THE UNIVERSITY OF
SYDNEY

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - GROUP ASSESSMENT

Unit of Study: Agile Software Development Practices (SOFT2412/COMP9412)

Assignment name: Group project Assignment 2 – Agile Software Development with Scrum and Agile Tools

Tutorial time: Tuesday 10:00 AM **Tutor name:** Ziyu Liu

Project team members				
Student name	**Student ID**	**Participated**	**Agree to share**	**Signature**
1. Patrick Hao	440299088	[✓ Yes] / No	[✓ Yes]/No	P.H.
2. Voon Ken Ren	480219084	[✓ Yes] / No	[✓ Yes] / No	VKR
3. Kunxi Sun	490375848	[✓ Yes] / No	[✓ Yes] / No	K.S.
4. Oscar Gao	480365749	[✓ Yes] / No	[✓ Yes] / No	O.G.
5. Alex Wong	470066919	[✓ Yes] / No	[✓ Yes] / No	A.W.
6.		Yes / No	Yes / No	
7.		Yes / No	Yes / No	
8.		Yes / No	Yes / No	
9.		Yes / No	Yes / No	
10.		Yes / No	Yes / No	

SOFT2412 Assignment 2

1. Roles

Product Owner: Patrick 440299088

Responsible for:
- liaising and maintaining a positive working relationship with the client.
- maximising the value of the end-product produced by the development.
- maintaining the product backlog and ensuring that items are ordered in a manner that will allow the development team to efficiently achieve their goals and missions.
- ensuring the backlog is visible, transparent and clear to all.

Scrum Master: Ken 480219084

Responsible for:
- ensuring that team objects, scope and product domain are understood by all on the scrum team
- finding techniques for effective product backlog management
- helping the scrum team understand the need for clear and concise product backlog items
- facilitating scrum events
- removing obstacles faced by the development team

Developer 1: Kunxi Sun 490375848
Developer 2: Oscar 480365749
Developer 3: Alex Wong 470066919

Responsible for using their interdisciplinary skills to participate in producing the required product

2. Sprint Planning & Goal

2.1. Sprint Planning

The sprint planning meeting was held in tandem with the first daily scrum. After meeting the client (in the tutorial), the Product Owner prepared the initial Product Backlog (see 3.1., note: this backlog is also the refined product backlog, see 4.4.2 for explanation) where the work was ordered based on priority and given story points. The team agreed upon this and used it to form the Sprint Goals (see 2.2).

2.2. Sprint Goals

- Create project scaffold to to make starting on the code more structured.
  - Includes all relevant objects like Food, ShoppingCart, Inventory, etc.
- Implement a simple command line UI so that users may interact with the system.
- Implement some basic functions for customer’s to make purchases.
- Implement basic staff specific features so that the staff may restock inventory.

- Implement a records class, with relevant features so that staff members can check a record of inventory.
- Implement a method for customers to pay for snacks by specific payment methods, so that customers can specify exactly how they want to pay.
    - E.g. a $1 can be paid with 2, 50c coins.
- Implement the ability to pay with different currency so that customers can make payments in different conversion rates.

3. Task Board: Product & Sprint Backlog

3.1. Product Backlog

The Product Backlog contains a list of all product features that we have estimated using the user stories. It is also ordered by business value to the client. The Product Owner, Patrick, is responsible for its management, including its content, availability and ordering.

The Product Backlog is dynamic and changes constantly. Any additional features should be added directly to the Product Backlog instead of the Sprint Backlog. However, these additions were essential to the implementation of other features, hence we added it to the Sprint Backlog. The context for such changes are given in the Sprint backlog.

Product Backlog (Desc. Order by Business Value)	Role	Story Points
Payment with different currencies (Functionality)	Collaborative	5
Select from a variety of snacks (Functionality)	Ken	3
Select a snack (Functionality)	Ken	3
Select different quantities of snacks (Functionality)	Quency	5
Remove different quantities of snacks (Functionality)	Quency	2
Paying for Food (Functionality)	Oscar	3
Ensure user can purchase a lot of snacks (Functionality)	N/A	5
Scaffold project (Architecture)	Patrick	5
Implement UI (Functionality)	Quency	2
Restock the machine inventory (Functionality)	Alex	3
Generating Records (Functionality)	Patrick	3
Create Food Classes (Architecture)	Quency	8
Create ShoppingCart Classes (Architecture)	Ken	2
Create Transaction Classes (Architecture)	Oscar	2
Create Record Classes (Architecture)	Patrick	2

Create Staff Classes (Architecture)	Alex	2
Implement Staff specific UI (Functionality) (NEW*)	Alex	2
Create Inventory Class (Architecture) (NEW*)	Collaborative	2
Handling item quantity (Functionality) (NEW*)	Collaborative	2
		61

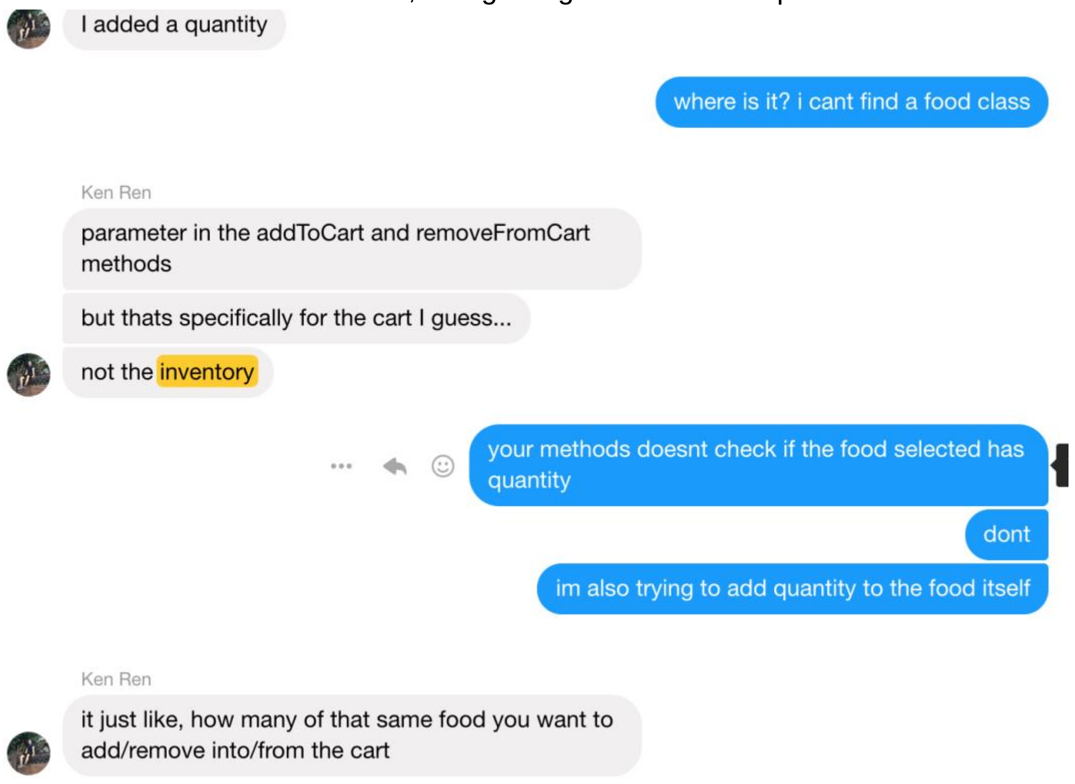
3.2. Sprint backlog

The Sprint backlog consists of features from the Product Backlog that are to be completed during the Sprint. It is visible to all stakeholders. The development team is responsible for selecting, modifying and implementing these features to meet the Sprint goal.

We recognise that the amount of story points allocated for this sprint outweighs the team's velocity. Initially, we thought it was feasible. However, most of the team were occupied with other assignments during this time which hindered progress.

One of the main changes made this sprint, is the inclusion of the Inventory class to handle changes to item quantity. The need for the additional class was first recognised by scrum master, Ken and developer Alex, as their tasks were directly related to item quantity.
Another change made this sprint is the inclusion of a Staff Interface, a separate interface to the Customer Interface. The Staff Interface has a separate set of operations, one of them allowing the staff to restock items.

Screenshot of the conversation, recognising a need for a separate class to handle quantity.



I added a quantity

where is it? i cant find a food class

Ken Ren

parameter in the addToCart and removeFromCart methods

but thats specifically for the cart I guess...

not the inventory

your methods doesnt check if the food selected has quantity

dont

im also trying to add quantity to the food itself

Ken Ren

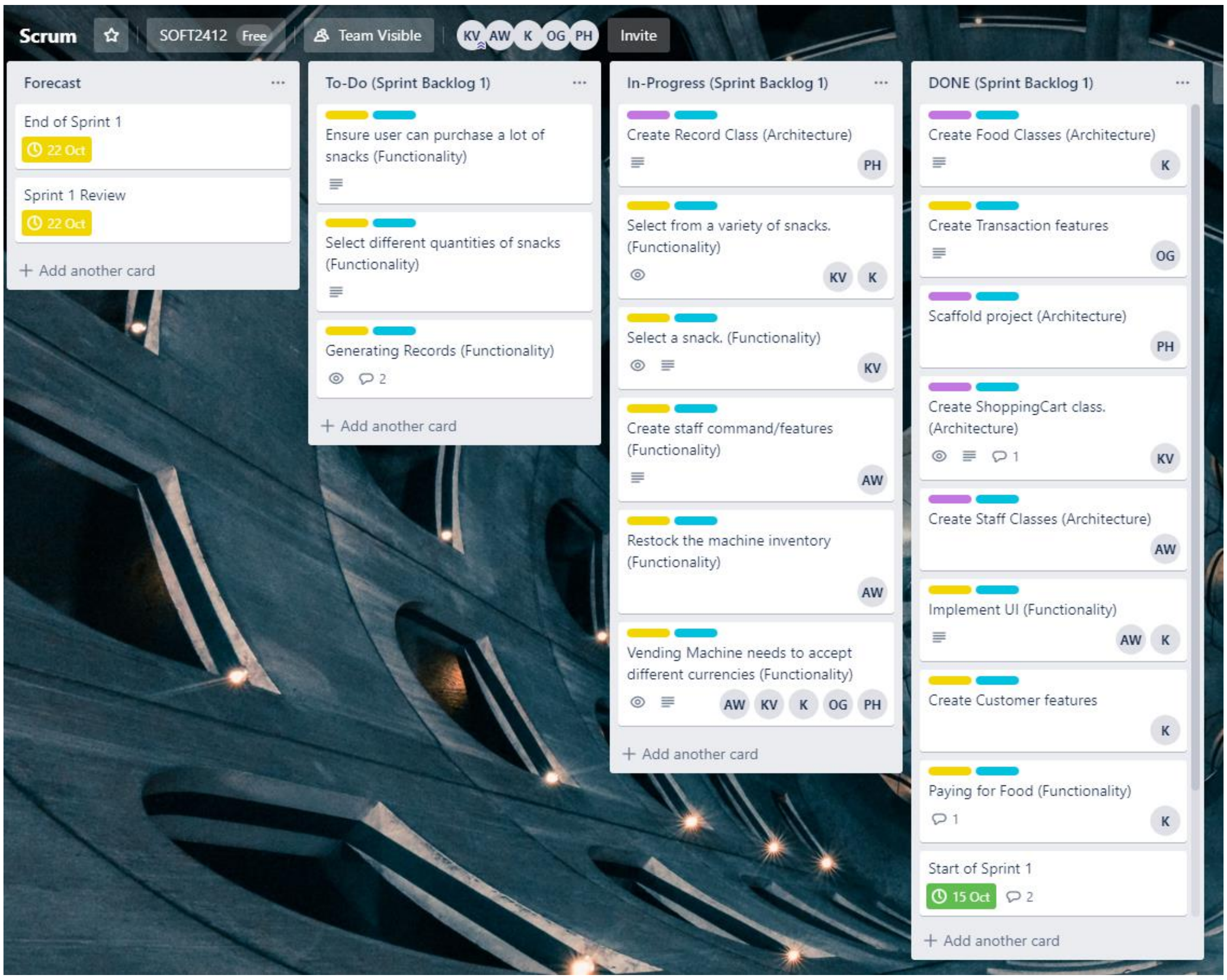
it just like, how many of that same food you want to add/remove into/from the cart

Features that meet the acceptance criteria during this sprint are marked with a ‘Y’ in the Completed column. On the other hand, those marked with a ‘N’ are referred to as features that are not “Done”. The selection and payment features are marked as incomplete as it does not satisfy all the conditions in the acceptance criteria. As of now, users are allowed to select more than the specified quantity and the item stock does not adjust accordingly when quantity is selected.

Sprint Backlog	Assigned	Story Points	Completed
Scaffold project (Architecture)	Patrick	5	Y
Create Food Classes (Architecture)	Quency	8	Y
Create ShoppingCart Classes (Architecture)	Ken	2	Y
Create Transaction Classes (Architecture)	Oscar	2	Y
Create Record Classes (Architecture)	Patrick	2	N
Create Staff Classes (Architecture)	Alex	2	Y
Implement UI (Functionality)	Quency	2	Y
Select a snack (Functionality)	Ken	3	Y
Select from a variety of snacks (Functionality)	Ken	3	Y
Select different quantities of snacks (Functionality)	Quency	5	N
Remove different quantities of snacks (Functionality)	Quency	2	N
Paying for Food (Functionality)	Oscar	3	N
Payment with different currencies (Functionality)	Collaborative	5	Y
Generating Records (Functionality)	Patrick	3	N
Implement Staff specific UI (Functionality) (NEW*)	Alex	2	Y
Restock the machine inventory (Functionality)	Alex	3	Y
Create Inventory Classes (Architecture) (NEW*)	Collaborative	2	Y
Handling item quantity (Functionality) (NEW*)	Collaborative	2	Y
		56	41

3.3. Trello Board

Example screenshot of the Trello Board towards the end of the Sprint.



Scrum ☆ SOFT2412 Free · Team Visible · KV AW K OG PH · Invite

Forecast
- End of Sprint 1 — 22 Oct
- Sprint 1 Review — 22 Oct
- + Add another card

To-Do (Sprint Backlog 1)
- Ensure user can purchase a lot of snacks (Functionality)
- Select different quantities of snacks (Functionality)
- Generating Records (Functionality) — 2
- + Add another card

In-Progress (Sprint Backlog 1)
- Create Record Class (Architecture) — PH
- Select from a variety of snacks. (Functionality) — KV K
- Select a snack. (Functionality) — KV
- Create staff command/features (Functionality) — AW
- Restock the machine inventory (Functionality) — AW
- Vending Machine needs to accept different currencies (Functionality) — AW KV K OG PH
- + Add another card

DONE (Sprint Backlog 1)
- Create Food Classes (Architecture) — K
- Create Transaction features — OG
- Scaffold project (Architecture) — PH
- Create ShoppingCart class. (Architecture) — 1 — KV
- Create Staff Classes (Architecture) — AW
- Implement UI (Functionality) — AW K
- Create Customer features — K
- Paying for Food (Functionality) — 1 — K
- Start of Sprint 1 — 15 Oct — 2
- + Add another card

After the overall goals were defined, we created a Trello board to break down the tasks. Each task were broken down and meant to be completed within a few hours of continuous work. If the tasks took any longer, they may have needed to be broken down even further.

Labels were added to make things clearer. A breakdown goes as follows:
- **Yellow:** functionality based tasks.
- **Light Blue:** Coding Tasks
- **Pink:** Design based Tasks
- **Orange:** Testing based Tasks
- **Purple:** Architecture/structure building Tasks.

We created a “Forecast”, “To-Do”, “In-Progress” and “DONE” lists. Which are meant to work as followed:
- **Forecast**: Simple due dates/reminders that aren’t actual tasks, so they don’t hold any points.
- **TO-DO:** Weighted tasks with points that need to be finished within the sprint.
- **In-Progress:** Tasks that have been assigned and work has been started, all cards that are in this list need to have 1 or member assigned to it..
- **DONE:** Completed tasks.

4. Scrum Artifacts

4.1. User Stories

The user stories were compiled during the sprint planning.

