THE UNIVERSITY OF SYDNEY

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - GROUP ASSESSMENT

Unit of Study: **SOFT2412**

Assignment name: **Tools for Agile Software Development**

Tutorial time: Thursday 18:00-20:00 **Tutor name:** Ge Jin

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 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. Xiaoqian Hu	500105263	[✓ Yes] / No	[✓ Yes] / No	Xiaoqian Hu
2. Haolin Jin	510085113	[✓ Yes] / No	[✓ Yes] / No	[illegible]
3. Zechao Sun	510048657	[✓ Yes] / No	[✓ Yes] / No	Zechao Sun
4. Yiheng Yang	510231499	[✓ Yes] / No	[✓ Yes] / No	Yiheng Yang
5.		Yes / No	Yes / No	
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 Assignment2 Sprint1

Name	SID
Yiheng Yang	510231499
Zechao Sun	510048657
Haolin Jin	510085113
Xiaoqian Hu	500105263

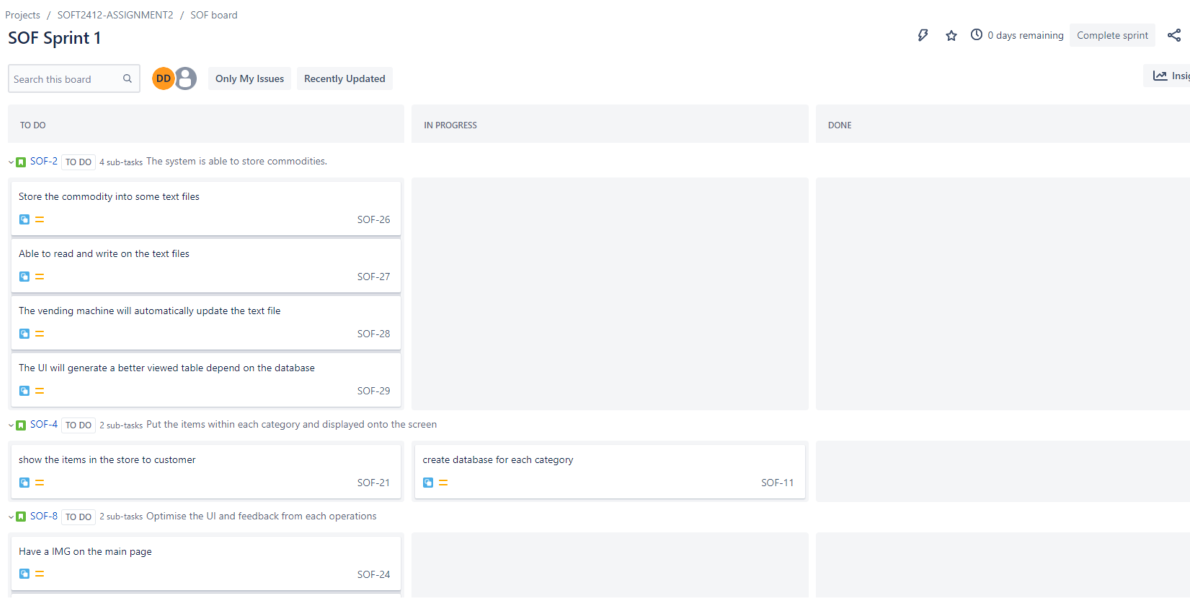
1. Scrum group collaboration

In this assignment we are developing the system using the scrum method of agile development, so we need to set up our scrum team first. There are three different roles in the scrum method which are Product owner, Scrum master and core team members. For the scrum master and product owner there is only one person responsible for these roles so we need to have a proper discussion of the role distribution before we actually start the assignment.

Our first meeting started on 8 October, we discussed the role distribution and the general structure of the vending machine system. After the first meeting we assigned Yiheng Yang to be the Product Owner and Haolin Jin to be the Scrum master and the rest of two teammates are core team members who are mainly responsible for the coding design of our software system.

