

**School of Information Technologies**  
Faculty of Engineering & IT

## ASSIGNMENT/PROJECT COVERSHEET - GROUP ASSESSMENT

**Unit of Study:** SOFT2412 Agile Software Development Practices

**Assignment name:** Agile Software Development with Scrum and Agile Tools - Sprint 3

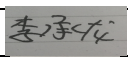
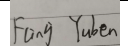
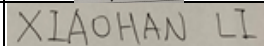
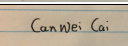
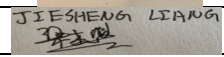
**Tutorial time:** Thu 6p.m. - 8p.m. R18G3 **Tutor name:** Muhit Saleh Anik

### DECLARATION

We the undersigned declare that we have read and understood the *University of Sydney Student Plagiarism: Coursework Policy and Procedure*, and except where specifically acknowledged, the work contained in this assignment/project is our own work, and has not been copied from other sources or been previously submitted for award or assessment.

We understand that failure to comply with the *Student Plagiarism: Coursework Policy and Procedure* can lead to severe penalties as outlined under Chapter 8 of the *University of Sydney By-Law 1999* (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

We realise that we may be asked to identify those portions of the work contributed by each of us and required to demonstrate our individual knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

Project team members				
Student name	Student ID	Participated	Agree to share	Signature
1. Chenglong Li	490035344	Yes	Yes	
2. Yuben Fang	480480835	Yes	Yes	
3. Xiaohan Li	470011746	Yes	Yes	
4. Canwei Cai	490032435	Yes	Yes	
5. Jiesheng Liang	480342832	Yes	Yes	
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	

# 1.Scrum Development

## 1.1 Project Team

The role is determined in the meeting before Sprint 1 starts:

Product Owner: Jackson(Canwei) Cai

Scrum Master: Jacky(Xiaohan) Li

Core Team members: Jerry(Chenglong) Li, Lucius(Yuben) Fang, Jackson(Jiesheng) Liang

## 1.2 Sprint goal

In this Sprint, we aimed to finish all the user stories of the whole project. We need to add new functionalities about cashier and owner. We also plan to finish the pay by cash functionality and the data interaction with the database. Besides, we need to find out a way to test the interface in the terminal to achieve the test coverage is over 75%.

## 1.3 Tasks Board

### 1.3.1 Product backlog

We have 4 User Stories for this Sprint. In this Sprint. There are 6 Story points in total.

- 1.As a registered customer , I want to pay the transaction with cash so that I can pay without a credit card.
- 2.As an anonymous customer I want to pay the transaction with cash so that I can pay without a credit card.
- 3.As an Owner, I want all the ability of both Seller and Cashier (include the report) so that I have the highest right.
- 4.As a Cashier, I want to modify the quantity of notes and coins in the machine so that I can.

### 1.3.2 Sprint backlog

In the Sprint backlog, we divide each user story into several tasks.

User Story 1.As a registered customer , I want to pay the transaction with cash so that I can pay without a credit card.

- 1 An interface that customer can input different amounts of different kinds of face values of money
- 2.After the customer pays the cash , the machine will return the exact change to me and also have an interface to show how many changes were returned.
- 3.If the machine does not have enough change, the machine will ask the customer to pay in another way(card or a smaller face amount dollar).Or customer can cancel the transaction.

User Story 2 .As an anonymous customer I want to pay the transaction with cash so that I can pay without a credit card.

- 1 An interface that customer can input different amounts of different kinds of face values of money
- 2.After the customer pays the cash , the machine will return the exact change to me and also have an interface to show how many changes were returned.
- 3.If the machine does not have enough change, the machine will ask the customer to pay in another way(card or a smaller face amount dollar).Or customer can cancel the transaction.

User story 3 .As an Owner, I want all the ability of both Seller and Cashier (including the report) so that I have the highest right.

- 1.The owner should obtain a report containing a list of usernames in the vending machine with the associated role (customer, seller, cashier, or admin).
2. The owner should obtain a summary of the cancelled transaction.
- 3.Owner should be able to add a seller, cashier, owner user.
- 4.Owner should be able to remove a seller, cashier, owner, user.
- 5.Functionality of change item details(all functionality of seller)
- 6.The interface of change item details
- 7.Functionality of modify available changes(all functionality of cashier)
- 8.The interface to modify changes

User story 4. As a Cashier, I want to modify the quantity of notes and coins in the machine so that I can.

- 1.The cashier should obtain a report containing a list of the current available change (the quantity of each coin and each note in the vending machine).
- 2.The cashier should obtain a summary of transactions that includes transaction date and time, item sold, amount of money paid, returned change and payment method.
- 3.The Cashier has an interface to modify the quantity of each coin .
- 4.The operation of the Cashier will affect the figure in the database.

## **1.4 Scrum Events Artefacts**

### **1.4.1 Sprint Planning**

In sprint 3, we have to finish all user stories, so we planned to complete cash payment, users should be able to pay transactions by inserting notes or coins and the vending machine needs to be able to get changes back. Then the cashier role and owner should be implemented, the cashier should have ability to change number of changes and generate transaction summary report and number of changes report. For the owner, the functionality and interface about adding and removing different kinds of user and generate the username list and cancelled transaction record as reports.

### **1.4.2 Daily Scrum**

The communication method has already been defined in the Sprint1 report. In this scrum two meetings are held because the common time of the whole group is less in this sprint.

In the first meeting, the Product Owner discusses and splits the remaining task with team members. Also, the Product Owner emphasises that this sprint should contribute and focus on the test case and the final product. As a result, independent tasks and related test cases should be completed and integrated in private before the second scrum meeting.

In the second meeting, team members complete most of the integration,so the group plans to complete the remaining integration first and ensure most of the functionalities are realized, then write test cases for the user interface and prepare for the final demonstration and report.

### **1.4.3 Challenge**

1. This sprint is also the last week of the semester, so every group member has lots of issues to do. As a result, it is hard to find a common time to meet or integrate things together. As a result, team members should complete their own work earlier so that when integrating the final product, team

members can firstly integrate as a pair. Because though finding the common time of the whole team is hard, finding the common time of two people is easy.

2. The group meets an issue that since it is a command-line UI product, the code coverage can hardly cover up the UI which may cause the coverage is under 75%. To solve the problem, group members decide it and search for the solution in the scrum meeting. Finally, they find they can simulate the user's input for the interface, but the input should concisely follow the steps of the UI which takes a while to achieve it.

3. The scenario demonstration for the product is also a challenge. Since every group member has to present in the final video demo, the consistency of the database is hard to control because the group uses the local database which would be different in everyone's local machine. What's more, the stability of remote control is not good. As a result, we try to solve this by uploading the current database once the operation in the previous scenario is done by the member. For the member is responsible for the next scenario, he/she should download the database, then perform the right operation shows in the next scenario and upload the uploaded database. This should be repeated until the last scenario.