4.1.1. Select a snack

User Story	Acceptance Criteria
1. As a customer 2. So that I can consume the snack I want 3. I want to select the exact snack I desire	Conditions of satisfaction: - User can select the desired snack User: Customer Status: Purchase Parameters: Food ID (Integer) Return: Food object

4.1.2. Select different quantities of snacks

User Story	Acceptance Criteria
1. As a customer 2. So that I can consume the amount of the snack I desire 3. I want to be able to select the quantity I desire	Conditions of satisfaction: - User can select the quantity of snacks - Snack Quantity reduces User: Customer Status: Purchase Parameters: Quantity (Integer) Return: Quantity

4.1.3. Select from a variety of snacks

User Story	Acceptance Criteria
1. As a customer 2. So that I have a variety of choices 3. I want to purchase from four categories: drinks, chocolates, chips and lollies a. Drinks (water, soft drink, juice) b. Chocolates (M&M, Bounty, Mars, Sneakers) c. Chips (original, chicken, BBQ,and sweet chillies) d. Lollies (sour worms, jellybeans, little bears, part mix)	Conditions of satisfaction: - Different snack types displayed - User can select different snacks User: Customer Status: Purchase Parameters: Food ID and quantity Return: Add Food to Cart

4.1.4. Ensure user can purchase a lot of snacks

User Story	Acceptance Criteria
1. As a customer 2. So that I can satisfy my snack cravings 3. I want to purchase up to 10 items of each snack	Conditions of satisfaction: - Snack quantities displayed - Each snack quantity has maximum of 10 - User can select up to 10 for each snack User: Customer Status: Purchase Parameters: Food ID and quantity Return: Add Food to Cart with 10 quantity

4.1.5. Ensure user can remove selected snacks

User Story	Acceptance Criteria
1. As a customer 2. So that I can revert my mistakes when selecting snacks 3. I want to remove selected snacks	Conditions of satisfaction: - User can remove selected snacks - Snack Quantity reduces User: Customer Status: Purchase Parameters: Food ID and quantity Return: Remove Food from Cart

4.1.6. Restock the machine inventory

User Story	Acceptance Criteria
1. As a staff 2. So that I can fill the machine up to max capacity 3. Using a fill command, I should be able to see an updated list of the product stock with the date/time displayed.	Conditions of satisfaction: - Snack quantities displayed - Each snack quantity has maximum of 10 - User can select up to 10 for each snack - Date/time displayed User: Staff Status: Refill Parameters: None Return: All Food has max quantity of 10

4.1.7. Paying for Food

User Story	Acceptance Criteria
1. As a customer, 2. So that I can pay for snacks correctly, 3. I want to be able to pay for snacks with coins and notes up to $20 and receive change.	Conditions of satisfaction: - User can enter coins and notes of up to 20 - User can pay for the snacks User: Customer Status: Payment Parameters: Number of each denomination Return: Cart successfully checked out

4.1.8. Generating Records

User Story	Acceptance Criteria
1. As a staff member, 2. So that I can receive information on vending machine use, 3. I want to be able to create a report on daily transactions, stock, or cancelled transactions at any time.	Conditions of satisfaction: - List daily transactions - List stock - List cancelled transactions User: Staff Status: Create Record Parameters: None Return: List all transactions/stock

4.1.9. Paying with different Currency (Additional Requirement)

User Story	Acceptance Criteria
1. As a customer, 2. So that I can pay for snacks with different currency rates. 3. I want to be able to pay for snacks with any conversion rates, using any amount (i.e. notes and coins denomination not required).	Conditions of satisfaction: - User can select a currency - User can pay for the snacks User: Customer Status: Payment Parameters: Currency (String), Number of each denomination Return: Cart successfully checked out

4.1.10. Customer UI

User Story	Acceptance Criteria
1. As a user, 2. So that I can use the vending machine, 3. I want a user interface that is easy to understand, and can perform the necessary actions: ● Print all snacks, their categories, costs and quantities ● Select or remove multiple items and its quantities ● Pay using coins(10c, 20c, 50c, $1, $2) / notes($5, $10, $20) ● Quit program	Conditions of satisfaction: - Customer can select and run the desired action User: Customer Status: Customer UI Parameters: Command number (integer) Return: Run the selected command

4.1.11. Staff UI

User Story	Acceptance Criteria
1. As a staff, 2. So that I can use the vending machine, 3. I want a user interface that is easy to understand, and can perform the necessary actions: ● Fill items to max capacity and display date / time ● Generate records: ○ Daily transactions ○ Available stock ○ Cancelled transactions ● Quit program	Conditions of satisfaction: - Staff can select and run the desired action User: Staff Status: Staff UI Parameters: Command number (integer) Return: Run the selected command

4.2. Daily Scrum

Daily Scrum Meeting 1 (**Sprint Planning Meeting)** **Date**: 14th October 2019

Attendance: Patrick, Quency (Kunxi), Ken, Oscar, Alex
Venue: Carslaw cafe area

Sprint Duration: 1 Week.

Sprint Planning Duration: 2 hours.

Agenda (prepared by Scrum Master):
- Part 1:
  - Review Product backlog and Identify sprint goals.
- Part 2:
  - Sprint Backlog planning
  - Initial task Selection by members.
  - Code structure planning
  - Select user stories and identify tasks.

Content:
- Members decided which tasks they wanted to work on.
- Brief discussion on code structure and how to proceed from here.

Each briefly answer:
- Patrick:
  - What I did yesterday: Compiled first rendition of the product backlog.
  - What I will do today: Start and finish building the project scaffold.
  - Do I see any obstacles: None at the moment.
- Kunxi:
  - What I did yesterday: Reviewed the product backlog.
  - What I will do today: Start work on the Food class scaffolding, and if I finish it with time to spare, I will work on the functionality behind making payments for food.
  - Do I see any obstacles: None at the moment.
- Ken
  - What I did yesterday: Reviewed the product backlog. Began planning what goes into the Sprint backlog.
  - What I will do today: Start work on the ShoppingCart class and organise the project board.
  - Do I see any obstacles: The team members handling the Food class might interpret it differently from how I would, so there may be a need to reimplement some features in the shopping cart class to get things working properly.
- Oscar
  - What I did yesterday: Reviewed the product backlog.
  - What I will do today: Start work on the Transaction class.
  - Do I see any obstacles: None at the moment.
- Alex:
  - What I did yesterday: Reviewed the product backlog.
  - What I will do today: Start work on the Staff class. And if time permits, will work on the staff commands/functionality.

    - Do I see any obstacles: My laptop has been sent for repairs, so it might delay my work progress.

Issues highlighted & recommended solutions by scrum master :
- First day, so no issues detected for now.

Daily Scrum Meeting 2

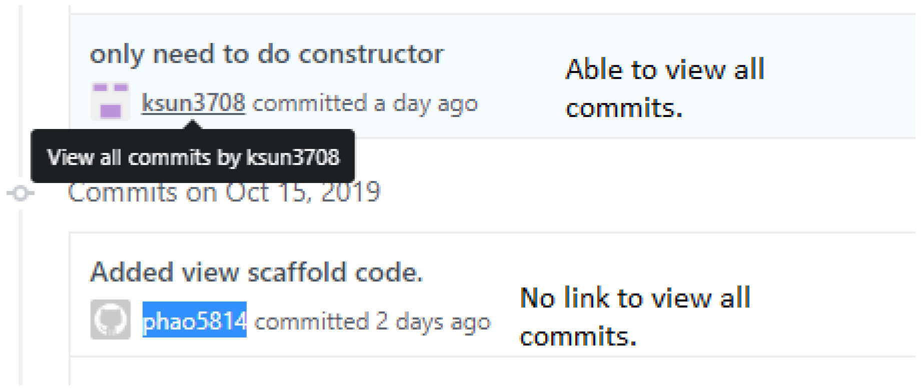
Date: 15th October 2019

Attendance: Patrick, Quency (Kunxi), Ken, Oscar, Alex
Venue: Online

Content:
- See chat log summary below.

Each briefly answer:
- Patrick:
    - What I did yesterday: Worked on the project scaffold.
    - What I will do today: Create the records class, continue work on the scaffold.
    - Do I see any obstacles: None at the moment.
- Kunxi:
    - What I did yesterday: Build all relevant Food classes, created a basic customer class, and implemented a basic draft command line UI.
    - What I will do today: Continue work on customer features. Look starting the currency conversion.
    - Do I see any obstacles: Because everyone is working independently on separate classes, might be problems when trying to combine.
- Ken
    - What I did yesterday: Completed the ShoppingCart class
    - What I will do today: Review the shopping cart class with member who did the Food class, work on food selection functionality.
    - Do I see any obstacles: None at the moment.
- Oscar
    - What I did yesterday: Added to the scaffold.
    - What I will do today: Continue work on Transaction class.
    - Do I see any obstacles: None at the moment.
- Alex:
    - What I did yesterday: Helped with work on artefacts.
    - What I will do today: If I am able to retrieve my laptop, I will begin work on the tasks I decided to work on from the previous scrum.
    - Do I see any obstacles: If my laptop is not repaired, work will be further delayed.

Issues highlighted & recommended solutions by scrum master :
- Noticed a team member’s commits not getting registered.



only need to do constructor

ksun3708 committed a day ago

View all commits by ksun3708

Able to view all commits.

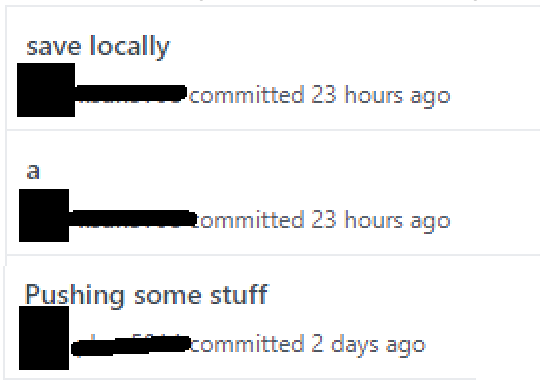
Commits on Oct 15, 2019

Added view scaffold code.

phao5814 committed 2 days ago

No link to view all commits.

-
    - Suggested: check login credentials of their git config, might not be using the right account.
- Commit messages tend to be too vague:



save locally

[redacted] committed 23 hours ago

a

[redacted] committed 23 hours ago

Pushing some stuff

[redacted] committed 2 days ago

-
    - Suggested: Add more specific sentences, and also suggested to make commits more frequently at smaller intervals.

Chat log summary:

Patrick Hao

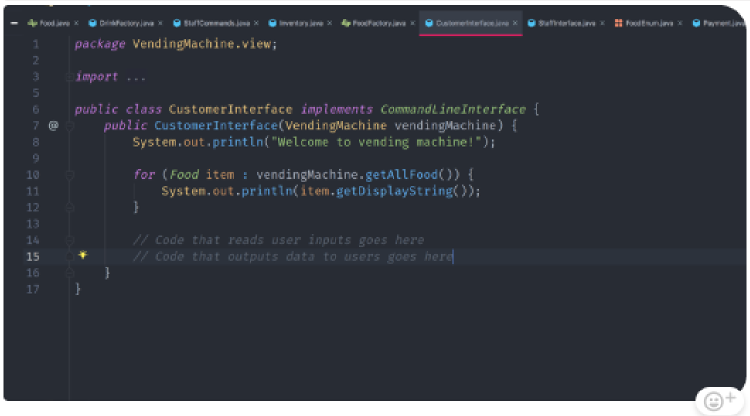
alright scaffold is ready

feel free to pull

create a scaffold branch locally first

and then run ken's command

So currently the plan should be that all print and scanning of customer input should go in the CustomerInterface class



```java
package VendingMachine.view;

import ...

public class CustomerInterface implements CommandLineInterface {
    public CustomerInterface(VendingMachine vendingMachine) {
        System.out.println("Welcome to vending machine!");

        for (Food item : vendingMachine.getAllFood()) {
            System.out.println(item.getDisplayString());
        }

        // Code that reads user inputs goes here
        // Code that outputs data to users goes here
    }
}
```

CustomerInterface class should only interact with the VendingMachine class in my opinion. // will add some more example methods alter

later*

Quency Sun

I just checked scaffold

Do we really need these food factory?

Can we only make one food factory

I like your design of inventory

Quency Sun

And I am confused about food config and vending machine confir

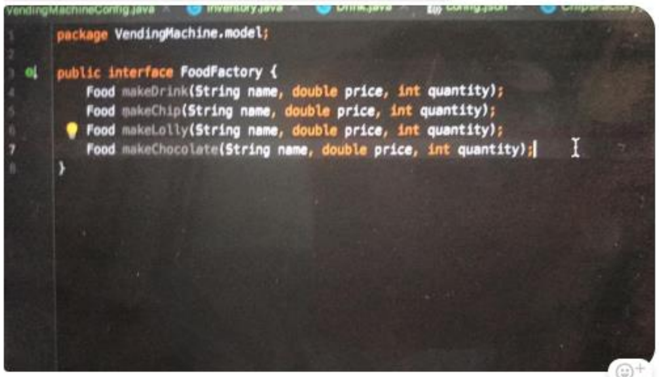
Patrick Hao

Right now we don't but I was thinking in the future the food items might have different parameters. Like drinks might have sizes (350mL vs one litre) and same with chip bags etc

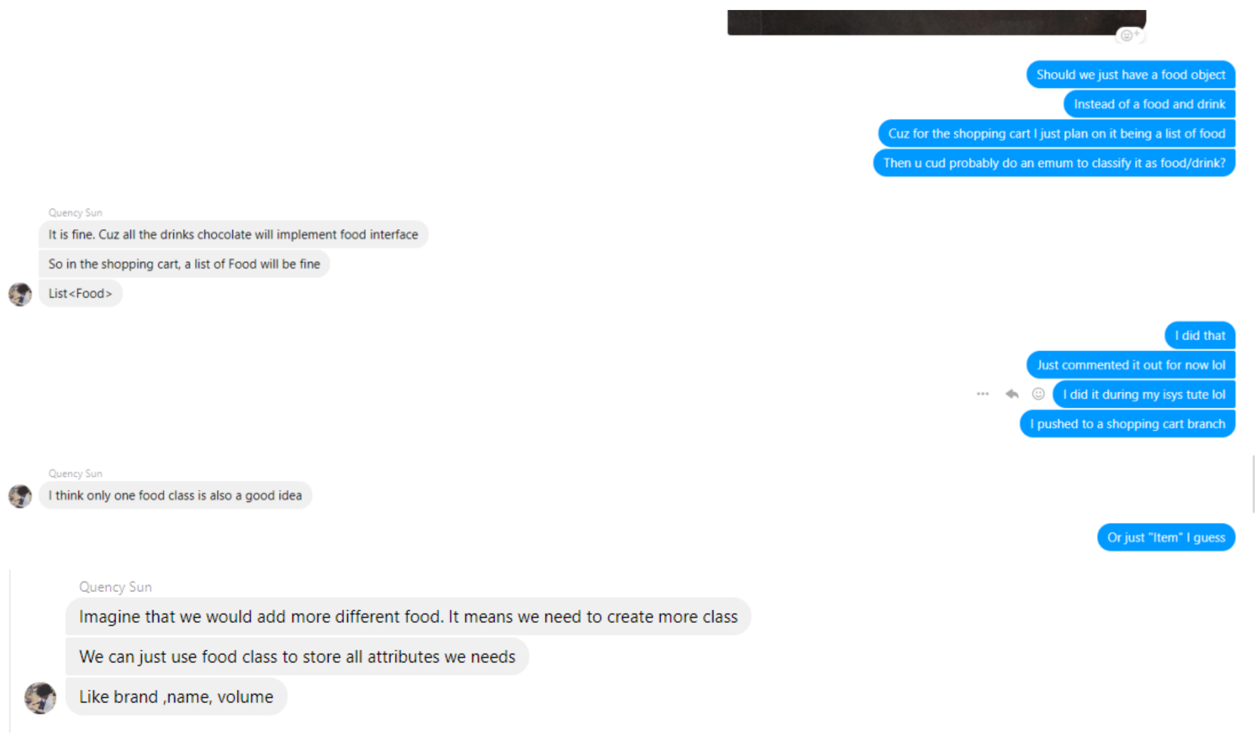
But I agree that right now the factories are pointless

Quency Sun

What I mean is we can have only one factory class



```java
package VendingMachine.model;

public interface FoodFactory {
    Food makeDrink(String name, double price, int quantity);
    Food makeChip(String name, double price, int quantity);
    Food makeLolly(String name, double price, int quantity);
    Food makeChocolate(String name, double price, int quantity);
}
```



Should we just have a food object

Instead of a food and drink

Cuz for the shopping cart I just plan on it being a list of food

Then u cud probably do an emum to classify it as food/drink?

