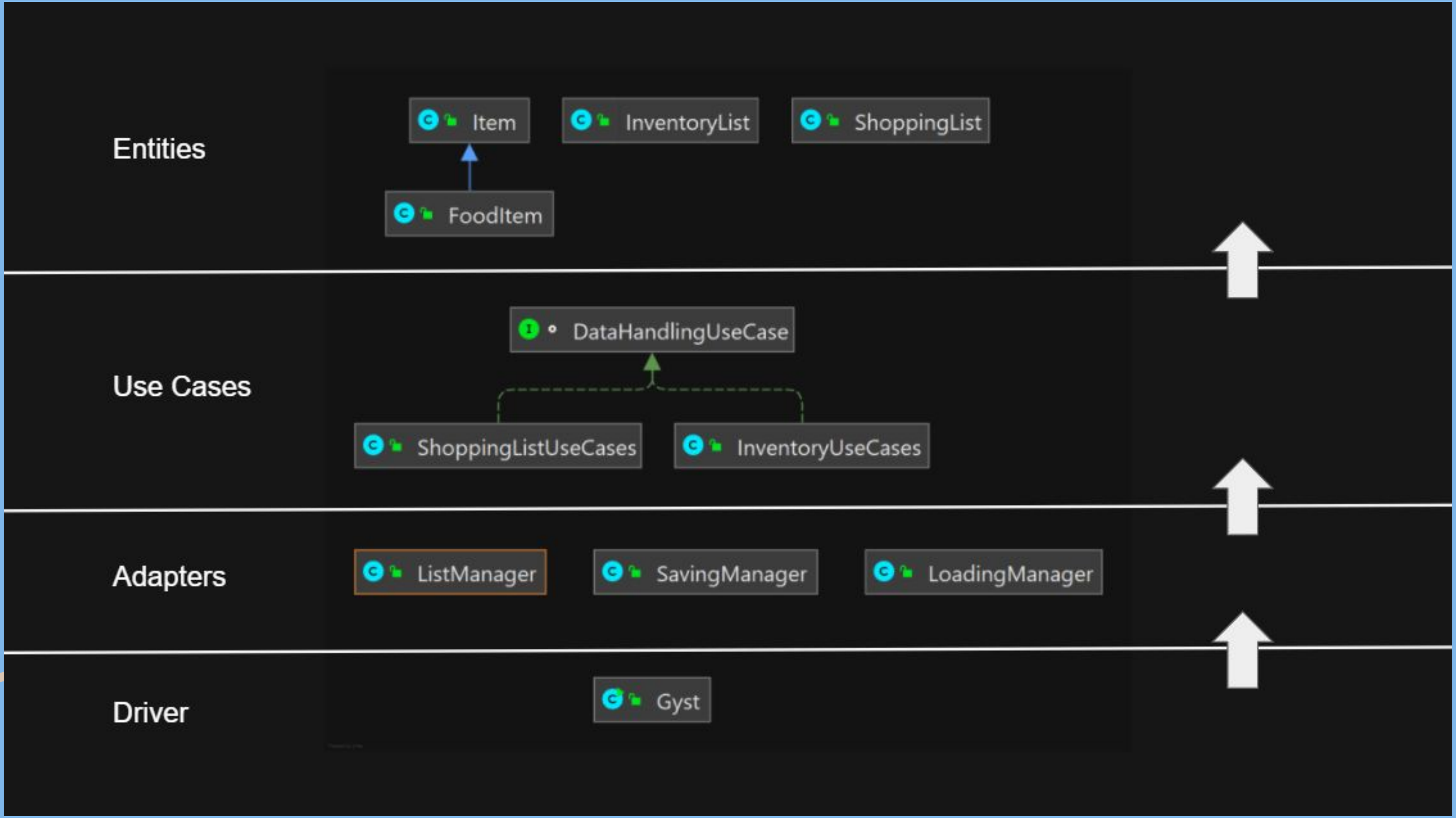
- Added entities, use cases, a controller, and unit tests for user authentication.
  - Updated our data persistence/serialization (via CSV files)
- Connected the program to the website.
- Added the user authentication to the website and connected it to the relevant Java code.
  - Added unit tests for this
- Implemented the strategy design pattern, which we used to sort the expiry dates of items in a user's inventory in terms of closest date to expiration.
  - Added unit tests for this

gyst
SINCE 2021
organization? you get the gyst.