#### 1.4.4 Sprint review

In this sprint, we almost completed everything in our plan which is much better than the last two sprints and we assign tasks to user stories properly and become more experienced about using jira, the burndown chart performs well for this sprint. However, we can do more testing if there was more time or we performed a better time arrangement.

#### 1.4.5 Sprint retrospective

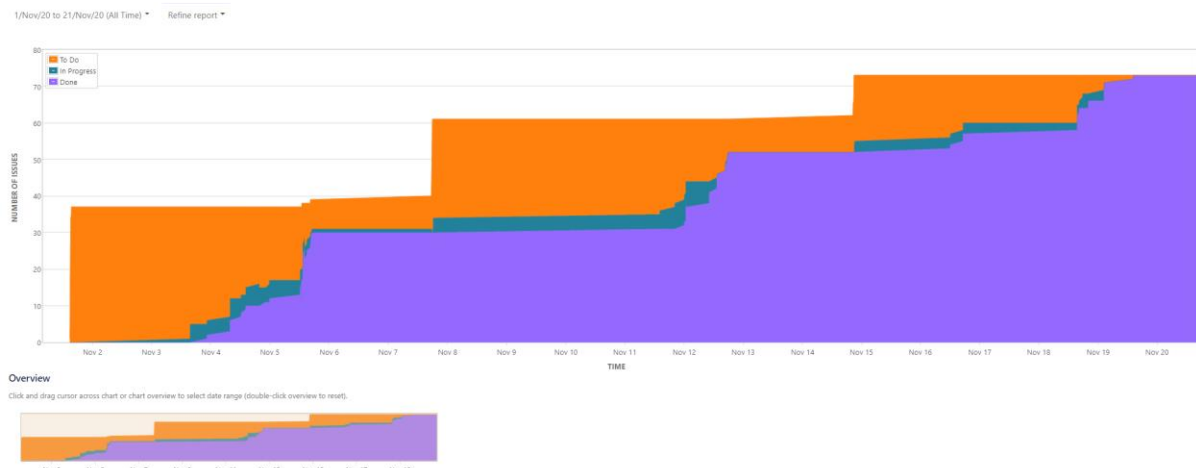
As mentioned in sprint review, we perform better in distribution of tasks and daily improvement of the implementation of the program which is shown in the burn down chart below. However, we lack daily scrum events to report our improvements, issues and plans for the next day. We can try to have more daily scrums in our future scrum projects which will be much helpful to the process of the construction of our program.

#### 1.4.6 Jira burndown chart



Learned from the previous two sprint's experiences, we improve and provide a more detailed user stories along the corresponding tasks. As a result, compared to the previous two burndown charts(see evidence), sprint 3 burndown chart is closer to the estimated line(guideline). This means the group is more and more able to master how to manage a team and process the product in the Scrum method.

## 1.4.7 Sprints overview



This is a Cumulative Flow Diagram which is used to show the statuses of issues over time. The flow shows the team performs a good level of team management and tasks allocation because the tasks ,which need to be completed, decrease step by step which means it remains more time for the final integration and commissioning. This is good since every product needs more time to test the implementation of functional or nonfunctional requirements and this work should be considered as a part of the project. What needs to improve is that s team member should respond to the Jira in real-time once a task’s status has changed. This can be considered as a part of team communication which can lead a more effective team management.

## 2. Agile development tools

The agile development tools we used can be found in the *sprint1 report 2Agile development tools*.

Jacoco - Overall Coverage Summary		
INSTRUCTION	80%	<div></div>
BRANCH	75%	<div></div>
COMPLEXITY	70%	<div></div>
LINE	75%	<div></div>
METHOD	85%	<div></div>
CLASS	91%	<div></div>

The jacoco Report is generated by jacoco plugin in jenkins which is used to check code coverage, it is clear that about 75% of lines are tested which is a great improvement from the last 2 sprints. In sprint 3, command line interfaces testing are performed, therefore the code coverage increased obviously.

\* The setup version and how to run the program is written in the readme.txt in the root folder of code.

## 3. Application development

### 3.1 Product guideline

#### 3.1.1 Sign up/ Log in (homepage):

```
make a selection
1. create an customer account
2. login
3. Anonymous customer
4. exit
please enter a selection
<=====----> 75% EXECUTING [23s]
> :run
```

When running the vending machine using ‘gradle run’, the user firstly will get the homepage which has 4 options shown in the image above. The user can input:

1. create a new consumer account
2. login
3. login as an anonymous
4. exit the vending machine

```
please input a name to sign up.
<=====----> 75% EXECUTING [31s]
> :run
please input a password to sign up.
<=====----> 75% EXECUTING [46s]
```

When the user input 1, it will ask the user to input a username followed by a password to sign up. If successful, it will automatically turn to customer guidelines with the new account.

```
please input a name to login.
<=====----> 75% EXECUTING [12s]
please input a password to login.
<=====----> 75% EXECUTING [23s]
```

When the user input 2, it will request the user to input username and password. After verification with the database, the user can login to the customer guideline as well.

When the user input 3, the user will log in as an anonymous then turn to the customer guideline.

The vending machine will be exited after the user input 4.

### 3.1.2 Selecting items (Customer guideline):

```
Hi left nice to meet you :)
last 5 items: null

choose a function you want to use
1. Shopping cart
2. List all categories
3. Cancel transaction
4. Pay for transaction
Please enter a selection
<=====----> 75% EXECUTING [6m 57s]
> :run
```

After login, you can see a page containing a hello message to the customer and some buttons. The line “last 5 items” means the last five items that were bought by this customer.

The first button can show the content of your shopping cart in this transaction, it is empty now since it is just logged in. The second button can show the product categories menu, it helps the user find what they need more easily. The third button and fourth button can make this transaction end in different ways.

choose a function you want to use

1. Drinks
2. Chocolates
3. Chips
4. Candies
5. Cancel transaction
6. Pay for transaction
7. Back

Please enter a selection

```
<=====-----> 75% EXECUTING [1m 13s]
> :run
```

After selecting the “List all categories”, the customer can see all product categories listed and they can select one of them to find product information in that category.

```
Item_code: 1001 Product: MineralWater Inventory: 7 Price: 1.5
Item_code: 1002 Product: Sprite Inventory: 7 Price: 3.0
Item_code: 1003 Product: Pepsi Inventory: 7 Price: 3.0
Item_code: 1004 Product: Juice Inventory: 5 Price: 2.5
```

choose a function you want to use

1. Put product into the cart
2. Cancel transaction
3. Pay for transaction
4. Back

Please enter a selection

```
<=====-----> 75% EXECUTING [1m 41s]
> :run
```

After choosing the drink category, we can see 4 items belong to this category with their information including name, inventory and price. The customer can choose option 1 to put items on this page into the shopping cart.