Quency Sun

It is fine. Cuz all the drinks chocolate will implement food interface

So in the shopping cart, a list of Food will be fine

List<Food>

I did that

Just commented it out for now lol

I did it during my isys tute lol

I pushed to a shopping cart branch

Quency Sun

I think only one food class is also a good idea

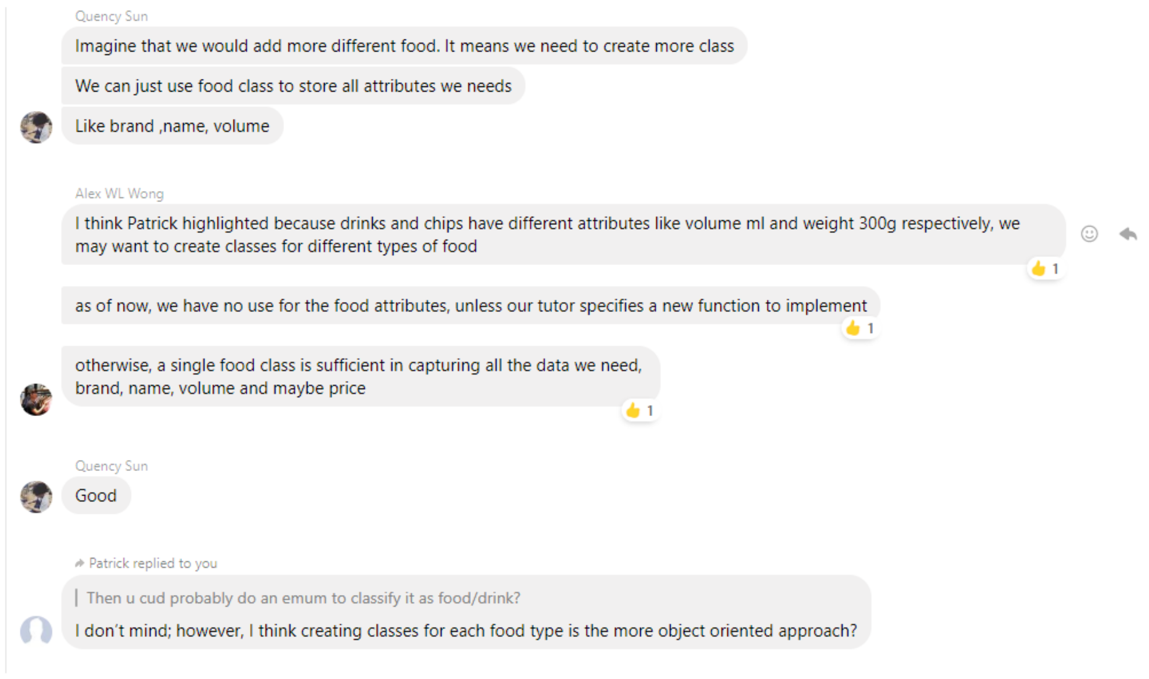
Or just "Item" I guess

Quency Sun

Imagine that we would add more different food. It means we need to create more class

We can just use food class to store all attributes we needs

Like brand ,name, volume



Quency Sun

Imagine that we would add more different food. It means we need to create more class

We can just use food class to store all attributes we needs

Like brand ,name, volume

Alex WL Wong

I think Patrick highlighted because drinks and chips have different attributes like volume ml and weight 300g respectively, we may want to create classes for different types of food

👍 1

as of now, we have no use for the food attributes, unless our tutor specifies a new function to implement

👍 1

otherwise, a single food class is sufficient in capturing all the data we need, brand, name, volume and maybe price

👍 1

Quency Sun

Good

Patrick replied to you

| Then u cud probably do an emum to classify it as food/drink?

I don't mind; however, I think creating classes for each food type is the more object oriented approach?

Daily Scrum Meeting 3

Date: 16th October 2019

Attendance: Patrick, Quency (Kunxi), Ken, Oscar, Alex
Venue: Carslaw cafe area

Agenda (prepared by Scrum Master):
- Ask Patrick about the scaffold
- Clarify if anything else needs to be done with ShoppingCart class.
- Discuss on the implementation of the selection features.

- Discuss how transaction class should work.

Content:
- Scaffold incomplete few things still need to be added. Team agreed to work on non-dependent things first.
- ShoppingCart seems good to go will add more Quency will add more payment features with this now finished.
- Team agreed that we should remove transaction class and Oscar will add the transaction features into the payment class.

Each briefly answer:
- Patrick:
  - What I did yesterday: Worked a bit more on the scaffold.
  - What I will do today: Putting the Record class on hold to work on Inventory class.
  - Do I see any obstacles: None at the moment.
- Kunxi:
  - What I did yesterday: Build all relevant Food classes, created a basic customer class, and implemented a basic draft command line UI.
  - What I will do today: Continue work on the customer features, and command line UI. Maybe work on payment methods.
  - Do I see any obstacles: None at the moment.
- Ken
  - What I did yesterday: Finished work on the ShoppingCart class, continue on artefacts and managing the trello board.
  - What I will do today: Start work on the snack selection feature for customers. Continue on documentation, managing trello board, and scrum master tasks.
  - Do I see any obstacles: None at the moment.
- Oscar
  - What I did yesterday: Reviewed the tasks, and some of the code.
  - What I will do today: Finally start on the transaction features.
  - Do I see any obstacles: I might not be able to work on this task as I have an assignment due soon.
- Alex:
  - What I did yesterday: Could not do my task as my laptop was away for repairs.
  - What I will do today: Review code, add features and work on the Staff related features.
  - Do I see any obstacles: None at the moment.

Issues highlighted & recommended solutions by scrum master :
- The team still sometimes make commits in big chunks.
  - The team should try to make smaller and more frequent commits as good practice.
- The team is a bit too focused on design, which over complicates things at the moment.
  - As this is an agile development project, the team should try to focus more on getting the functionality working first before taking into account design for this sprint at least. As it is causing some unnecessary confusion with team members unfamiliar with design patterns.

Daily Scrum Meeting 4 **Date**: 17th October 2019

Attendance: Patrick, Quency (Kunxi), Ken, Oscar, Alex
Venue: Online

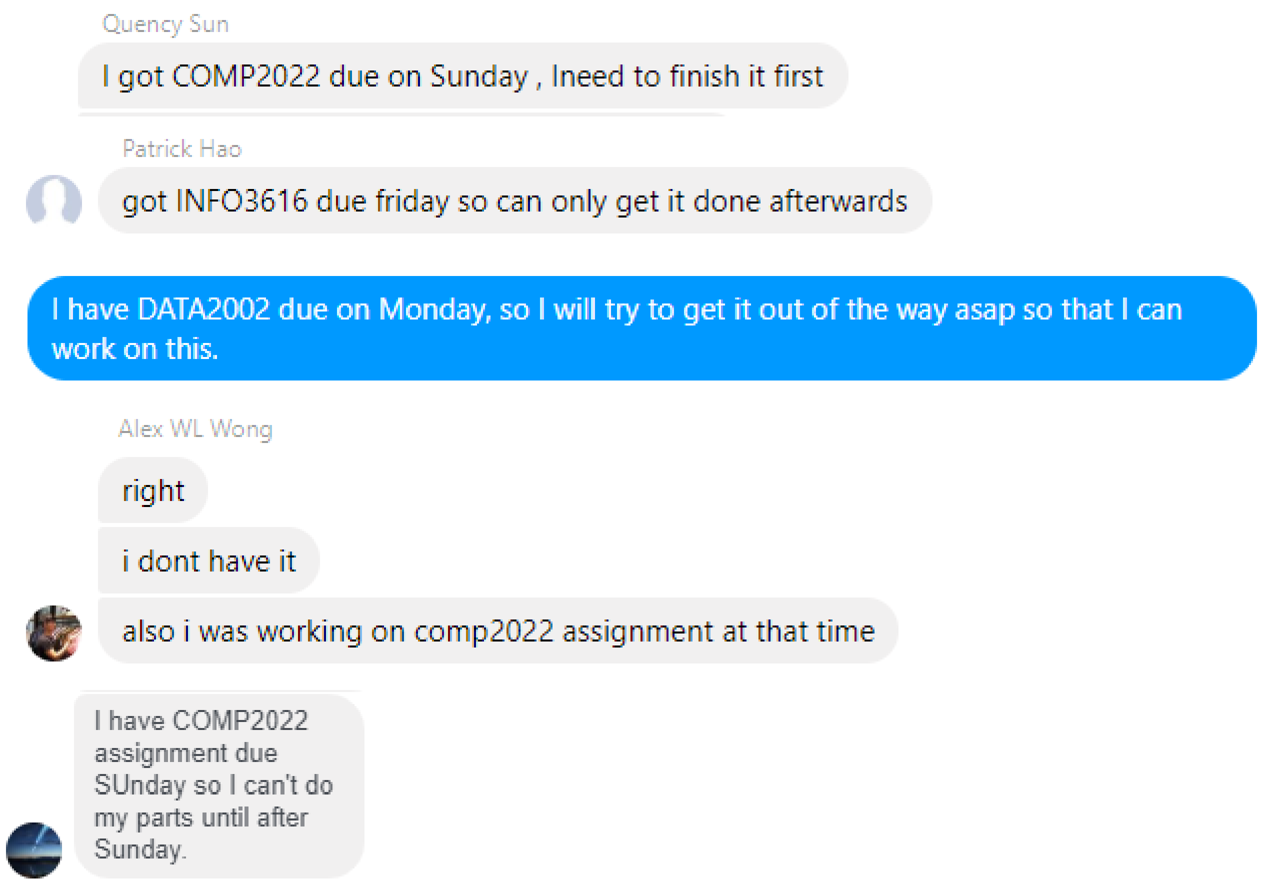
Each briefly answer:

- Patrick:
    - What I did yesterday: None assignment due today.
    - What I will do today: I have other assignments due in the coming few days, so work may be further delayed.
    - Do I see any obstacles: Cause of other commitments work may be delayed, and time may be an issue.
- Kunxi:
    - What I did yesterday: Finished the command line UI, continued with customer features. Created basic payment features.
    - What I will do today: Continue work on the customer features.
    - Do I see any obstacles: I have an assignment due soon, so may need to delay work until that gets done.
- Ken
    - What I did yesterday: Had to make changes on the ShoppingCart class again to get it to work properly with other features added, continue on artefacts and managing the trello board. Selection feature was delayed due to other tasks being delayed.
    - What I will do today: Continue on some work on the artefacts, reviewing the code and trello board management.
    - Do I see any obstacles: May need to delay work because I have an assignment due soon.
- Oscar
    - What I did yesterday: Reviewed some of the code, but didn’t work on it because I had another assignment due soon and needed to start work on it.
    - What I will do today: Can’t work on it today, I need to get the other assignment done as it’s due soon.
    - Do I see any obstacles: I might not be able to work on this task as I have an assignment due soon.
- Alex:
    - What I did yesterday: Could not do my task as my laptop was away for repairs.
    - What I will do today: Review code to understand it better and add features.
    - Do I see any obstacles: I have 2 assignments due over the weekend, so work may be severely delayed.

Issues highlighted & recommended solutions by scrum master :

- None in terms of team organization and operation. However, all members have assignments due, which will severely hinder progress.

Chat log summary:


Quency Sun

I got COMP2022 due on Sunday , Ineed to finish it first

Patrick Hao

got INFO3616 due friday so can only get it done afterwards

I have DATA2002 due on Monday, so I will try to get it out of the way asap so that I can work on this.

Alex WL Wong

right

i dont have it

also i was working on comp2022 assignment at that time

I have COMP2022 assignment due SUnday so I can't do my parts until after Sunday.

Daily Scrum Meeting 5-7 **Date**:18th-20 October 2019

Meeting(s) postponed due to all members being busy with other commitments that required more immediate attention.

Daily Scrum Meeting 8 (Final) **Date**: 21 October 2019

Attendance: Patrick, Quency (Kunxi), Ken, Oscar, Alex
Venue: Online

Content:
See chat log summary

Each briefly answer:

- Patrick:
  - What I did yesterday: Long absence from the code, so had to review it.
  - What I will do today: Finish all outstanding tasks.
  - Do I see any obstacles: Last day of the sprint, so time is a massive problem.
- Kunxi:
  - What I did yesterday: Didn’t work on the code because of other work.

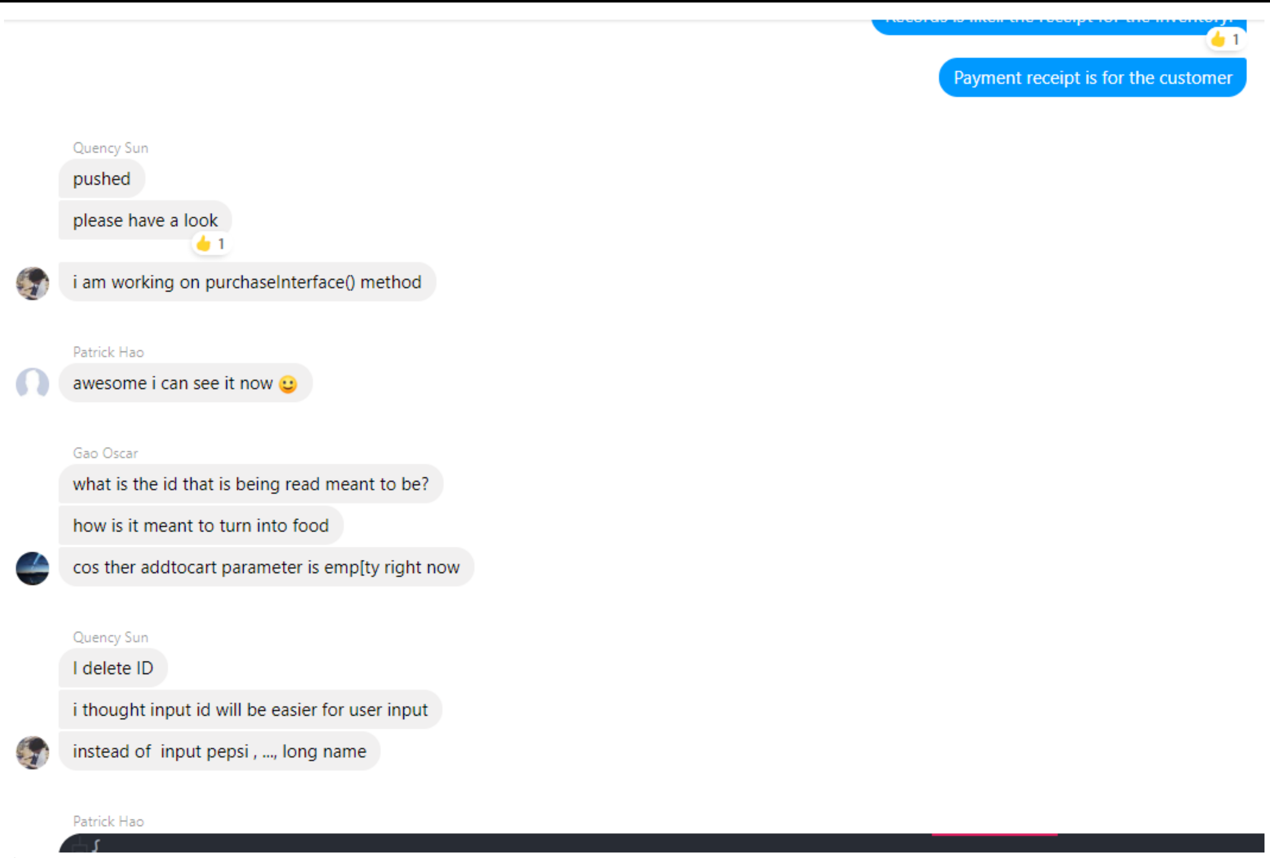
    - What I will do today: Work on all the remaining tasks.
    - Do I see any obstacles: There isn't enough time.

- Ken
    - What I did yesterday: Reviewed the code.
    - What I will do today: Continue on some work on the artefacts, do whatever outstanding tasks that remain.
    - Do I see any obstacles: Time is a major problem.
- Oscar
    - What I did yesterday: Had to look at the code again, because I haven't touched it in awhile.
    - What I will do today: I will finish all my outstanding tasks.
    - Do I see any obstacles: Time.
- Alex:
    - What I did yesterday: Reviewed the code.
    - What I will do today: Work on all outstanding tasks.
    - Do I see any obstacles: Might not be able to finish all the tasks, due to the lack of time.

Issues highlighted & recommended solutions by scrum master :

- Due to other commitments team members had to postpone their tasks. However, delays could have been reduced.
    - Solution suggested: As this is the final day of the sprint, this is a suggestion mostly for the next sprint to set stricter, but more concise milestones and fixed due dates for each member's tasks.
- Members began working on just 1 branch (scaffold), instead of separate features on separate branches.