After we set up the scrum team, we also generally talked about the design structure of our vending machine system. We decided to use the java swing as the UI tools in this assignment because our team members are quite familiar with this language. Before we actually work on the code we create a new github project and invite our group members, then we can set up the CI/CD pipeline by the agile tool which is the Jenkins to achieve the auto building and auto testing, Haolin Jin is still responsible for the Jenkins setup same with the previous assignment after that we are ready to go with the next processes.

Our second meeting was on 13 October, during this meeting we mainly focused on the start of scrum development so we discussed the user story and used the Jira to create the product backlog. Inside of the Jira software we created a new sprint for our first sprint by adding user stories and their subtasks into the product backlog.


Projects / SOFT2412-ASSIGNMENT2 / SOF board

SOF Sprint 1

0 days remaining | Complete sprint

Search this board | DD | Only My Issues | Recently Updated | Insi

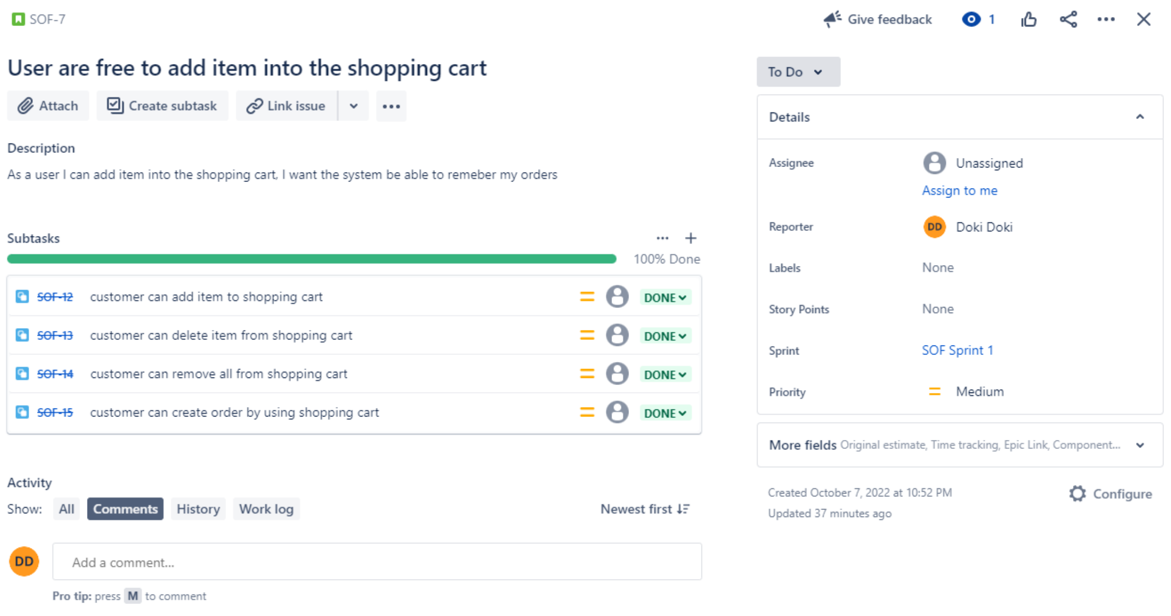
TO DO	IN PROGRESS	DONE
SOF-2 TO DO 4 sub-tasks The system is able to store commodities.		
Store the commodity into some text files SOF-26		
Able to read and write on the text files SOF-27		
The vending machine will automatically update the text file SOF-28		
The UI will generate a better viewed table depend on the database SOF-29		
SOF-4 TO DO 2 sub-tasks Put the items within each category and displayed onto the screen		
show the items in the store to customer SOF-21	create database for each category SOF-11	
SOF-8 TO DO 2 sub-tasks Optimise the UI and feedback from each operations		
Have a IMG on the main page SOF-24		

The scrum master Haolin Jin is responsible to add those substaks and user stories into the project board and assign each task to the different teammates.