Input the name of the product(case insensitive):

```
<=====-----> 75% EXECUTING [2m 20s]
> :run
sprite
```

Input the quantity of the product

```
<=====-----> 75% EXECUTING [3m 23s]
> :run
1
```

Program will request the customer input the product name and amount they want to buy, the name is case insensitive and the quantity must be positive.

Product added

-----

choose a function you want to use

1. Drinks
2. Chocolates
3. Chips
4. Candies
5. Cancel transaction
6. Pay for transaction
7. Back

Please enter a selection

```
<=====-----> 75% EXECUTING [3m 47s]
> :run
```

After the user correctly input information, the success message will show up at top and back to the category menu.

```
Shopping cart:
product name,quantity
Sprite,1
total price: 3.0
-----
```

The customer can go back to the page after login and select button 1: shopping cart to see the product just added and the total price of the product inside the cart.

### 3.1.3 Pay by card:

Process the transaction when customers choose their products:

```
total price: 3.0

choose a function you want to use
1. Pay by cash
2. Pay by card
3. Cancel transaction
4. Back
Please enter a selection
<=====--> 75% EXECUTING [28s]
> :run
█
```

1. Customer can have two selections by inputting number 1 or 2 with: provide the card details or cancel the transaction

```
make a selection
1. provide your card details
2. cancel the transaction
please enter a selection
```

2. When customer decides to input 1

2.1 If the registered customer has not saved his card in the system, he/she needs to provide the card details including card holder's name and card number. The transaction would process only if the customer input the right format of card details shown in the json file. Then the customer can input y or n to save or not save his card in the system.

```
<=====--> 75% EXECUTING [21s]
please provide cardholder's name:
<=====--> 75% EXECUTING [26s]
please provide your card number:
<=====--> 75% EXECUTING [38s]
Congratulation! The system would process with your provided card.
Your transaction has been successfully processed!
Do you want to save your card details for next time shopping? y/n?
<=====--> 75% EXECUTING [50s]
System has successfully saved your card!
Please take your items and welcome for your next shopping!
```

2.2 If the registered customer has already saved his/her card in the system, the transaction would automatically process and the card holder's name would be output.



```
<=====--> 75% EXECUTING [12s]
Congratulation! The system would process with your saved card.
Card Holder: Charles
Your transaction has been successfully processed!
Please take your items and welcome for your next shopping!
```

2.3 If it is an anonymous customer, he/she has no choice for save the card in the system even if he inputs the right card format.

```
<====<=====--> 75% EXECUTING [2m 28s]
Congratulation! The system would process with your provided card.
Your transaction has been successfully processed!
Please take your items and welcome for your next shopping!
```

### 3.1.4 Pay by cash:

Make payment when customers choose their products :

```
total price: 0.00
```

choose a function you want to use

1. Pay by cash
2. Pay by card
3. Cancel transaction
4. Back

Please enter a selection

```
<=====--> 75% EXECUTING [28s]
```

```
> :run
```

```
█
```

1. Let the customers to choose the way they want pay

```
make a selection
```

1. start paying cash
2. cancel the transaction
3. back

please enter a selection

```
<=====--> 75% EXECUTING [42s]
```

```
> :run
```

```
█
```

2. After the customer chooses use cash to pay, it will comes up the option as the image shows

```
make a selection
```

1. start paying cash
2. cancel the transaction
3. back

please enter a selection

```
<=====--> 75% EXECUTING [1m 10s]
```

```
insert the quantity for face 100.00:
```

```
<=====--> 75% EXECUTING [1m 12s]
```

```
insert the quantity for face 50.00:
```

```
<=====--> 75% EXECUTING [1m 13s]
```

```
insert the quantity for face 20.00:
```

```
<<=====--> 75% EXECUTING [1m 16s]
```

```
insert the quantity for face 10.00:
```

```
<=====--> 75% EXECUTING [1m 18s]
```

```
insert the quantity for face 5.00:
```

```
<<=====--> 75% EXECUTING [1m 30s]
```

```
> :run
```

```
1█
```

3. when the customers choose start pay by cash, it will let customer to enter the number of each face-value's currency

```

please enter a selection
<=====--> 75% EXECUTING [2m 18s]
insert the quantity for face 100.00:
<=====--> 75% EXECUTING [2m 19s]
Blank entry is not allowed.
insert the quantity for face 100.00:
<=====--> 75% EXECUTING [2m 20s]
<insert the quantity for face 50.00:
<=====--> 75% EXECUTING [2m 21s]
insert the quantity for face 20.00:
<=====--> 75% EXECUTING [2m 23s]
insert the quantity for face 10.00:
<=====--> 75% EXECUTING [2m 24s]
insert the quantity for face 5.00:
<=====--> 75% EXECUTING [2m 25s]
insert the quantity for face 2.00:
<=====--> 75% EXECUTING [2m 27s]
insert the quantity for face 1.00:
<=====--> 75% EXECUTING [2m 28s]
insert the quantity for face 0.50:
<=====--> 75% EXECUTING [2m 29s]
insert the quantity for face 0.20:
<=====--> 75% EXECUTING [2m 30s]
insert the quantity for face 0.10:
<=====--> 75% EXECUTING [2m 31s]
insert the quantity for face 0.05:
<=====--> 75% EXECUTING [2m 32s]
Shopping cart:
product name,quantity
Juice,2
total price: 5.0

Inserted cash total amount: 5.0
return total change amount: 0.00
Changes:100.00:0 50.00:0 20.00:0 10.00:0 5.00:0 2.00:0 1.00:0 0.50:0 0.20:0 0.10:0 0.05:0
Thank you for the payment, have a good day
    
```

4. Once the customer finishes their payment, it will show the status of the transaction, display the amount of currency received, the total change amount returned and the amount of each currency returned. Shows the transaction was successful and print have a good day.