Chat log summary:



Payment receipt is for the customer

Quency Sun

pushed

please have a look

i am working on purchaseInterface() method

Patrick Hao

awesome i can see it now 🙂

Gao Oscar

what is the id that is being read meant to be?

how is it meant to turn into food

cos ther addtocart parameter is emp[ty right now

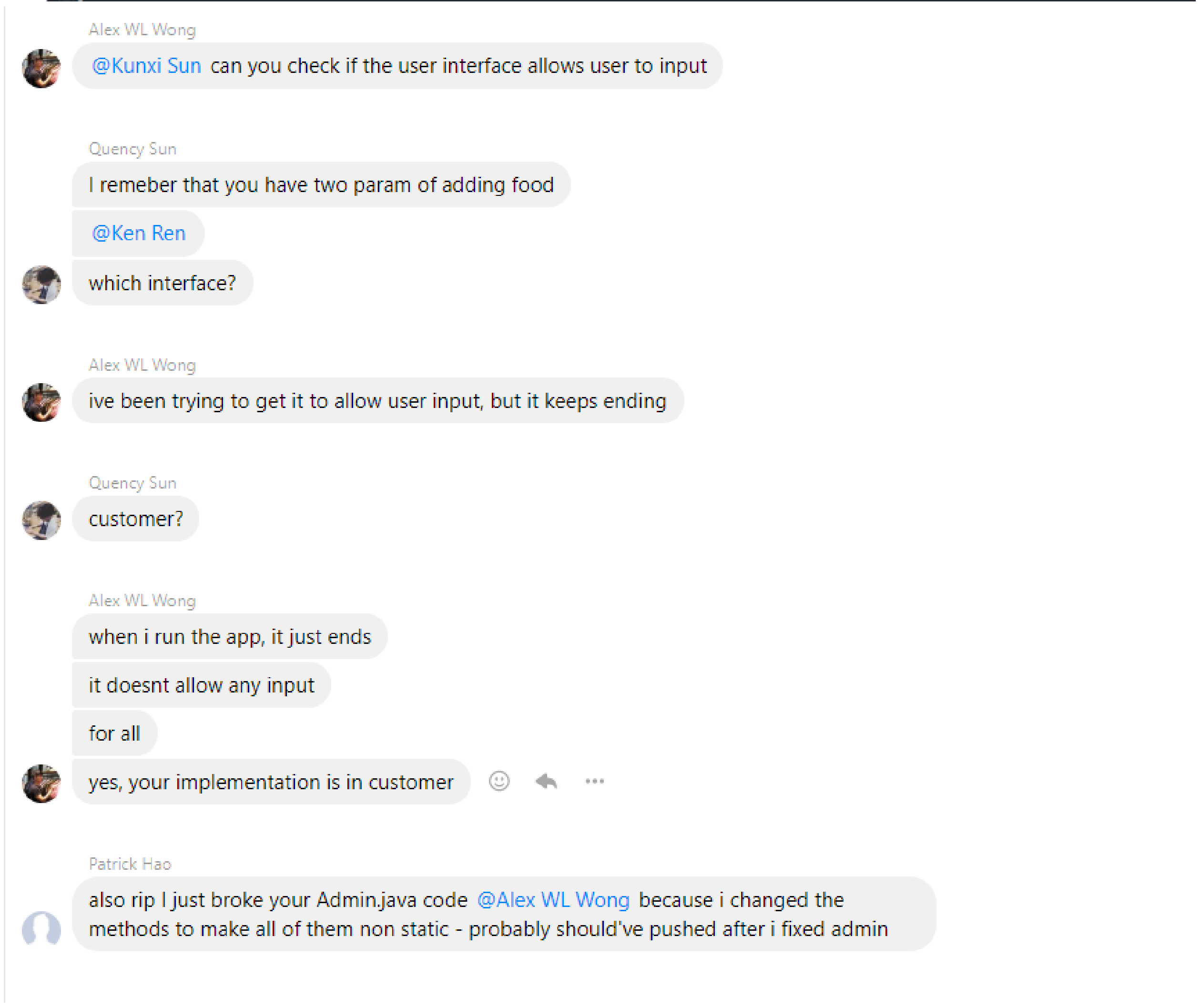
Quency Sun

I delete ID

i thought input id will be easier for user input

instead of input pepsi , ..., long name

Patrick Hao



Alex WL Wong

@Kunxi Sun can you check if the user interface allows user to input

Quency Sun

I remeber that you have two param of adding food

@Ken Ren

which interface?

Alex WL Wong

ive been trying to get it to allow user input, but it keeps ending

Quency Sun

customer?

Alex WL Wong

when i run the app, it just ends

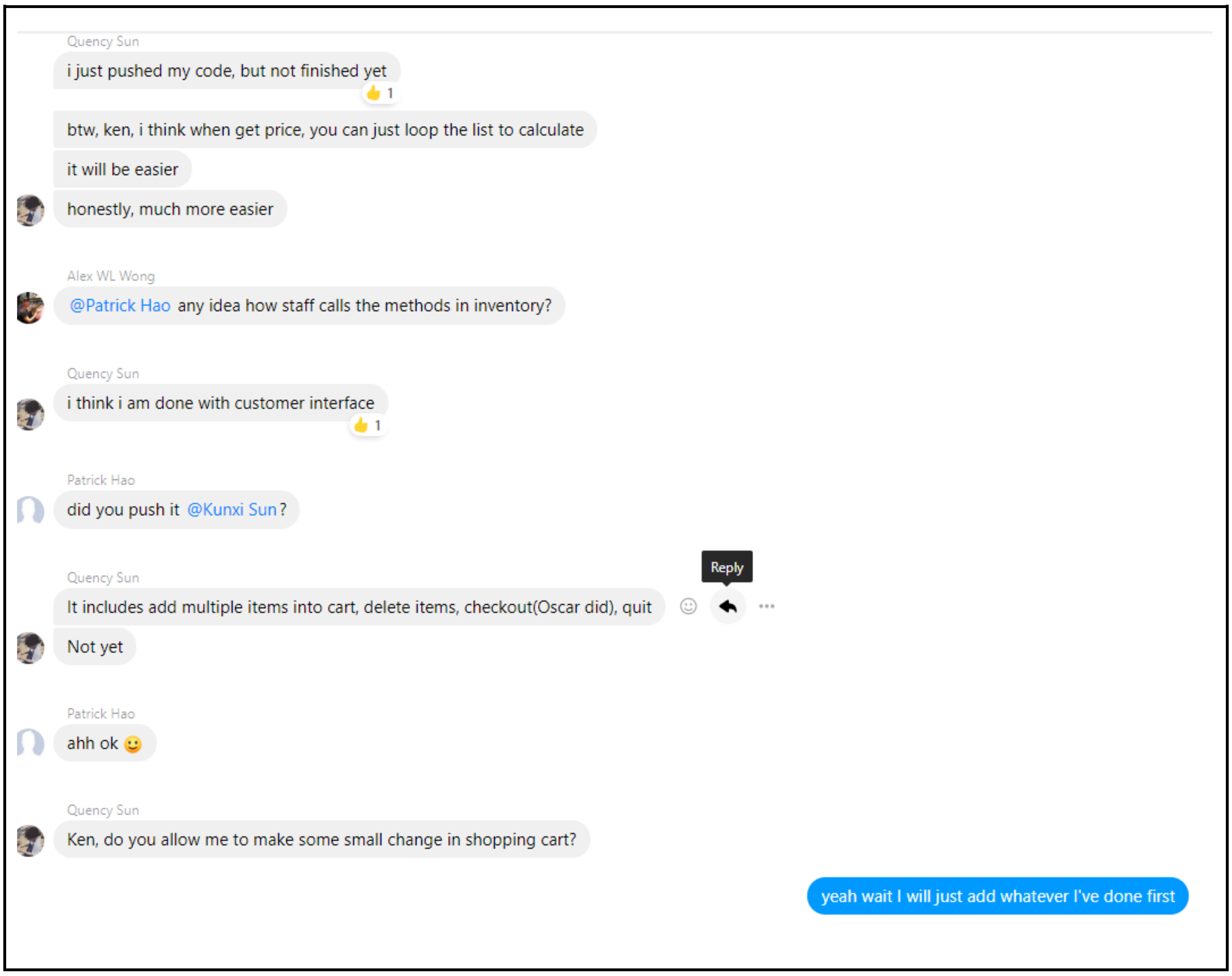
it doesnt allow any input

for all

yes, your implementation is in customer

Patrick Hao

also rip I just broke your Admin.java code @Alex WL Wong because i changed the methods to make all of them non static - probably should've pushed after i fixed admin


Quency Sun

i just pushed my code, but not finished yet

👍 1

btw, ken, i think when get price, you can just loop the list to calculate

it will be easier

honestly, much more easier

Alex WL Wong

@Patrick Hao any idea how staff calls the methods in inventory?

Quency Sun

i think i am done with customer interface

👍 1

Patrick Hao

did you push it @Kunxi Sun?

Quency Sun

It includes add multiple items into cart, delete items, checkout(Oscar did), quit

Not yet

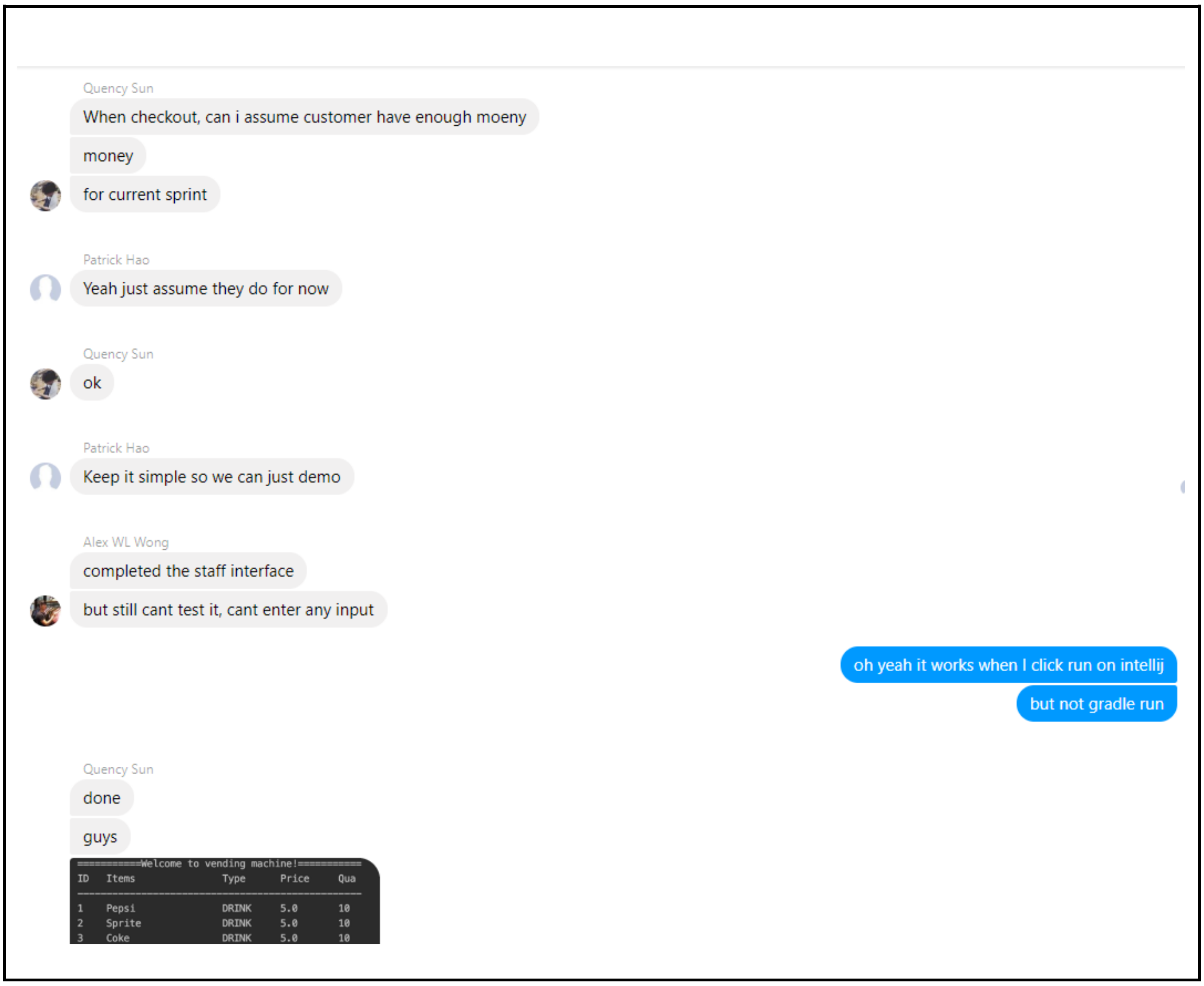
Patrick Hao

ahh ok 🙂

Quency Sun

Ken, do you allow me to make some small change in shopping cart?

yeah wait I will just add whatever I've done first



Quency Sun

When checkout, can i assume customer have enough moeny

money

for current sprint

Patrick Hao

Yeah just assume they do for now

Quency Sun

ok

Patrick Hao

Keep it simple so we can just demo

Alex WL Wong

completed the staff interface

but still cant test it, cant enter any input

oh yeah it works when I click run on intellij

but not gradle run

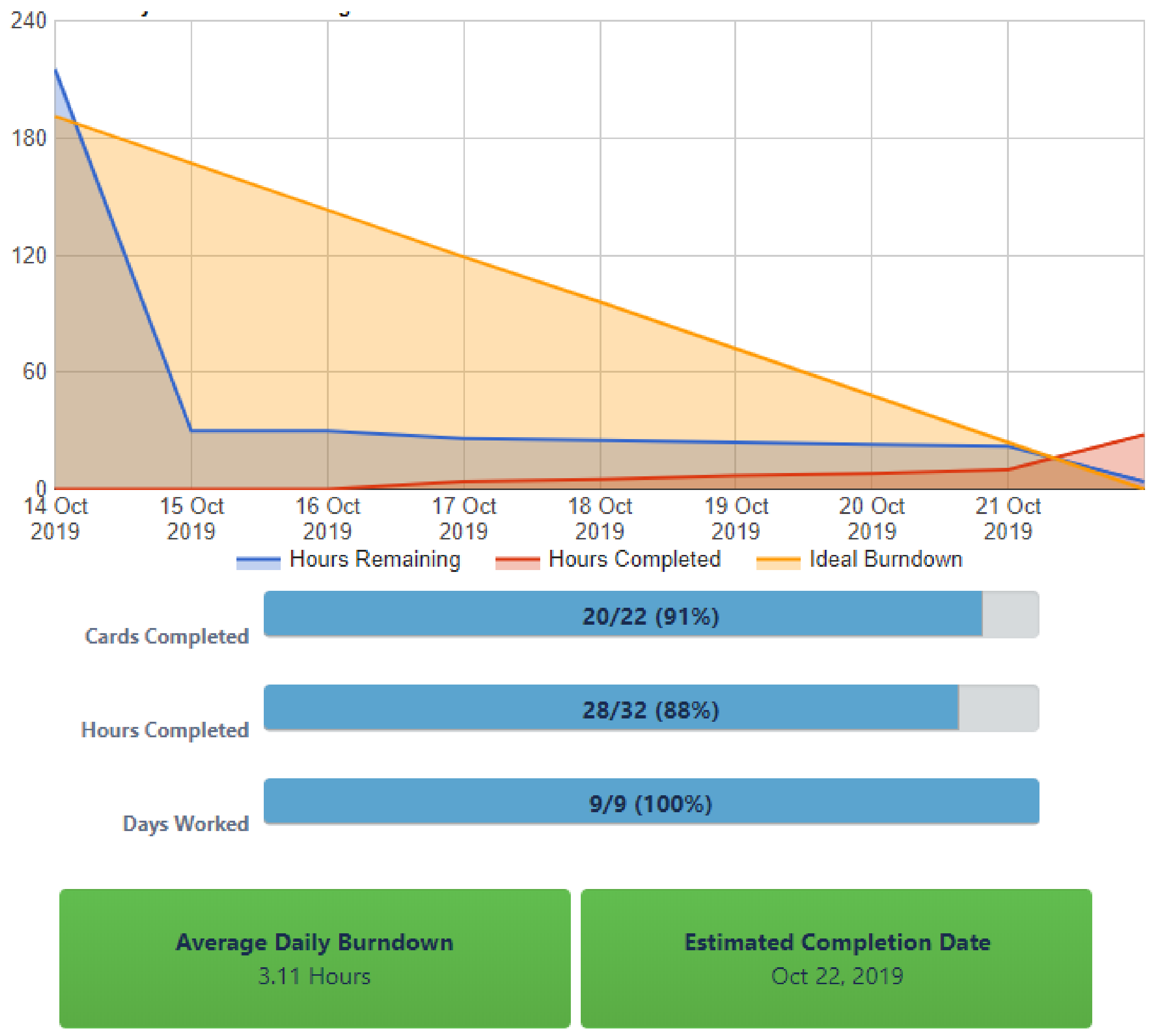
Quency Sun

done

guys

```
===========Welcome to vending machine!===========
ID   Items                Type      Price     Qua
-------------------------------------------------
1    Pepsi                DRINK     5.0       10
2    Sprite               DRINK     5.0       10
3    Coke                 DRINK     5.0       10
```

4.3. Burndown Chart



Cards Completed	20/22 (91%)
Hours Completed	28/32 (88%)
Days Worked	9/9 (100%)

Average Daily Burndown
3.11 Hours

Estimated Completion Date
Oct 22, 2019

Burndown chart, dated 22nd October 2019.

Note: The strange spike in hours completed towards the end was due to completed tasks being moved over to the “DONE” list a day later than they were actually finished.