2. Planning and Goals

Because we overall have three different sprints, the first sprint is mainly focused on the scrum team set up and preparing the product backlog, so we didn’t do a lot of coding during the first sprint instead we had some planning for the vending machine system and the final goals for the rest of two sprints

What we planned to do during the first sprint is to finish the basic UI frame for the user first, and then we decide to work on the item displacement and to achieve the purchase function for our vending machine system. So we created several user stories and subtasks inside of the project board, and we assigned different people to these subtasks.


SOF-7

User are free to add item into the shopping cart

Attach | Create subtask | Link issue

Description

As a user I can add item into the shopping cart. I want the system be able to remeber my orders

Subtasks

100% Done

~~SOF-12~~	customer can add item to shopping cart	DONE
~~SOF-13~~	customer can delete item from shopping cart	DONE
~~SOF-14~~	customer can remove all from shopping cart	DONE
~~SOF-15~~	customer can create order by using shopping cart	DONE

Activity

Show: All | Comments | History | Work log — Newest first

Add a comment...

Pro tip: press M to comment

Give feedback

To Do

Details

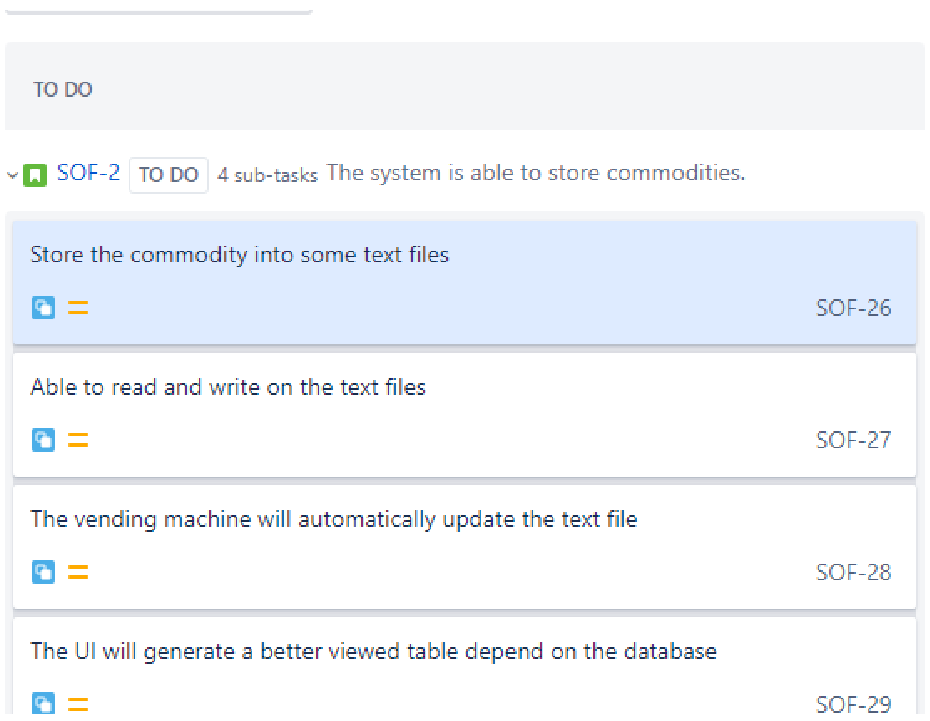
Assignee	Unassigned Assign to me
Reporter	DD Doki Doki
Labels	None
Story Points	None
Sprint	SOF Sprint 1
Priority	Medium

More fields Original estimate, Time tracking, Epic Link, Component...

Created October 7, 2022 at 10:52 PM
Updated 37 minutes ago

Configure

Just like the above screenshot, our plan is to accomplish the purchase function of the vending machine system, so we require a single shopping card to store the customer’s order, so we create a subtask and assign it to LeoYang. Then we need to consider how to store the vending machine’s item, and how to achieve the interaction between the customer and items, so we decide to store the items inside of their corresponding category file and create a class for this.