### 3.1.5 Seller guideline

1. After the user runs gradle run and login in with account(seller) and password(seller), he will login in as a seller.

```

1. create an customer account
2. login
3. Anonymous customer
4. exit
please enter a selection
<=====-----> 75% EXECUTING [6s]
please input a name to login.
<=====-----> 75% EXECUTING [8s]
please input a password to login.
<=====-----> 75% EXECUTING [13s]
Hi seller nice to meet you :)

```

```

choose a function you want to use
1. change item name
2. change item code
3. change item category
4. change item quantity
5. change item price
6. obtain <available items report>
7. obtain <summary report>
8. Exit
Please enter a selection

```

2. There are several functions for the seller. If the seller wants to change the name of the item, he needs to input the item\_code and input the new name for the item. And the item\_code has to be in the system and the new name of item should not be the same as the existing name in the system.

```

choose a function you want to use
1. change item name
2. change item code
3. change item category
4. change item quantity
5. change item price
6. obtain <available items report>
7. obtain <summary report>
8. Exit
Please enter a selection
<=====-----> 75% EXECUTING [1m 2s]
> :<=====-----> 75% EXECUTING [1m 3s]
<=====-----> 75% EXECUTING [1m 11s]
please enter ItemCode for this items
<=====-----> 75% EXECUTING [1m 14s]
please enter the new item_name of the item that you want to modify
<=====-----> 75% EXECUTING [1m 18s]

```

3. If the seller wants to change the code of the item, he needs to input the item's name and input the new item code for the item. And the name of item has to be in the system and the new code of item should not be the same as the existing code in the system.

```

choose a function you want to use
1. change item name
2. change item code
3. change item category
4. change item quantity
5. change item price
6. obtain <available items report>
7. obtain <summary report>
8. Exit
Please enter a selection
<=====-----> 75% EXECUTING [1m 52s]
please enter the item_name of the item that you want to modify
<=====-----> 75% EXECUTING [1m 56s]
please enter new ItemCode for this items
<=====-----> 75% EXECUTING [1m 58s]

```

4.If the seller wants to change the category of the item, he needs to input the item's name and input the new category for the item. And the name of the item has to be in the system and the new category of item should be in the system ,too.

```
choose a function you want to use
1. change item name
2. change item code
3. change item category
4. change item quantity
5. change item price
6. obtain <available items report>
7. obtain <summary report>
8. Exit
Please enter a selection
<=====--> 75% EXECUTING [2m 27s]
please enter the item_name of the item that you want to modify
<=====--> 75% EXECUTING [2m 31s]
please enter new category for this items
```

5. If the seller wants to change the quantity of the item, he needs to input the item's name and input the new quantity for the item. And the name of the item has to be in the system and the new quantity of item should be bigger than -1 and smaller than 16.

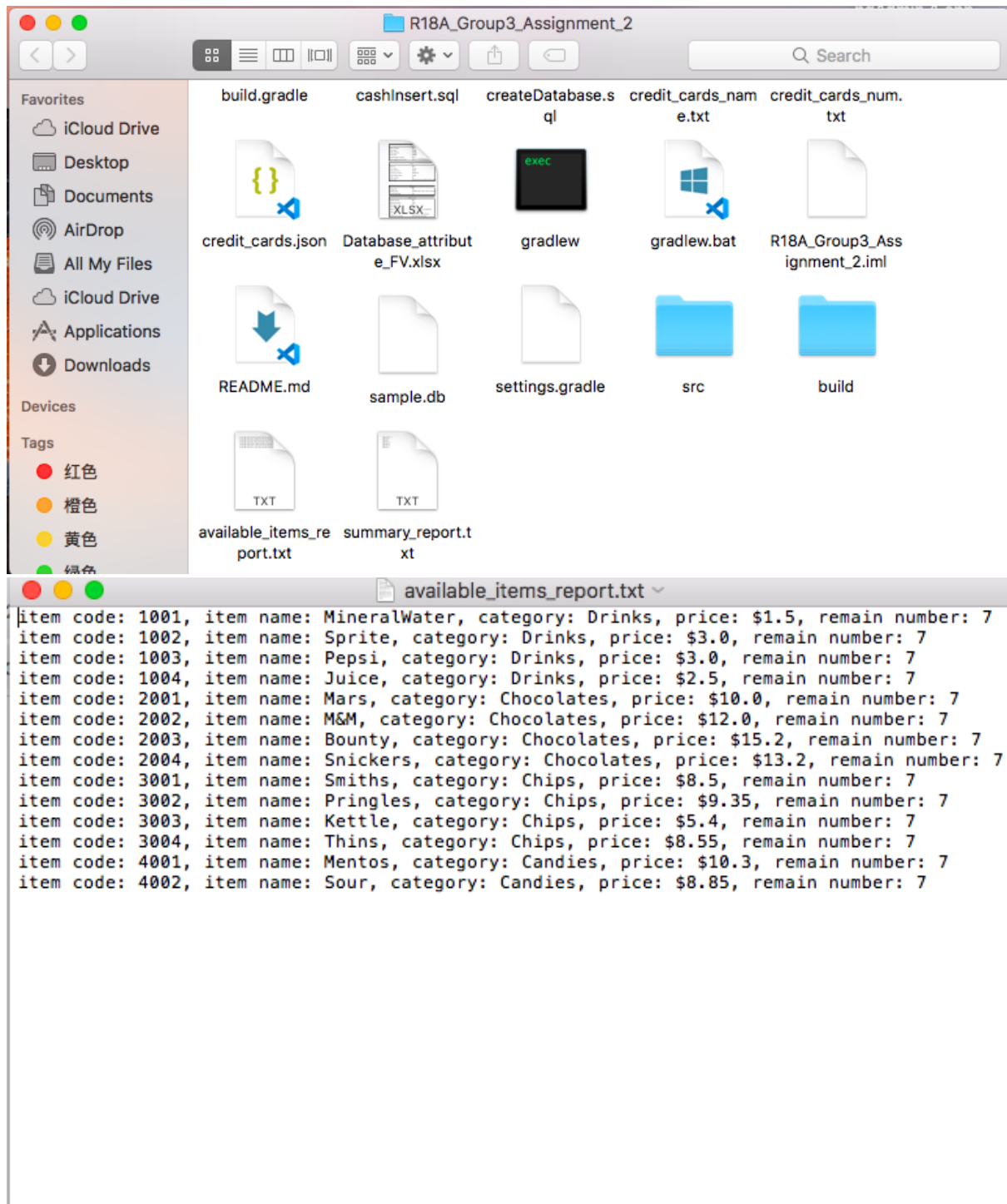
```
choose a function you want to use
1. change item name
2. change item code
3. change item category
4. change item quantity
5. change item price
6. obtain <available items report>
7. obtain <summary report>
8. Exit
Please enter a selection
<=====--> 75% EXECUTING [3m 11s]
please enter the item_name of the item that you want to modify
<=====--> 75% EXECUTING [3m 14s]
please enter new quantity
```