5. Development Tools & Practices

5.1. Github

5.1.1. Initialisation

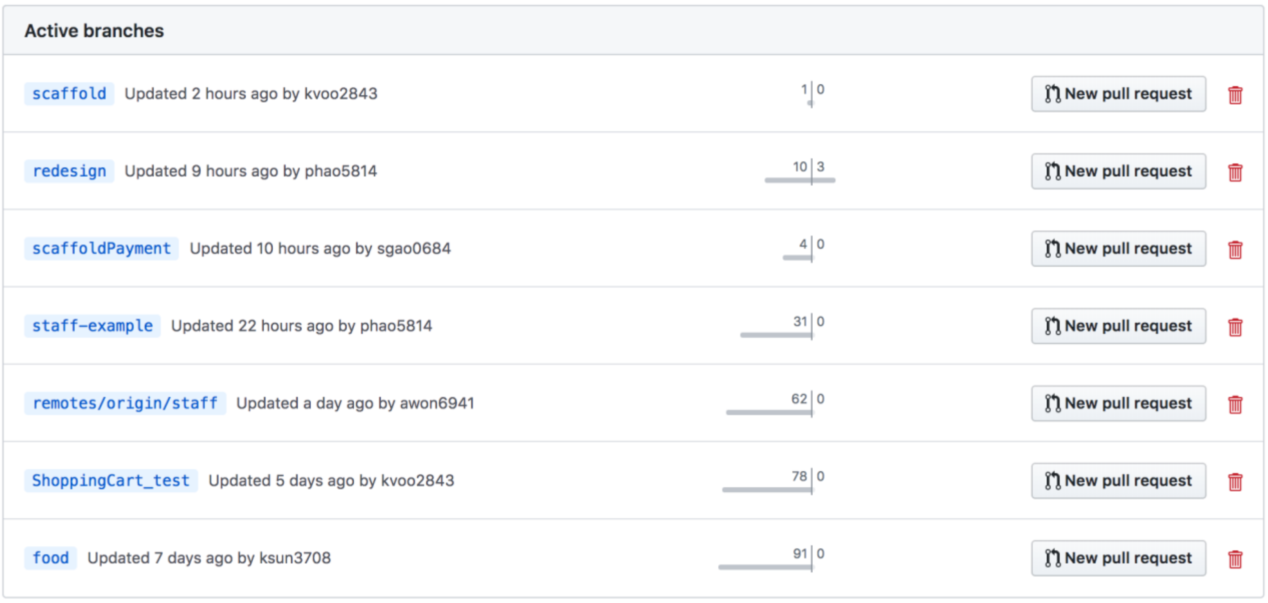
Our repository was initialised using the following standard sequence of git commands, which linked our master branch to the remote server’s “origin” repository.

Link to repository: https://github.sydney.edu.au/ksun3708/VendingMachine

```
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.sydney.edu.au/ksun3708/VendingMachine.git
git push -u origin master
```

5.1.2. Branches

The following branches were used for development. They broadly represent the features that we developed during our sprint.



Active branches

scaffold	Updated 2 hours ago by kvoo2843	1 | 0	New pull request
redesign	Updated 9 hours ago by phao5814	10 | 3	New pull request
scaffoldPayment	Updated 10 hours ago by sgao0684	4 | 0	New pull request
staff-example	Updated 22 hours ago by phao5814	31 | 0	New pull request
remotes/origin/staff	Updated a day ago by awon6941	62 | 0	New pull request
ShoppingCart_test	Updated 5 days ago by kvoo2843	78 | 0	New pull request
food	Updated 7 days ago by ksun3708	91 | 0	New pull request

5.2. Gradle

5.2.1. Explanation of build.gradle file

The following is a breakdown and explanation of our build.gradle file.

5.2.2. Plugins

Plugins are a set of tasks that are used by a project. Below are the plugins that are used by our project.

- **java:** Adds Java compilation, testing, and bundling capabilities to our project.
- **application:** Facilitates creating an executable JVM application. It makes it easy to start the application locally during development, and to package the application as a TAR and/or ZIP including operating system specific start scripts.

5.2.3. Repositories

The repository we are using to source our dependencies from is jcentre. Another popular repository is mavenCentral; however, both are similar in the breadth of packages they have available.

5.2.4. Dependencies

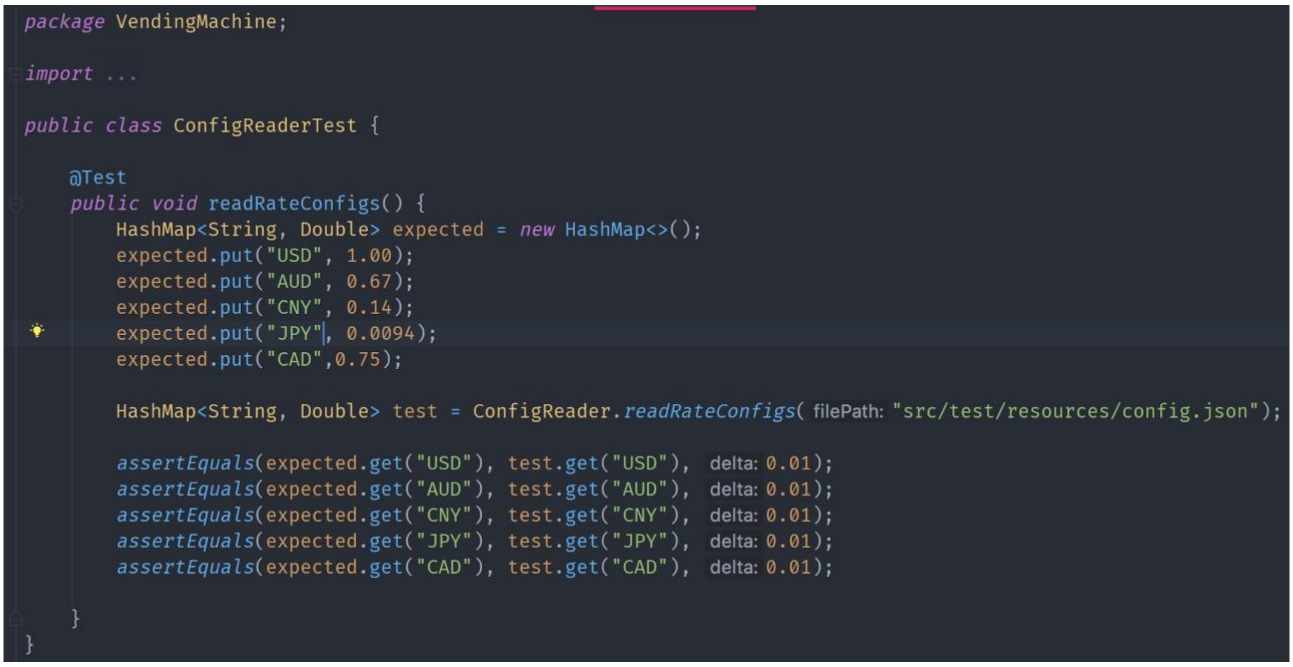
The following dependencies are used within our project.

- **guava:** A set of core libraries such as new collection types like multimap and multiset etc., which is used by the application.
- **junit4:** The testing library used by our project.
- **javatuples:** To allow us to use tuples within the application since Java does not have a tuple data type by default.
- **json-simple:** To allow us to parse JSON files, since we use a JSON file to load default exchange rates.

```
/*
 * This file was generated by the Gradle 'init' task.
 *
 * This generated file contains a sample Java project to get you started.
 * For more details take a look at the Java Quickstart chapter in the Gradle
 * User Manual available at
https://docs.gradle.org/5.5.1/userguide/tutorial_java_projects.html
 */

plugins {
    // Apply the java plugin to add support for Java
    id 'java'

    // Apply the application plugin to add support for building a CLI
application
    id 'application'

}

repositories {
    // Use jcenter for resolving dependencies.
    // You can declare any Maven/Ivy/file repository here.
    jcenter()
}

dependencies {
    // This dependency is used by the application.
    implementation 'com.google.guava:guava:27.1-jre'

    compile 'org.javatuples:javatuples:1.2'

    implementation 'com.googlecode.json-simple:json-simple:1.1.1'
    compile 'org.apache.commons:commons-lang3:3.9'

    //compile 'com.fasterxml.jackson.core:jackson-databind:2.0.1'

    // Use JUnit test framework
    testImplementation 'junit:junit:4.12'
}

application {
    // Define the main class for the application
    mainClassName = 'VendingMachine.App'
}

jacocoTestReport {
    reports {
        html.enabled = true
        csv.enabled = true
    }
}

test {
    test.finalizedBy jacocoTestReport // <- add this line
}
```

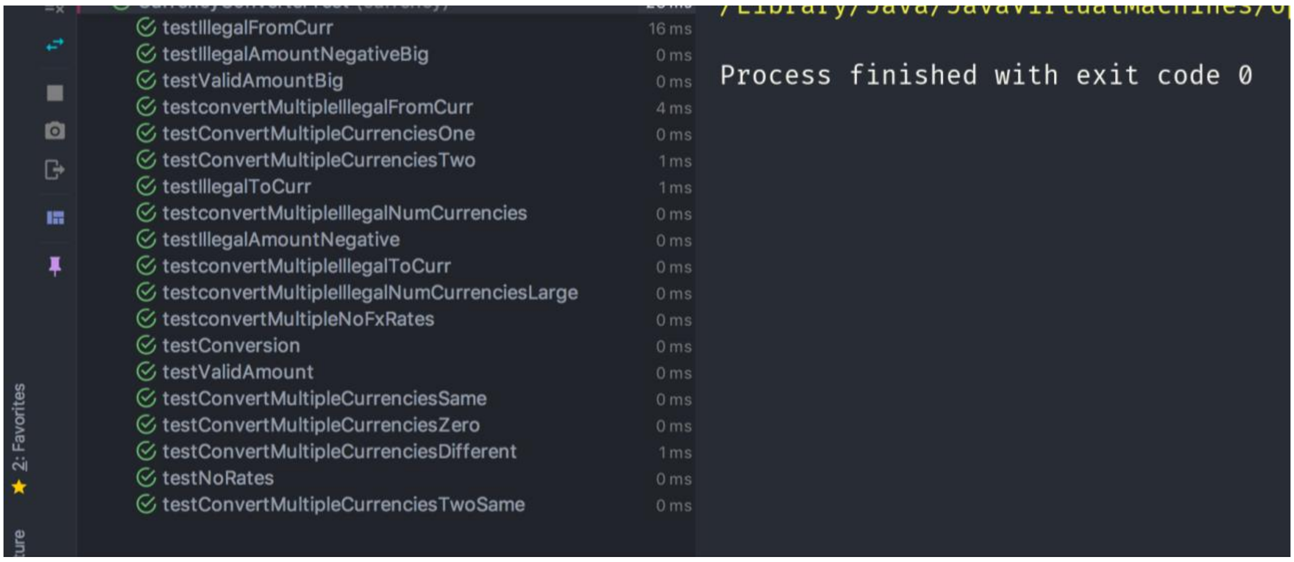
5.3. JUnit

We set up JUnit tests for our application (as shown below) to encourage developers to use a Test Driven Development philosophy.



```
package VendingMachine;

import ...

public class ConfigReaderTest {

    @Test
    public void readRateConfigs() {
        HashMap<String, Double> expected = new HashMap<>();
        expected.put("USD", 1.00);
        expected.put("AUD", 0.67);
        expected.put("CNY", 0.14);
        expected.put("JPY", 0.0094);
        expected.put("CAD",0.75);

        HashMap<String, Double> test = ConfigReader.readRateConfigs( filePath: "src/test/resources/config.json");

        assertEquals(expected.get("USD"), test.get("USD"),  delta: 0.01);
        assertEquals(expected.get("AUD"), test.get("AUD"),  delta: 0.01);
        assertEquals(expected.get("CNY"), test.get("CNY"),  delta: 0.01);
        assertEquals(expected.get("JPY"), test.get("JPY"),  delta: 0.01);
        assertEquals(expected.get("CAD"), test.get("CAD"),  delta: 0.01);

    }
}
```

The following outputs affirm to the user that all the test cases within this particular file have passed successfully.



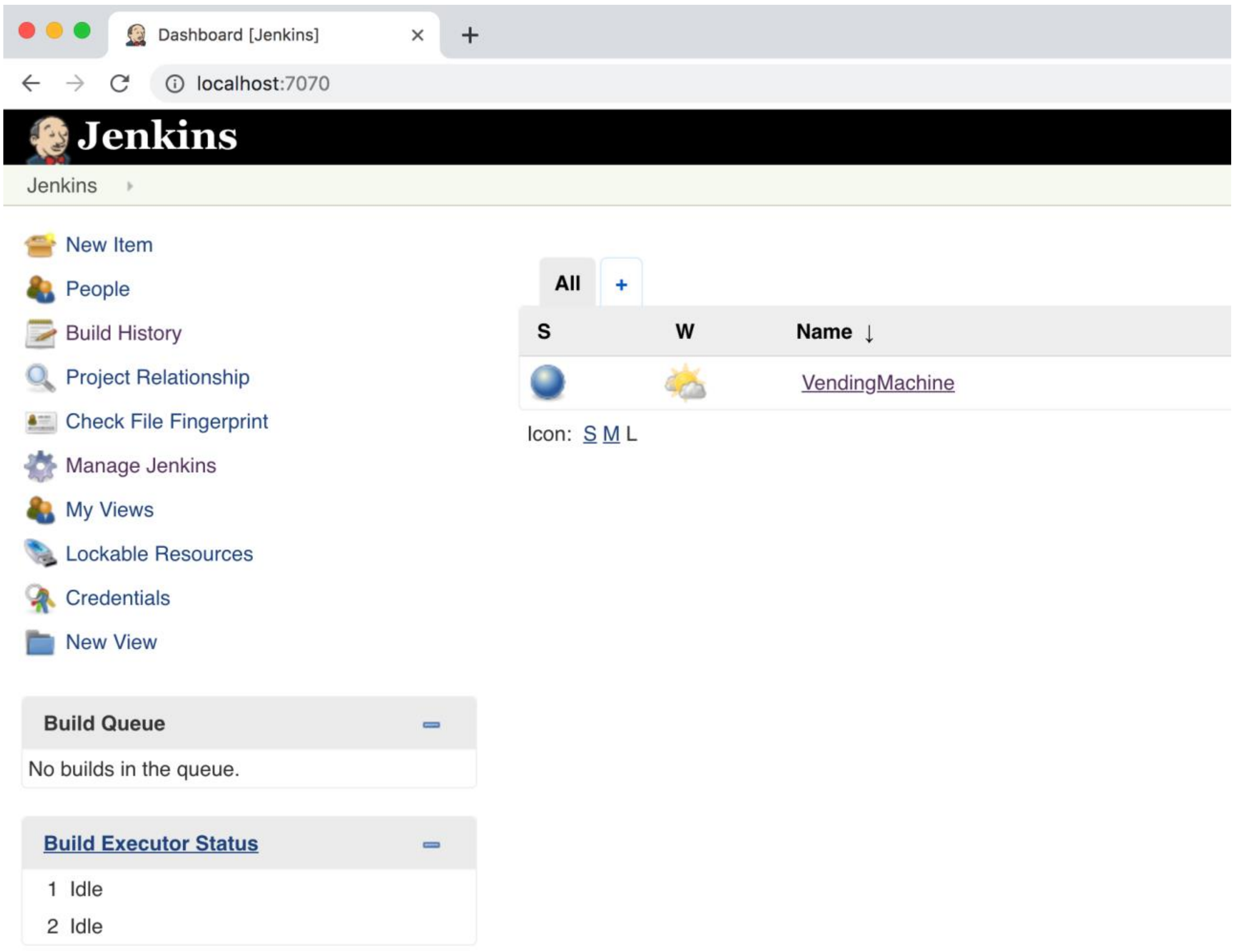
```
testIllegalFromCurr                               16 ms
testIllegalAmountNegativeBig                       0 ms
testValidAmountBig                                 0 ms
testconvertMultipleIllegalFromCurr                 4 ms
testConvertMultipleCurrenciesOne                   0 ms
testConvertMultipleCurrenciesTwo                   1 ms
testIllegalToCurr                                  1 ms
testconvertMultipleIllegalNumCurrencies            0 ms
testIllegalAmountNegative                          0 ms
testconvertMultipleIllegalToCurr                  0 ms
testconvertMultipleIllegalNumCurrenciesLarge       0 ms
testconvertMultipleNoFxRates                       0 ms
testConversion                                     0 ms
testValidAmount                                    0 ms
testConvertMultipleCurrenciesSame                  0 ms
testConvertMultipleCurrenciesZero                  0 ms
testConvertMultipleCurrenciesDifferent             1 ms
testNoRates                                        0 ms
testConvertMultipleCurrenciesTwoSame               0 ms
```

```
Process finished with exit code 0
```

5.4. Jenkins CI

We also set up Jenkins CI to automate the running of tests as well as the build process.

We created a local Jenkins server on http://localhost:7070 as shown in the below screenshot.