Web Demo

Implementation

Clean Architecture – UML



Clean Architecture

Layers of Clean Architecture in the Inventory System and in the Login System

(credit to the Computer Science teaching team)

- Driver
- Controller
- Gateway
- Use cases
- Entities

gyst SINCE 2021
organization? you get the gyst.

SOLID Principles

- ❖ Single Responsibility principle
- ❖ Open/Closed principle
- ❖ Liskov Substitution principle
- ❖ Interface Segregation principle
- ❖ Dependency Inversion principle

gyst SINCE 2021
organization? you get the gyst.

Major Design Decision

- ❖ User-Specific Data Storage System
- ❖ Inheritance vs Interface

gyst SINCE 2021
organization? you get the gyst.

Design Nuances

Design Patterns

- Dependency injection
- Private class data
- Strategy

organization? you get the gyst.

Dependency Injection Design Pattern

- To add an Item or FoodItem to an InventoryList object, the InventoryUseCases class creates the Item and FoodItem objects and injects them into the InventoryList.
- This way, we avoid a hardcoded dependency between the Item/FoodItem objects and the InventoryList.
- Both the superclass Item and its subclass FoodItem can be added to the InventoryList.

gyst SINCE 2021
organization? you get the gyst.

Private Class Data Design

- The point of this design pattern was to reduce the exposure of instance variables by limiting their visibility outside of the class
- Variables cannot be randomly changed without the class' "permission."
- For example, the Item entity class has private name and quantity instance variables so that other classes cannot access them and possibly change them to an unexpected value.

Strategy Design Pattern

- Sort items by expiry date or alphabetically by name if the item does not have an expiry date.
- Helper classes:
  - Sorter interface
  - TimSorter class
- Item & FoodItem entities:
  - Implement Comparable
  - compareTo() method
- InventoryList & ShoppingList entities:
  - sortItems() method
- Use cases for inventory and shopping list:
  - sortInventory()
  - sortShoppingList()
- Controller:
  - saveLists()
- Sort inventory and shopping list when user saves them.

gyst SINCE 2021
organization? you get the gyst.

Code Organization

Packaging Strategy

The packaging strategies we considered:

Packaging By Layer Vs. Packaging By Feature

gyst
SINCE 2021
organization? you get the gyst.

Testing

Coverage:

- ❖ Tested user authentication (controller, use cases, entities)
- ❖ Tested design patterns
- ❖ Tested all other user cases, controllers, and entities

organization? you get the gyst.

Progress Report

Author Details

Ayanaa: Refactored variable names to follow proper style conventions. Assisted with the creation of use cases and entities for user authentication, unit tests for authentication (use case and controller). Assisted with project accessibility report and design document.

Jennifer: Implemented the strategy design pattern to sort the items in inventory and shopping list. Created an interface for the common methods of InventoryList and ShoppingList entities. Refactored code, such as moving the toStringBuilder() method from the entity layer to the use case layer. Assisted with project accessibility report and design document.

Alissa: Assisted with the creation of use cases and entities for user authentication, unit tests for authentication. Assisted with project accessibility report and design document. Also assisted in the development of the login system.

Ali: Updated data persistence/serialization (via CSV files) to save user information and the user's corresponding inventory and shopping lists. Assisted with project accessibility report and design document.

Aamishi: Worked front-end/on the outer layer. Unit tests for user authentication. Assisted with project accessibility report and design document.

Sam: Worked front-end/on the outer layer. Connected website to Java code. Assisted with project accessibility report and design document.

gyst SINCE 2021
organization? you get the gyst.

What Worked Well With Our Design

- We followed the dependency rule of Clean Architecture (as shown in the UML diagram).
- We consistently checked whether our implementation/design choices adhere to the five SOLID principles of object oriented design.
- We organized and packaged our code efficiently and effectively.
- The design patterns work well.

gyst SINCE 2021
organization? you get the gyst.

Thank You!
Questions?