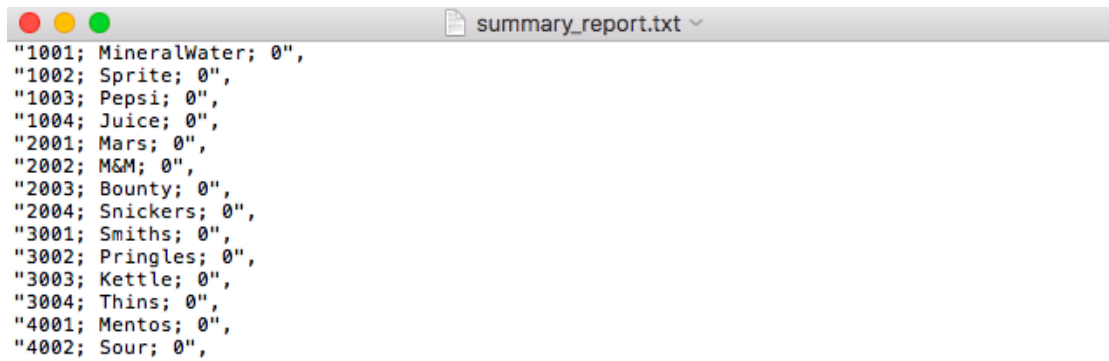
6. If the seller wants to change the price of the item, he needs to input the item's name and input the new price for the item. And the name of the item has to be in the system .

```
choose a function you want to use
1. change item name
2. change item code
3. change item category
4. change item quantity
5. change item price
6. obtain <available items report>
7. obtain <summary report>
8. Exit
Please enter a selection
<=====--> 75% EXECUTING [3m 55s]
please enter the item_name of the item that you want to modify
<=====--> 75% EXECUTING [4m 5s]
please enter new price
<=====--> 75% EXECUTING [4m 9s]
```

7. If the seller wants to see the available item report , he needs to input 6.(if the item quantity is 0, the item will not be available item report). If the seller wants to see the summary report, he needs to input 7

The reports are in the folder which is named R18A\_Group3\_Assignment\_2.

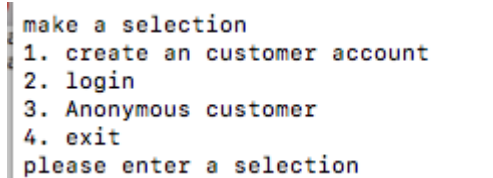




```
"1001; MineralWater; 0",  
"1002; Sprite; 0",  
"1003; Pepsi; 0",  
"1004; Juice; 0",  
"2001; Mars; 0",  
"2002; M&M; 0",  
"2003; Bounty; 0",  
"2004; Snickers; 0",  
"3001; Smiths; 0",  
"3002; Pringles; 0",  
"3003; Kettle; 0",  
"3004; Thins; 0",  
"4001; Mentos; 0",  
"4002; Sour; 0",
```

The report will not be automated. So, if the seller wants to see the newest report, he needs to input 6 ,7 to get the responding new report.

8. If a seller wants to exit the system, he can input 8 . And the screen will go back to login page



```
make a selection  
1. create an customer account  
2. login  
3. Anonymous customer  
4. exit  
please enter a selection
```

### 3.1.6 Cashier guideline



1. After the user runs gradle run and login in with account(cashier) and password(cashier), he will login in as a cashier.

```
make a selection
1. create an customer account
2. login
3. Anonymous customer
4. exit
please enter a selection
<=====75% EXECUTING [3s]
please input a name to login.
<=====75% EXECUTING [8s]
please input a password to login.
<=====75% EXECUTING [9s]
Hi cashier nice to meet you :)

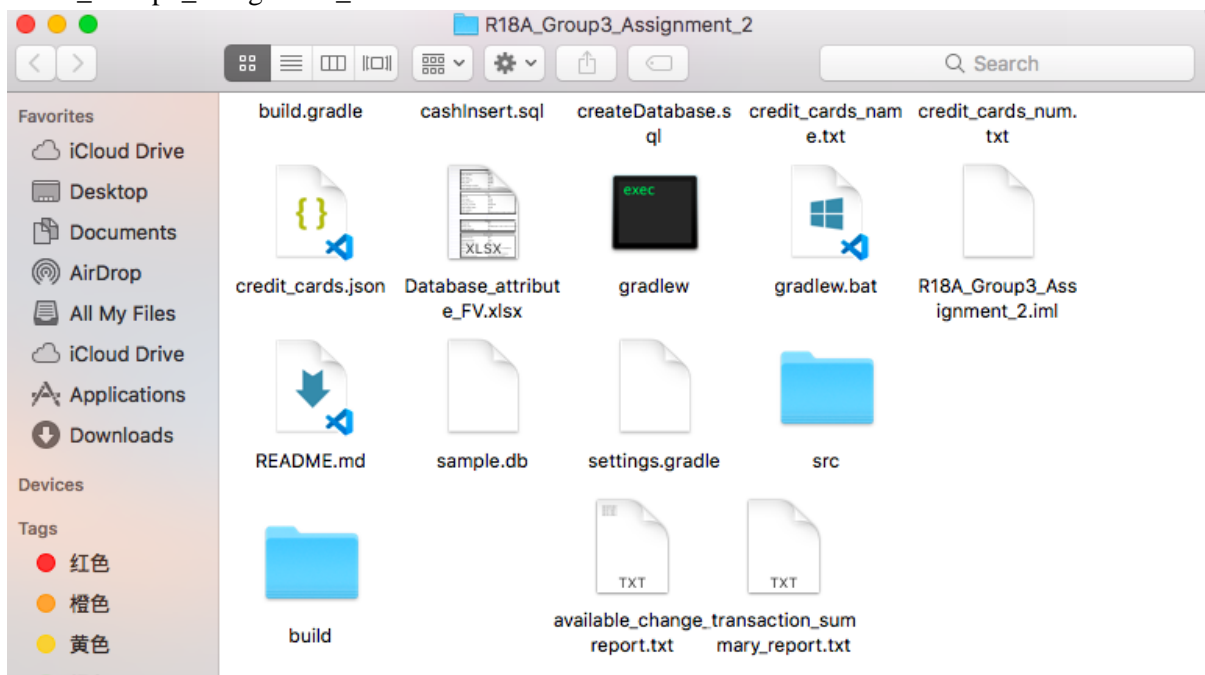
choose a function you want to use
1. change the quantity 100 dollars
2. change the quantity 50 dollars
3. change the quantity 20 dollars
4. change the quantity 10 dollars
5. change the quantity 5 dollars
6. change the quantity 2 dollars
7. change the quantity 1 dollar
8. change the quantity 0.5 dollars
9. change the quantity 0.2 dollars
10. change the quantity 0.1 dollars
11. change the quantity 0.05 dollars
12. obtain <available change report>
13. obtain <transaction summary report>
14. exit
Please enter a selection
<=====75% EXECUTING [11s]
```

2. Cashiers can modify the quantity of 100 dollars by type 1 and then input the new quantity.

```
Please enter the new quantity of 100 dollars
_____
```

Cashiers can also change the quantity of 50, 20, 10, 5, 2, 1, 0.5, 0.2, 0.1, 0.05 by using function 2 to 9. They have a similar operation.