S	W	Name ↓
		VendingMachine

Icon: S M L

Build Queue

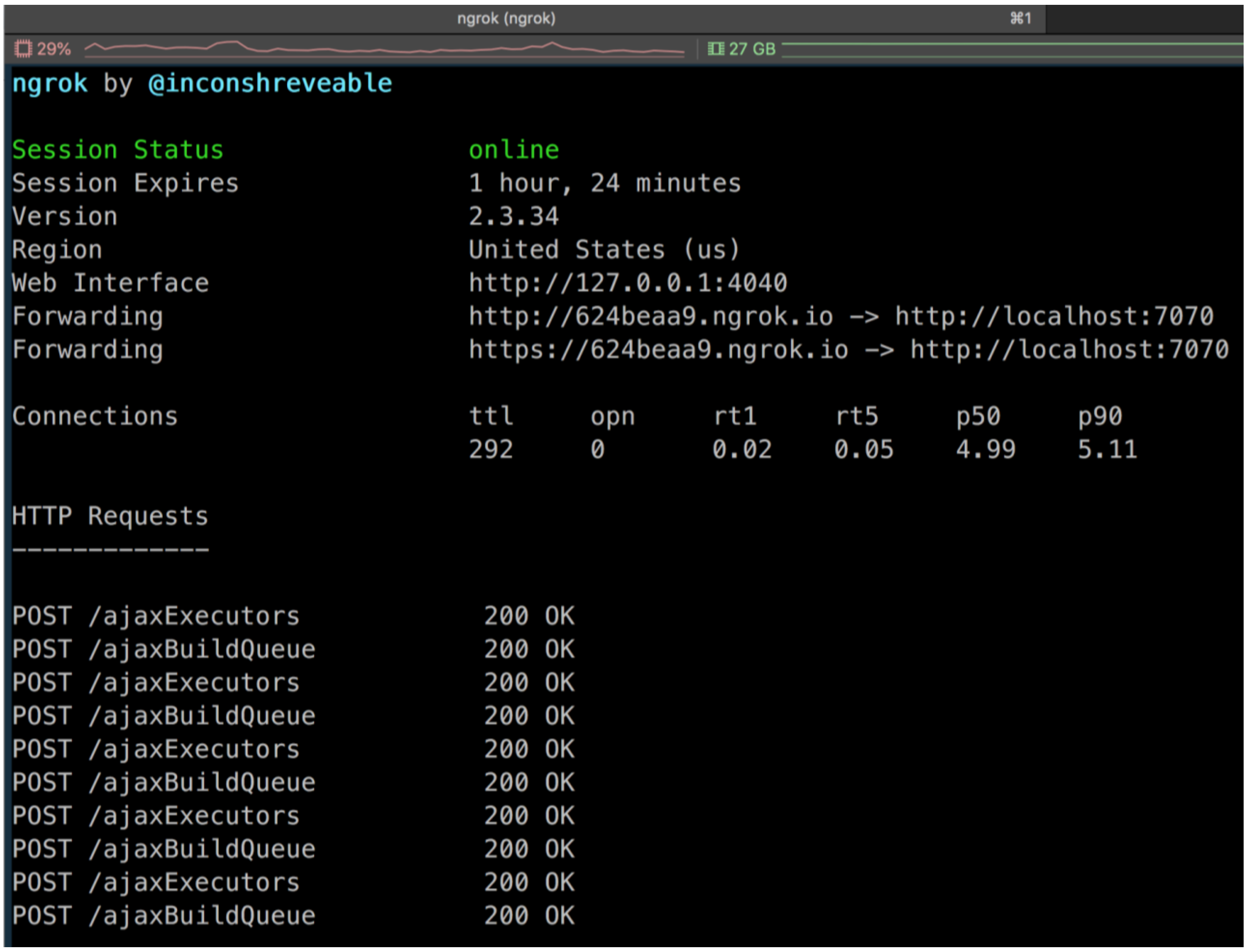
No builds in the queue.

Build Executor Status

1 Idle
2 Idle

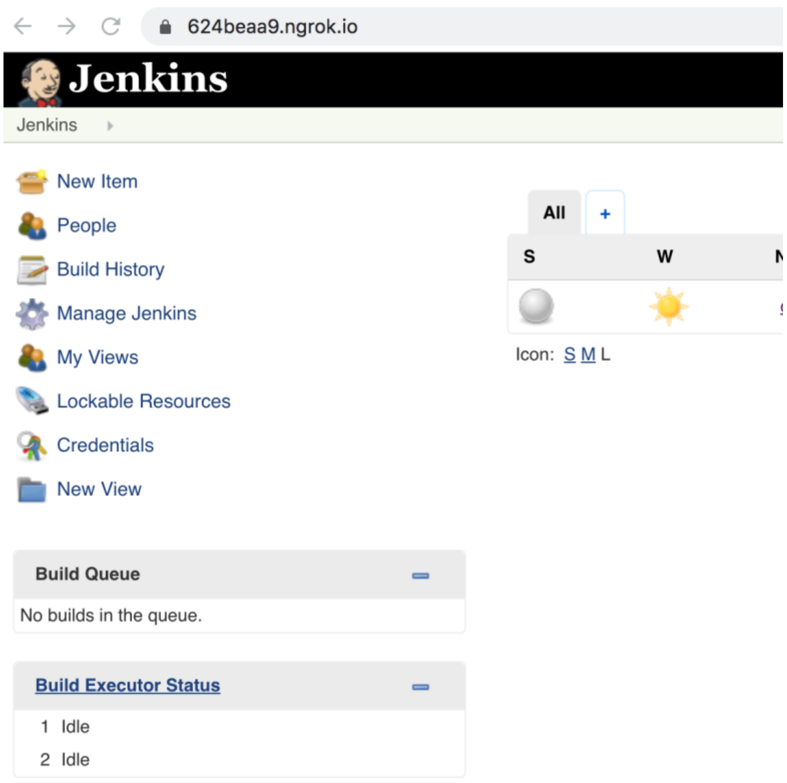
5.4.1. Expose Local Server to GitHub using ngrok

Below is a screenshot demonstrating that we're forwarding http://locahost:7070 to https://624beaa9.ngrok.io.



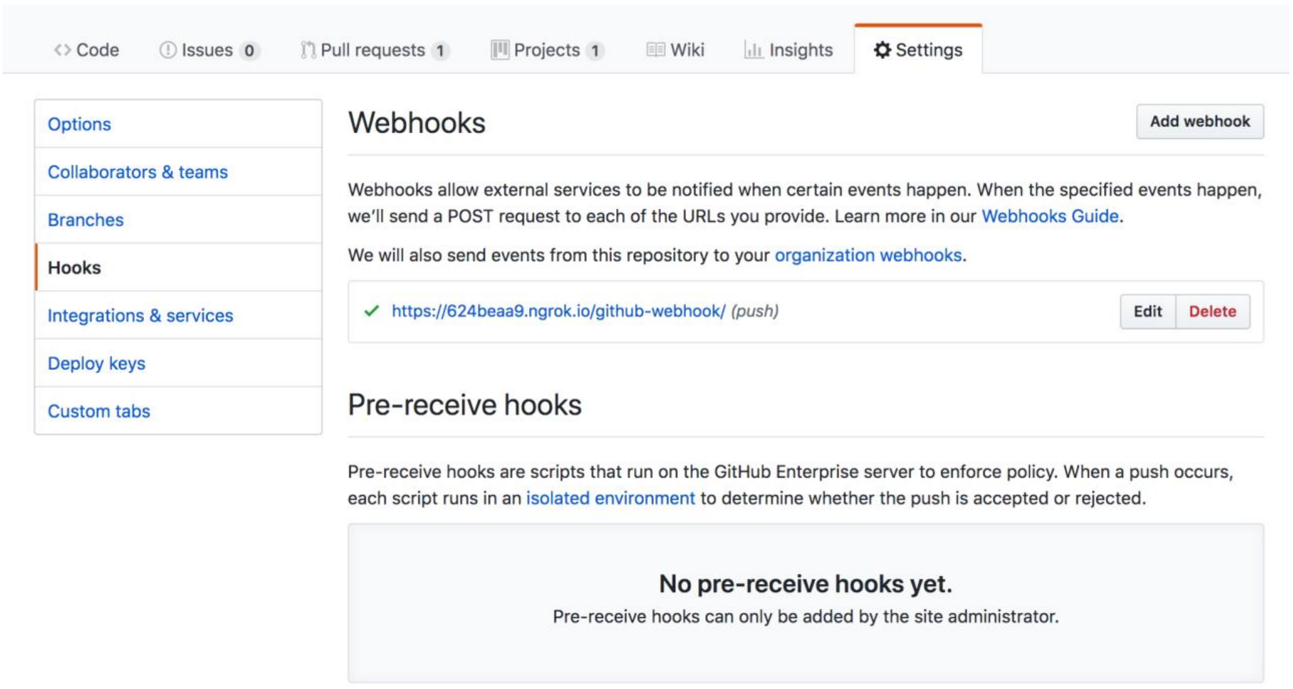
```
ngrok by @inconshreveable

Session Status                online
Session Expires               1 hour, 24 minutes
Version                       2.3.34
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://624beaa9.ngrok.io -> http://localhost:7070
Forwarding                    https://624beaa9.ngrok.io -> http://localhost:7070

Connections                   ttl     opn     rt1     rt5     p50     p90
                              292     0       0.02    0.05    4.99    5.11

HTTP Requests
-------------

POST /ajaxExecutors            200 OK
POST /ajaxBuildQueue           200 OK
POST /ajaxExecutors            200 OK
POST /ajaxBuildQueue           200 OK
POST /ajaxExecutors            200 OK
POST /ajaxBuildQueue           200 OK
POST /ajaxExecutors            200 OK
POST /ajaxBuildQueue           200 OK
POST /ajaxExecutors            200 OK
POST /ajaxBuildQueue           200 OK
```

By visiting https://624beaa9.ngrok.io, we land on our local Jenkins server.



5.4.2. Set up Webhook

Below is a screenshot demonstrating that we have successfully set up our GitHub webhook.



Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

We will also send events from this repository to your organization webhooks.

✓ https://624beaa9.ngrok.io/github-webhook/ *(push)*

Pre-receive hooks

Pre-receive hooks are scripts that run on the GitHub Enterprise server to enforce policy. When a push occurs, each script runs in an isolated environment to determine whether the push is accepted or rejected.

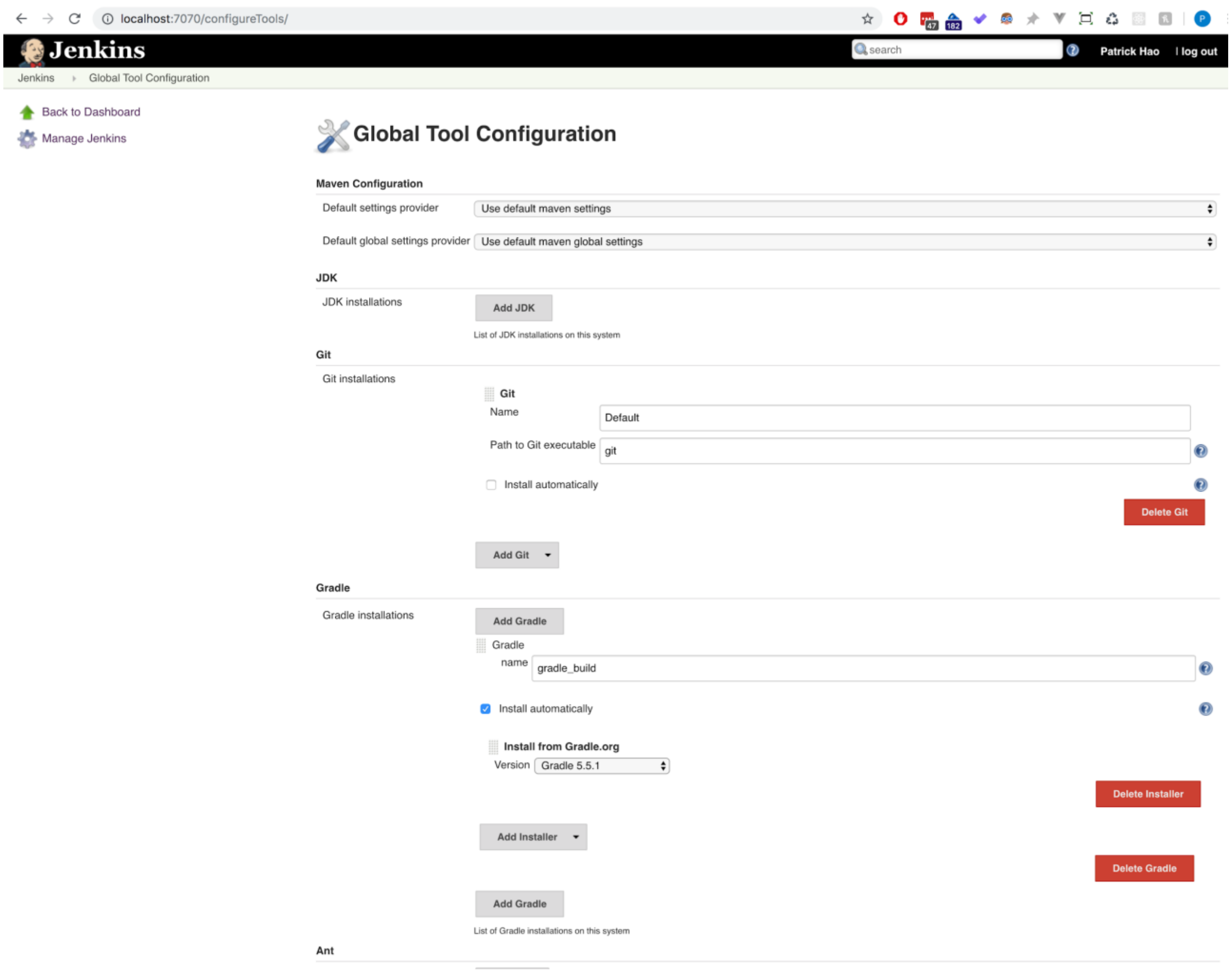
No pre-receive hooks yet.

Pre-receive hooks can only be added by the site administrator.

5.4.3. Set up Automated JUnit Tests

Automated JUnit tests were set up after we integrated Gradle into Jenkins. This was to ensure that after every push to our remote repository, we could automatically run a specific Gradle task like running the “test” task.

First we added Gradle to Jenkins as shown in the following screenshot.


localhost:7070/configureTools/

Jenkins › Global Tool Configuration

Patrick Hao | log out

Global Tool Configuration

Maven Configuration

Default settings provider: Use default maven settings

Default global settings provider: Use default maven global settings

JDK

JDK installations: Add JDK

List of JDK installations on this system

Git

Git installations

Git

Name: Default

Path to Git executable: git

Install automatically: ☐

Gradle

Gradle installations

Gradle

name: gradle_build

Install automatically: ☑

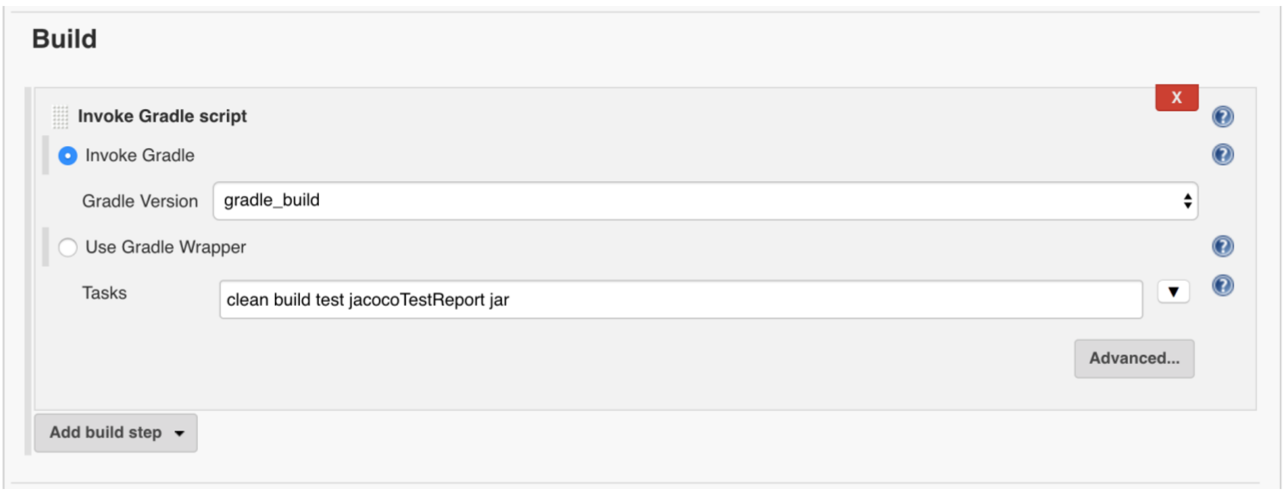
Install from Gradle.org

Version: Gradle 5.5.1

List of Gradle installations on this system

Ant

Then we went to the “Configure” tab of Jenkins for our project and scrolled down to “Build” as shown below. We selected “gradle_build”, which was the name of our gradle installation (as shown in the screenshot above) and in “Tasks” we entered “clean”, “build”, “test”, “jacocoTestReport” and “jar”.