3. If a cashier wants to obtain an “available change report” they can get a report in a folder named R18A\_Group3\_Assignment\_2.



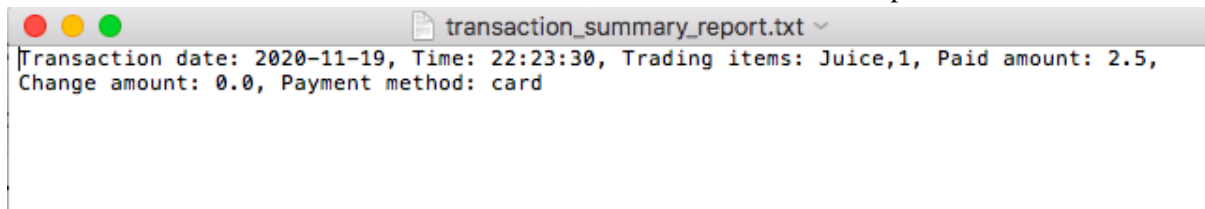


```

Change type: 100.0, Quantity: 5
Change type: 50.0, Quantity: 5
Change type: 20.0, Quantity: 5
Change type: 10.0, Quantity: 5
Change type: 5.0, Quantity: 5
Change type: 2.0, Quantity: 5
Change type: 1.0, Quantity: 5
Change type: 0.5, Quantity: 5
Change type: 0.2, Quantity: 5
Change type: 0.1, Quantity: 5
Change type: 0.05, Quantity: 5
    
```

4.If a cashier wants to obtain an “transaction\_summary\_report” they can get a report in a folder named R18A\_Group3\_Assignment\_2.

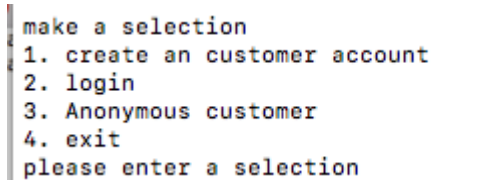
If there is a successful transaction in the machine, the cashier can see the report like this.



```

Transaction date: 2020-11-19, Time: 22:23:30, Trading items: Juice,1, Paid amount: 2.5,
Change amount: 0.0, Payment method: card
    
```

5. If a cashier wants to exit the system, he can input 14 . And the screen will go back to login page



```

make a selection
1. create an customer account
2. login
3. Anonymous customer
4. exit
please enter a selection
    
```

### 3.1.7 Owner guideline

Hi admin nice to meet you :)

choose a function you want to use

1. Add User
2. Remove User
3. use seller functions
4. use cashier functions
5. obtain <cancelled transaction summa:
6. obtain <usernames\_summary>
7. Exit

Please enter a selection

After login as an owner with username: admin and password: admin, there are 7 options which can be chosen by the owner.

1. It will require input username, password and user type of the user. after that, a new user will be created.



```
Please enter information of the user you want to add
username:
<=====----> 75% EXECUTING [1m 9s]
password:
<=====----> 75% EXECUTING [1m 43s]
Choose and input a kind of user type (customer, cashier, seller, owner)
<=====----> 75% EXECUTING [2m 6s]
```

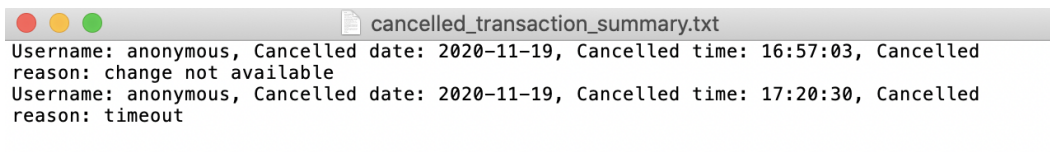
2. It will require input username to delete the user if it exists.

```
password:
<=====----> 75% EXECUTING [1m 43s]
```

3. the owner can get into the seller interface to use the functionality as a seller

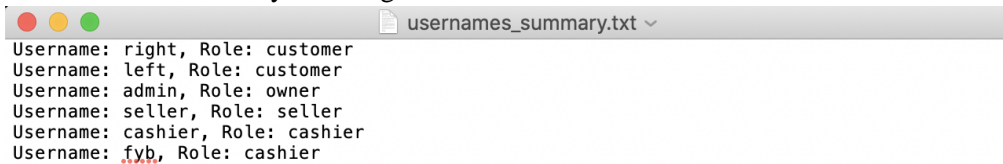
4. the owner can get in to cashier interface to use the functionality as a cashier

5. A cancelled transaction summary will be generated in the local folder which contains information of all the cancelled transactions.



```
cancelled_transaction_summary.txt
Username: anonymous, Cancelled date: 2020-11-19, Cancelled time: 16:57:03, Cancelled
reason: change not available
Username: anonymous, Cancelled date: 2020-11-19, Cancelled time: 17:20:30, Cancelled
reason: timeout
```

6. An account summary will be generated in the local folder with information of each user.



```
usernames_summary.txt
Username: right, Role: customer
Username: left, Role: customer
Username: admin, Role: owner
Username: seller, Role: seller
Username: cashier, Role: cashier
Username: fyyb, Role: cashier
```

7. log out and back to the homepage.

## 3.2 Functionalities

### 3.2.1 Pay by cash:

In pay by cash function both registered customers and anonymous customers can use cash payment to checkout their shopping cart.

- Select start payment. They should insert the quantity of each cash to the machine. The machine would calculate
- the changes for the customer base on the amount of the inserted money.
- If the inserted prices is not sufficient to pay the total prices in the cart or the machine fails to make changes based on the database changes quantities, the transaction is aborted and it would go back to the paymentInterface.
- If the inserted amount is sufficient and valid returned changes can be made, the changes state and transaction would be recorded in the database and go back to the home age.
- If the transaction is cancelled, cancelled details should be recorded in the database and the system should go back to the home age.

### 3.2.2 Owner functions:

The owner can add all kinds of types of user including customer, cashier, seller, and owner with an unique username and a password. Besides, it can remove an existing user with its username. Additionally, it can use all functionalities in seller and cashier. At last, a cancelled transaction report and a user account report can be generated in the local folder.

### 3.2.3 Owner reports:

There are two reports that can only be generated by the owner, cancelled transaction summary report and username list report. The first one is actually just created by obtaining data stored in the database which is updated by transaction functions, every time a transaction is paid successfully the program will update this information needed by this report. The second one is basically the same as the first one, it also reads data in the database, the last five items bought by each user are stored as a part of user information in the database, once a transaction is paid successfully the program will ignore the newly bought items if it exists in the last five items list because a item should not be added twice. In addition, if the list is full already, the program will remove the oldest item which always be leftmost in the list until every item bought this time is added into the last five items list.

### 3.2.4 Seller functions:

There are 7 functions for sellers. Five of them are used to modify the information of items and the last two functions are used to get the report. The operation will interact with the Item table in the database.

1. change item name : cashier needs to find the item code for a specific item and input a new name for that item.(conflicting item name is not allowed).
2. change item code : cashier needs to find the item name for specific item and input a new item code for that item.(conflicting item code is not allowed)
3. change item category: cashier need to find the item name for a specific item and input a new category of that item.(The category should be Drinks or Chocolates or Chips or Candies)
4. change item quantity: cashier need to find the item name for specific item and input a new quantity of that item.(New item cannot be negative and bigger than 15)
5. change item price : cashiers need to find the item name for a specific item and input a new price of that item.
6. obtain available items report : cashier can obtain all the related information for items. Such as, item code, item name , category , price , remaining quantity.
7. obtain a summary report :The report will record items codes, item names and the total number of quantities sold for each item.

### 3.2.5 Cashier functions:

There are 13 functions for sellers. Eleven of them are used to modify the quantity of all the notes and coins available in the vending machine. Once the cashier input a new quantity of notes/coins , the system will check whether the input is positive. If the quantity is positive, the system will interact with the database and update.

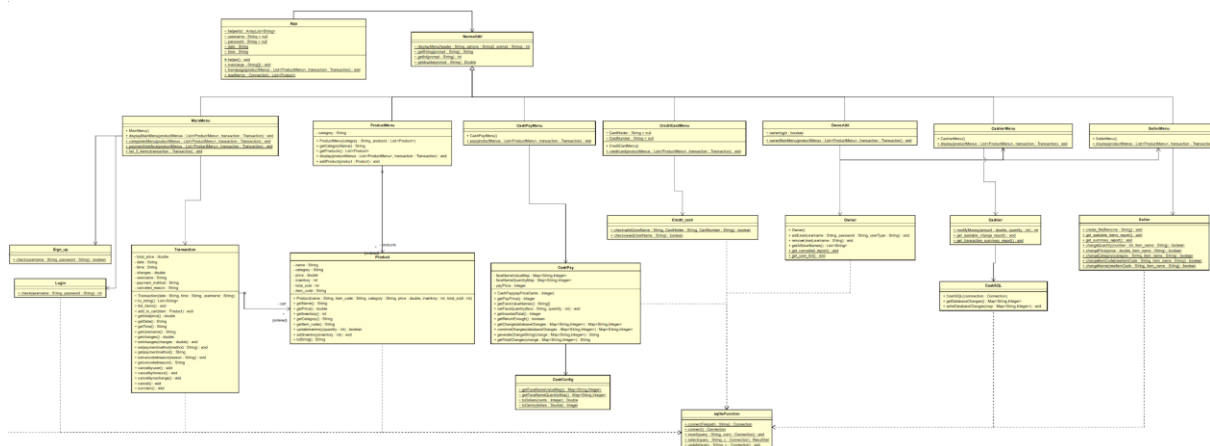
### 3.2.6 Cashier report:

There are two reports that can be generated by the cashier, available change report and transaction summary report. Once generated successfully, they would appear in the root folder of the product as txt files. The first report is actually just created by obtaining *Change* data stored in the database which contains the face value of the change and corresponding quantity. The second one is basically the same as the first one, but it obtains *Transaction* data in the database. A transaction is created by a

successful transaction with two different types, card and cash. It also includes payment date, time, items, total price and returned change.

\* Other functionalities can reference the previous two sprint reports.

### 3.3 Class diagram



Here is the class diagram of our whole program.

Please go to the link here to download the original file to zoom up if you can't see the picture here clearly.

<https://drive.google.com/file/d/1LneUGwap0W2-TtBNQSW0k3y4rRHUMqul/view?usp=sharing>

### 3.4 Sequence diagram/ User case diagram

Use case diagram:



There is a snack vending system. There are four roles in this system, customer, seller, cashier and owner.

Customers can create a user account and log in to the system, select a certain category of snacks, and then buy. When buying, the system provides two payment methods, cash and credit card. After the user uses a credit card, it will Record the information, and the credit card data will be automatically filled next time.

The seller can manage the information of each snack and modify their information.

The cashier will record the user's payment and add change to our system.

The owner can use all the functions of other roles, including but not limited to snack information management, payment information management, account management, and cashier management. Each role in the system has a different account, and permissions are strictly divided.

### 3.5 Database structure(recall)

The database structure can be found in *Sprint 1 report 3.2.5Database(see evidence)*

### 3.6 Setup & Version requirements

Java Version Requirement: Java 14 or higher

Set Up Database: If you want to refresh the database to the original version, you have to uncomment the ConnectSQLite.java class first. Then, change the mainClassName in the build.gradle to be 'ConnectSQLite' and use gradle run in the terminal to refresh the database.

Run the program: To run the program, firstly you need to check the mainClassName in gradle.run is 'App'. Then, use 'gradle build' to build the build the program. Finally, use gradle run to run the program.

### 3.7 Contribution

Jackson (Canwei) Cai - Sprint planning, Cashier UI, modify functions and related test cases, UI test cases,Sprint-3-report

Jacky (Xiaohan) Li - Scrum event records, Cashier report functions and related test cases ,Sprint-3-report

Jerry (Chenglong) Li - Agile tools management, Owner report functions, making cashier and seller functions available in owner menu and related test cases ,Sprint-3-report

Lucius (Yuben) Fang - Agile tools management, Owner UI, modify functions and related test cases ,UI test cases, Sprint-3-report