Build

Invoke Gradle script

[✓ Invoke Gradle]

Gradle Version: gradle_build

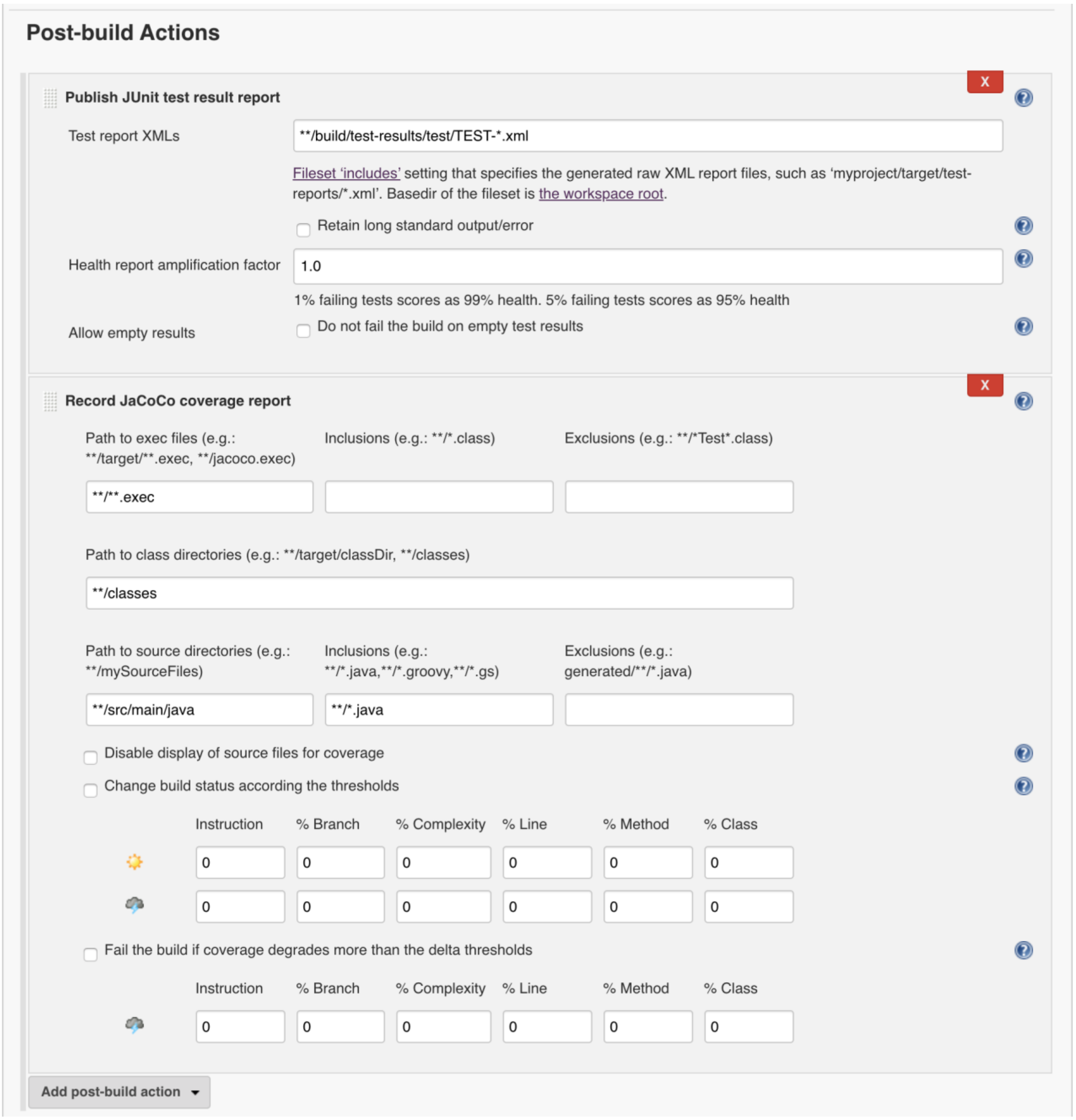
Use Gradle Wrapper

Tasks: clean build test jacocoTestReport jar

Essentially we are telling gradle that in the build stage, execute the following gradle command, which involves cleaning the build directory (clean), building the app (build), running JUnit tests (test), running code coverage tests (jacocoTestReport) and outputting a runnable jar file (jar).

```
gradle clean build test jacocoTestReport jar
```

Given we've told Jenkins to run JUnit tests and calculate code coverage in the above command, we've also added two post-build actions that would help visualise those results in Jenkins. Below are our settings for the "Publish JUnit test result report" and "Record JaCoCo coverage report" post-build actions.


Post-build Actions

Publish JUnit test result report

Test report XMLs: **/build/test-results/test/TEST-*.xml

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is the workspace root.

☐ Retain long standard output/error

Health report amplification factor: 1.0

1% failing tests scores as 99% health. 5% failing tests scores as 95% health

Allow empty results: ☐ Do not fail the build on empty test results

Record JaCoCo coverage report

Path to exec files (e.g.: **/target/**.exec, **/jacoco.exec)	Inclusions (e.g.: **/*.class)	Exclusions (e.g.: **/*Test*.class)
/.exec		

Path to class directories (e.g.: **/target/classDir, **/classes): **/classes

Path to source directories (e.g.: **/mySourceFiles)	Inclusions (e.g.: **/*.java,**/*.groovy,**/*.gs)	Exclusions (e.g.: generated/**/*.java)
**/src/main/java	**/*.java	

☐ Disable display of source files for coverage

☐ Change build status according the thresholds

	Instruction	% Branch	% Complexity	% Line	% Method	% Class
☀	0	0	0	0	0	0
⛈	0	0	0	0	0	0

☐ Fail the build if coverage degrades more than the delta thresholds

	Instruction	% Branch	% Complexity	% Line	% Method	% Class
⛈	0	0	0	0	0	0

5.4.4. Jenkins archived outputs

In order to save our test results and artefacts, we added the following to our Jenkins file to ensure that Jenkins tracks our JUnit tests.

```
pipeline {
    agent any
    stages {
        stage('Test') {
            steps {
                sh './gradlew check'
            }
        }
    }
    post {
        always {
            junit 'build/reports/**/*.xml'
        }
    }
}
```

6. Sprint Review

6.1. Stakeholder feedback summary:

Our client was surprised that almost all the required features were already added, and everything seemed to be in order. Client then requested for additional features to be added (listed below).

Additional client requirements (as noted by the Product Owner)

- Stateful vending machine with amount of cash inside
- Admin can keep track of cash in the Vending Machine
- Additional denominations (up to $100)
    - Greedy algorithm for payment acceptance

The additional requirements will be broken down into user stories and tasks in the next sprint planning meeting.

Overview:

- **What has been done?**
    - **About 90% of the tasks** that were created as part of the initial product backlog were completed in the initial sprint. This included the first additional client requirement of including a currency payment feature.
- **What has not been done?**
    - The records class and the functionality to generate a record of daily transactions were not completed.
    - Not enough test cases and code coverage.
- **Work that has been added:**
    - The need to create a staff specific User interface.
    - An Inventory class.
    - Inventory management functions
- **Work removed from the sprint**
    - None.

6.2. Revised Product Backlog

During the sprint, we realised there were some tasks that could not be completed without implementing certain utility functions, such as Inventory, and inventory management which ‘Records’ is dependent on for example. As such these were subsequently implemented and added onto the product backlog.

This is admittedly not good scrum practice, and should have been included in the revised product backlog instead. To make things a bit clearer, the new additions are labelled with (*NEW) in the original product backlog, which is the one that appears at the start of the report. And that will also technically be our revised product backlog. As such, please refer to that product backlog as both the initial and revised version that is to be used for the next sprint.

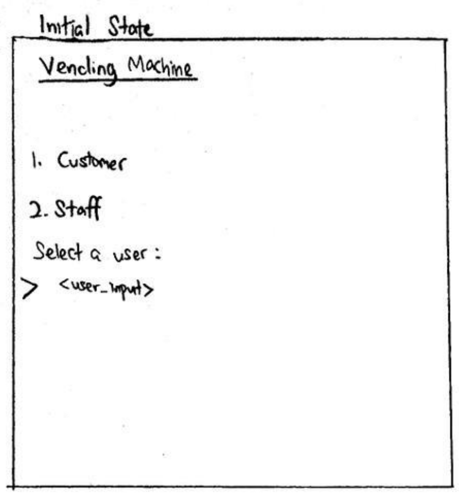
7. Sprint Retrospective

Template used: Start, Stop, Continue

Start
● Working on separate branches specific to certain features. ● Using clear, concise commit messages. ● Communicating on what they wish to do, and how they are going to do it. ● Using timeboxed tasks/events
Stop
● All working on 1 branch. ● Using vague commit messages ● Doing too many tasks in 1 go.
Continue
● Following good OOP practices. ● Attempting to use good design patterns.
TEAM: Patrick, Kunxi, Ken, Oscar, Alex DATE: 22/10/2019

8. Output/outcomes of key scrum events

8.1. Sketches of Interfaces

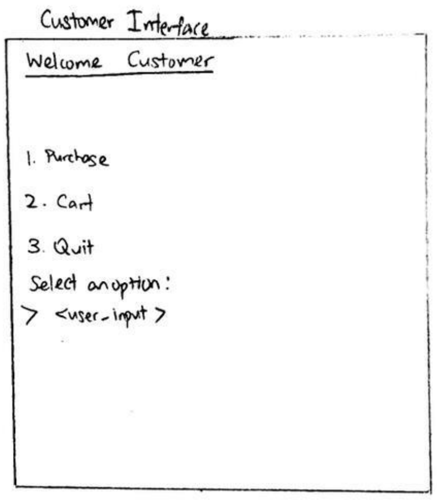

Initial State

Vending Machine

1. Customer
2. Staff

Select a user :

> <user_input>

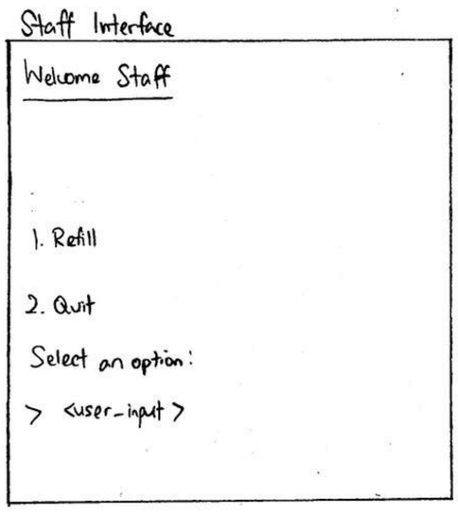

Customer Interface

Welcome Customer

1. Purchase
2. Cart
3. Quit

Select an option :

> <user_input >

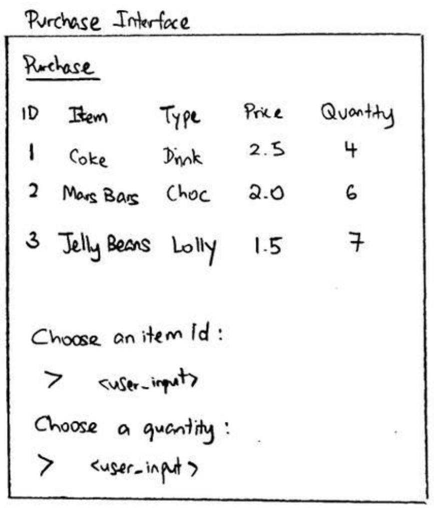

Staff Interface

Welcome Staff

1. Refill
2. Quit

Select an option :

> <user_input >


Purchase Interface

Purchase

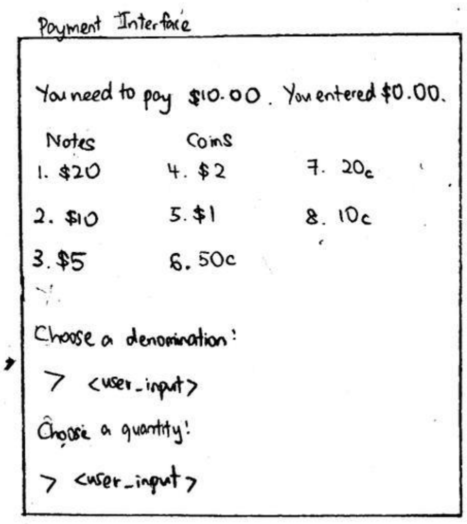
ID	Item	Type	Price	Quantity
1	Coke	Drink	2.5	4
2	Mars Bars	Choc	2.0	6
3	Jelly Beans	Lolly	1.5	7

Choose an item Id :

> <user_input>

Choose a quantity :

> <user_input >


Payment Interface

You need to pay $10.00 . You entered $0.00.

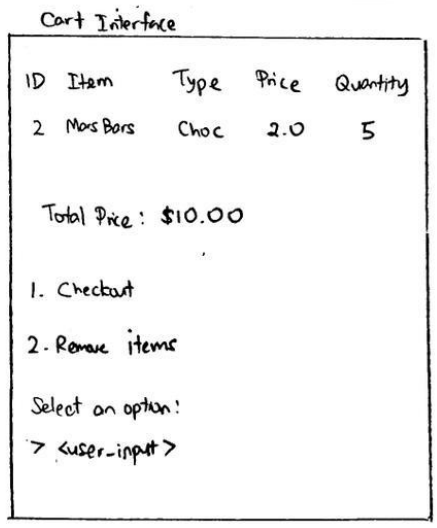
Notes	Coins	
1. $20	4. $2	7. 20c
2. $10	5. $1	8. 10c
3. $5	6. 50c	

Choose a denomination :

> <user_input>

Choose a quantity :

> <user_input >


Cart Interface

ID	Item	Type	Price	Quantity
2	Mars Bars	Choc	2.0	5

Total Price : $10.00

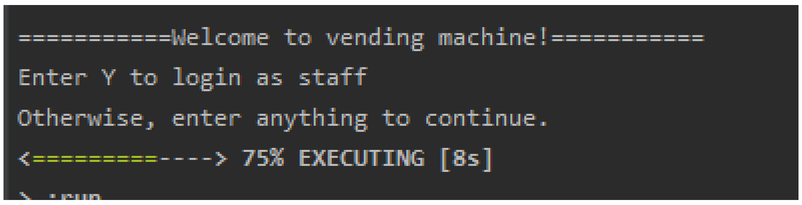
1. Checkout
2. Remove items

Select an option :

> <user_input >

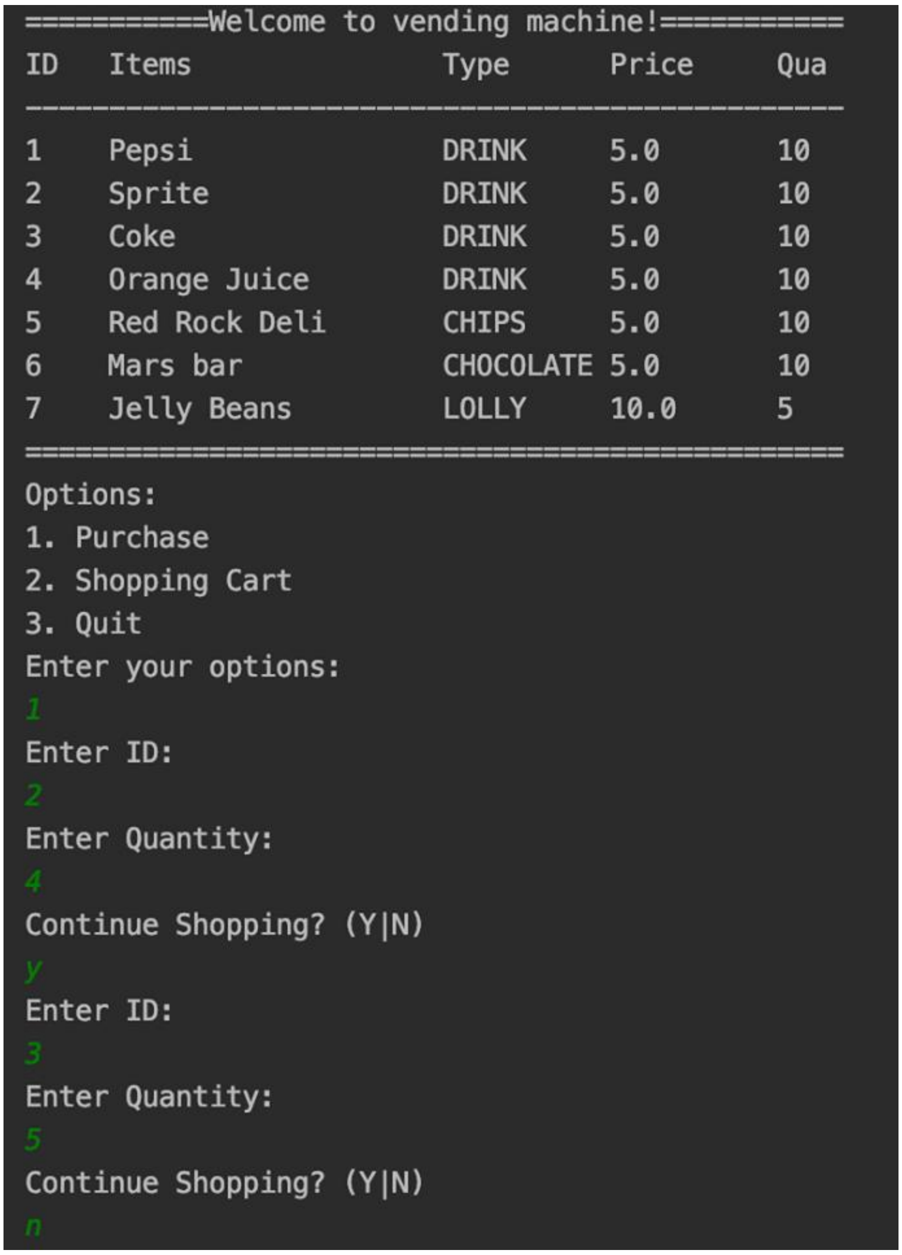
8.2. Program Demo (Screenshots)