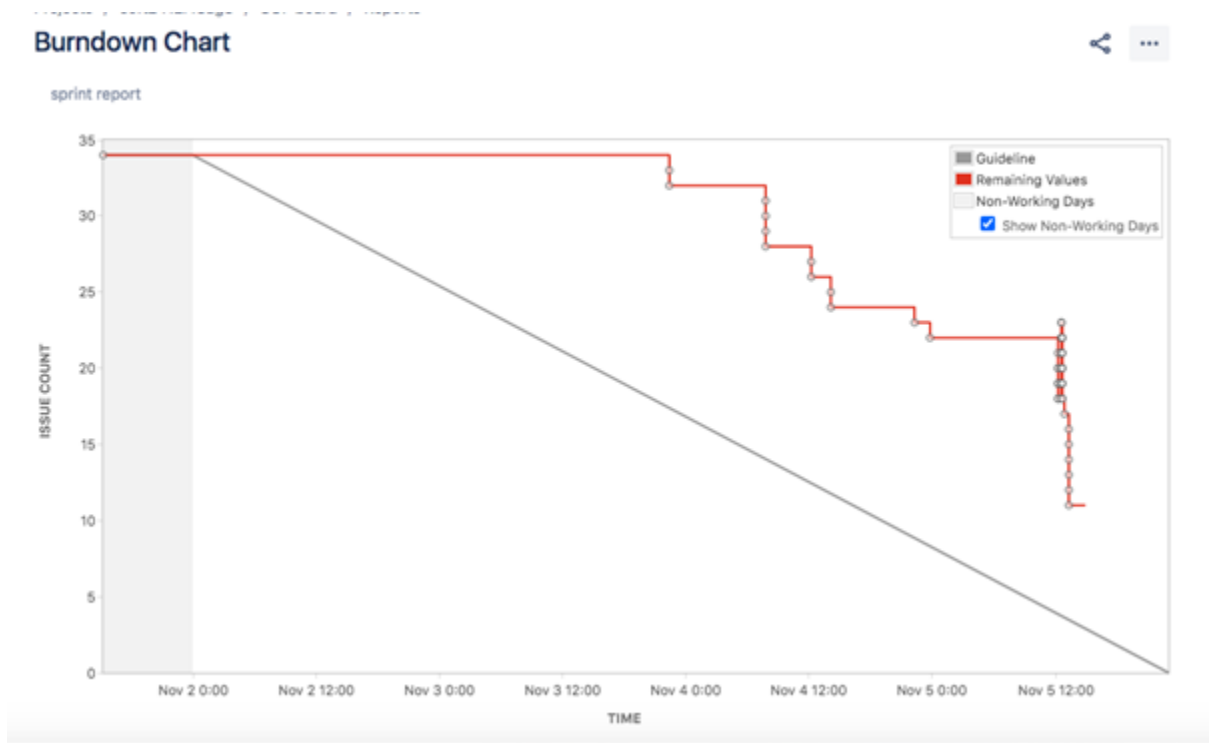
Jackson (Jiesheng) Liang - Pay by cash functions, Sprint-3-report

## 4. Evidence

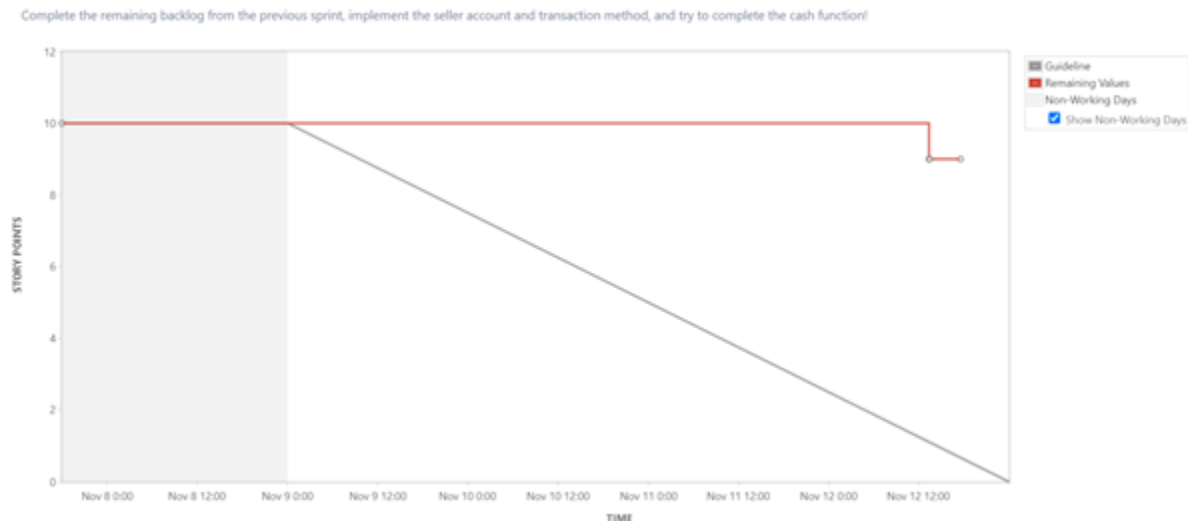
**Daily scrum:**



**Sprint 1 Jira Burndown chart:**



**Sprint 2 Jira Burndown chart:**



### Agile development tools:

We use github, gradle, junit and jenkins for this project as our tools to develop the product. In the github, members will create branches when they need to implement new features on the product. For gradle, we add dependencies so that we can use junit to test codes and generate jacoco test reports and also allow us to use the sql database. Using jenkins, we can apply our continuous integration on our project, it provides functionality like automated build and test. Every time group members push their code to the git repository, they can see their build and test result on jenkins server which is maintained by one of our members.

### Database structure:

The class 'ConnectSQLite' is used for creating a local sqlite database, generating the schema of each table and initialing some data before the whole program runs. The database is created in a local repository file called 'sample.db'. After that, we can connect to the database from the backend, obtain and send data from/to the database.

Item		
Item_name	PK	String
Item_category	not NULL	String(Drink or Chocolate or Chip or Candy)
Item_price	not NULL	Double
item_quantity_remain	not NULL	Int
Item_quantity_total_sold	not NULL	Int
User		
User_name	PK	String("anonymous" if anonymous user)
User_password	defalut NULL	String("NULL" if anonymous user)
User_last_5_items	defalut NULL	StringList
Card_holder_name	defalut NULL	String
Card_number	defalut NULL	String
User_type	not NULL	String(Customer,Seller,Cashier,Owner,Anonymous)
Change		
Change_id	PK	Int
Change_type	not NULL	Double(100 or 50 or 20 or 10 or 5 or 2 or 1 or 0.5 or 0.2 or 0.1 or 0.05)
Change_quantity	not NULL	Int
Transaction		
Transaction_id	PK	Int
Transaction_date	not NULL	Date
Transaction_time	not NULL	Time
Transaction_amount	not NULL	Double
Transaciton_items	not NULL	String(e.g. 1xcola,2xsneaker)
Transaction_change	not NULL	Double
User_name	FK	String
Transaction_method	not NULL	String(Cash or Card)
CancelledTransaction		
CancelledTransaction_id	PK	Int
CancelledTransaction_date	not NULL	Date
CancelledTransaction_time	not NULL	Time
User_name	FK	String
CancelledTransaction_reason	not NULL	String("timeout", "user cancelled", "change not available")

Sprint backlog - Jira Burndown chart