1. Welcome Screen



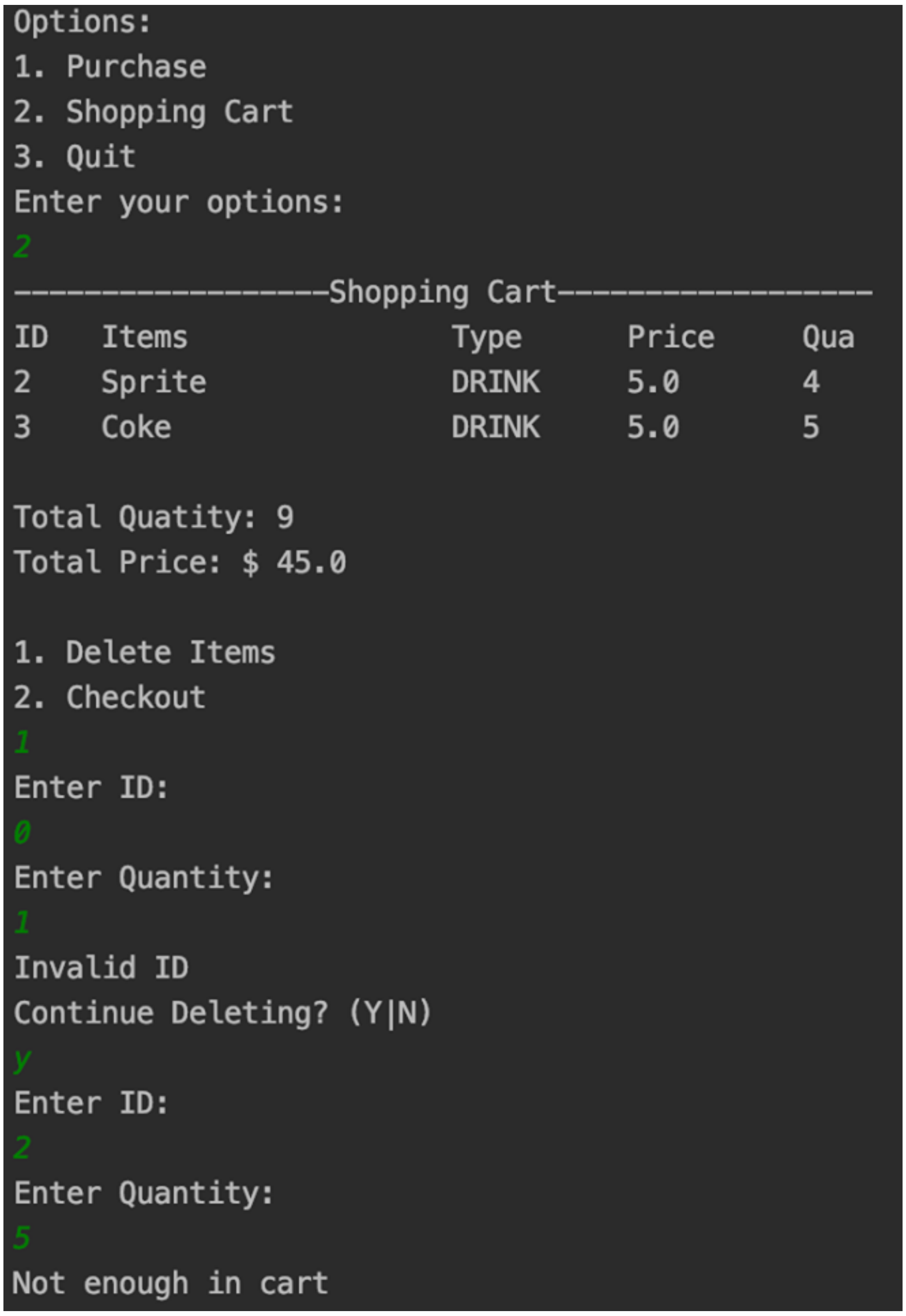
```
===========Welcome to vending machine!===========
Enter Y to login as staff
Otherwise, enter anything to continue.
<=========----> 75% EXECUTING [8s]
> :run
```

2. Display menu



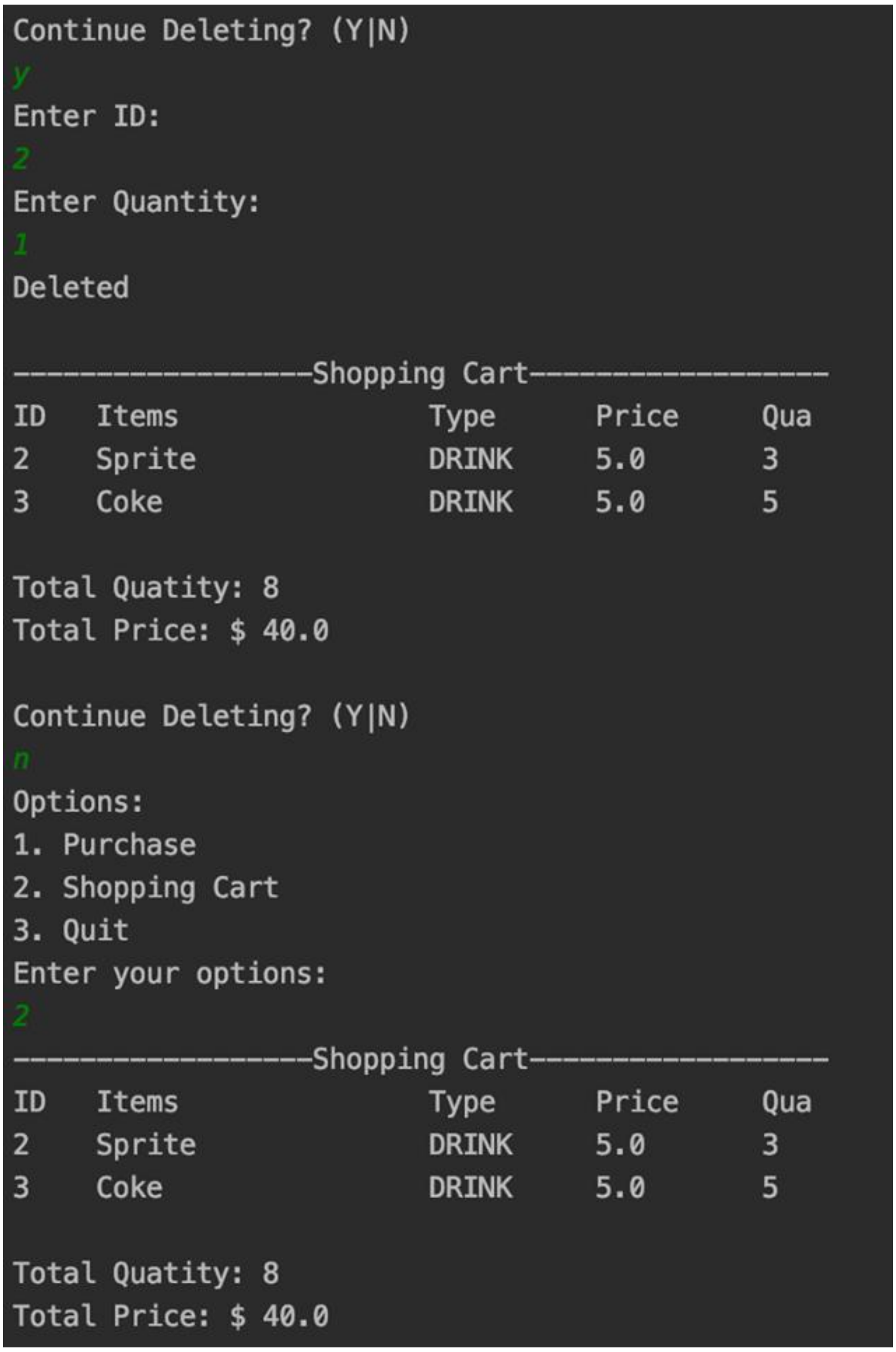
```
===========Welcome to vending machine!===========
ID   Items                Type      Price     Qua
-------------------------------------------------
1    Pepsi                DRINK     5.0       10
2    Sprite               DRINK     5.0       10
3    Coke                 DRINK     5.0       10
4    Orange Juice         DRINK     5.0       10
5    Red Rock Deli        CHIPS     5.0       10
6    Mars bar             CHOCOLATE 5.0       10
7    Jelly Beans          LOLLY     10.0      5
=================================================
Options:
1. Purchase
2. Shopping Cart
3. Quit
Enter your options:
1
Enter ID:
2
Enter Quantity:
4
Continue Shopping? (Y|N)
y
Enter ID:
3
Enter Quantity:
5
Continue Shopping? (Y|N)
n
```

3. Shopping Cart



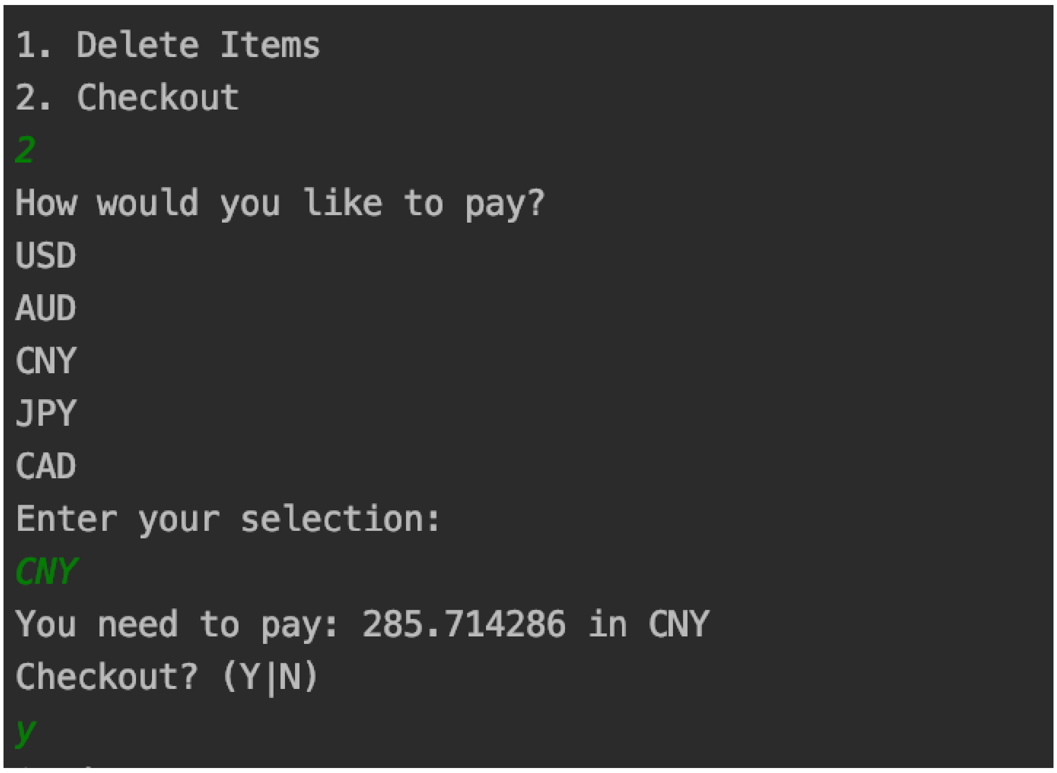
```
Options:
1. Purchase
2. Shopping Cart
3. Quit
Enter your options:
2
------------------Shopping Cart------------------
ID   Items              Type     Price     Qua
2    Sprite             DRINK    5.0       4
3    Coke               DRINK    5.0       5

Total Quatity: 9
Total Price: $ 45.0

1. Delete Items
2. Checkout
1
Enter ID:
0
Enter Quantity:
1
Invalid ID
Continue Deleting? (Y|N)
y
Enter ID:
2
Enter Quantity:
5
Not enough in cart
```

4. Shopping Cart cont.



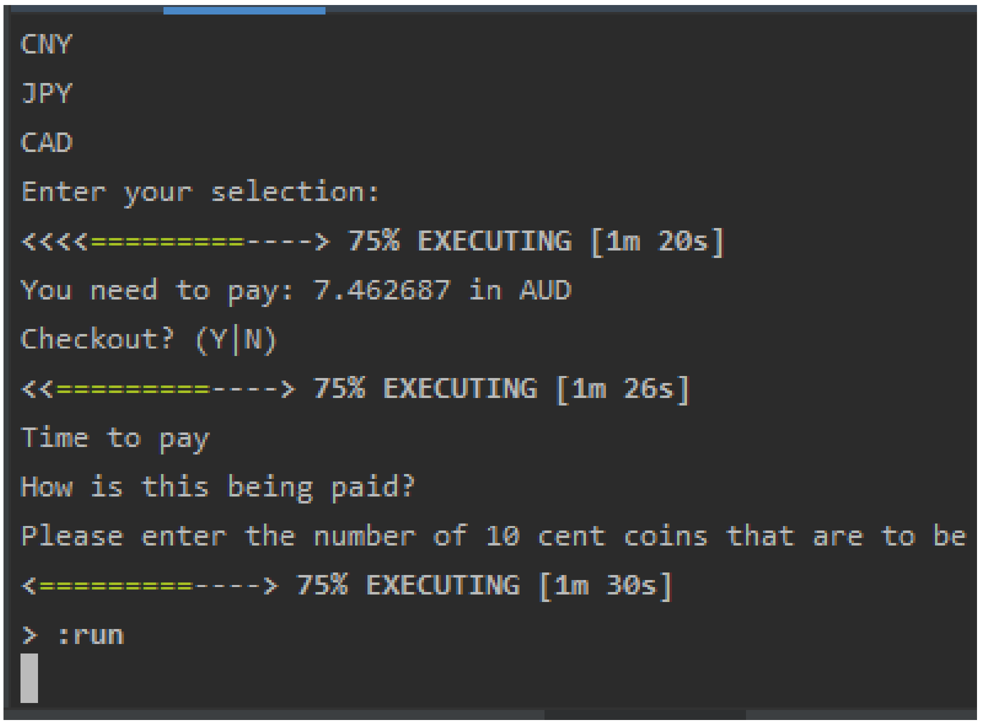
```
Continue Deleting? (Y|N)
y
Enter ID:
2
Enter Quantity:
1
Deleted

------------------Shopping Cart------------------
ID   Items                Type      Price     Qua
2    Sprite               DRINK     5.0       3
3    Coke                 DRINK     5.0       5

Total Quatity: 8
Total Price: $ 40.0

Continue Deleting? (Y|N)
n
Options:
1. Purchase
2. Shopping Cart
3. Quit
Enter your options:
2
------------------Shopping Cart------------------
ID   Items                Type      Price     Qua
2    Sprite               DRINK     5.0       3
3    Coke                 DRINK     5.0       5

Total Quatity: 8
Total Price: $ 40.0
```

5. Payment (currency)



```
1. Delete Items
2. Checkout
2
How would you like to pay?
USD
AUD
CNY
JPY
CAD
Enter your selection:
CNY
You need to pay: 285.714286 in CNY
Checkout? (Y|N)
y
```

6. Payment with denomination



```
CNY
JPY
CAD
Enter your selection:
<<<<=========----> 75% EXECUTING [1m 20s]
You need to pay: 7.462687 in AUD
Checkout? (Y|N)
<<=========----> 75% EXECUTING [1m 26s]
Time to pay
How is this being paid?
Please enter the number of 10 cent coins that are to be
<=========----> 75% EXECUTING [1m 30s]
> :run
```

9. Next Sprint Cycle

Implementation Forecast:

- Implement Records class and related features.
- Implement the staff fill() command with Id.
- Refactor code, instead of Food knowing about quantity, only inventory should know about quantity. **(high priority improvement)**
- Implement additional requirements stated by client post spring cycle 1.
- Correct payment where user does not have to pay exact and can receive change
- Correct when user selects, stock is reduced accordingly
- Correct user can only select where quantity <= stock

- Don't standard input to enter staff mode
- Standard input string to match the staff mode
- Report feedback
  - User stories should be in the backlog
  - Talk about how story points are calculated / what they mean
-
    -