TO DO

SOF-2 TO DO 4 sub-tasks The system is able to store commodities.

Store the commodity into some text files SOF-26

Able to read and write on the text files SOF-27

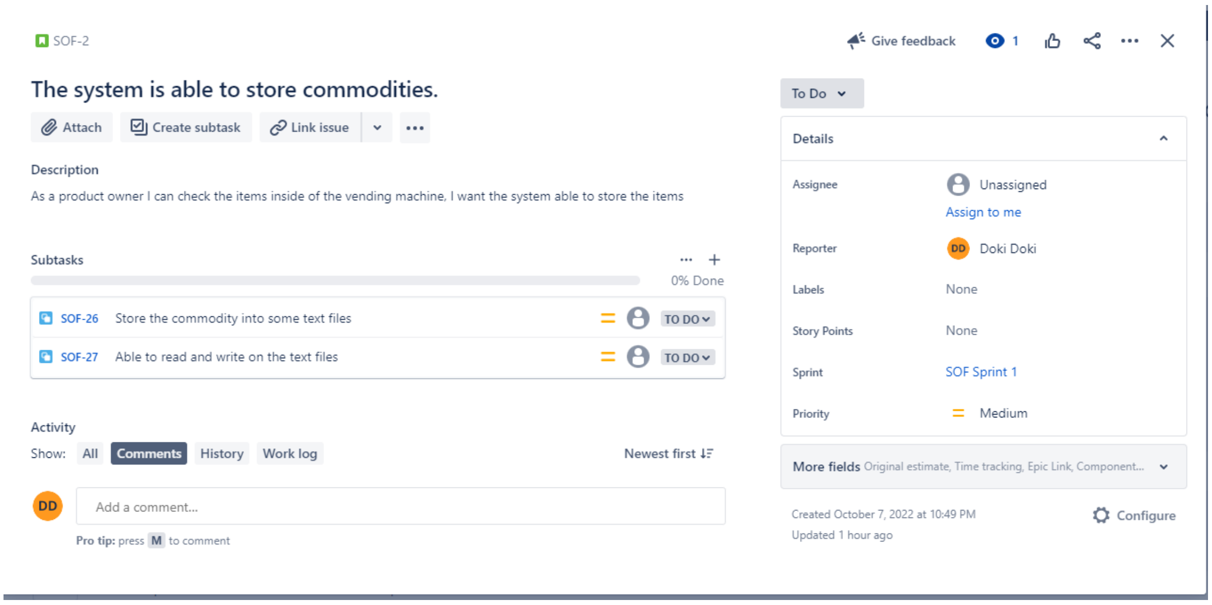
The vending machine will automatically update the text file SOF-28

The UI will generate a better viewed table depend on the database SOF-29

Then comes to the goals of our vending machine system, we decide to complete the shopping cart function, and in the future sprint the user should be able to behave in different roles like cashier and product owner. It’s not easy to achieve using java swing, so it’s necessary to have continuous testing for our code algorithms which is responsible for Zechao Sun. We also expect very frequent interaction between each teammate, and we need to prepare a good product backlog and write some useful user stories inside of our Jira software, so we can better accomplish the scrum development for this assignment.

3. User story test cases

User story one:



SOF-2

Give feedback 1

The system is able to store commodities.

Attach | Create subtask | Link issue

To Do

Description

As a product owner I can check the items inside of the vending machine, I want the system able to store the items

Subtasks

0% Done

SOF-26	Store the commodity into some text files	TO DO
SOF-27	Able to read and write on the text files	TO DO

Details

Assignee	Unassigned Assign to me
Reporter	DD Doki Doki
Labels	None
Story Points	None
Sprint	SOF Sprint 1
Priority	Medium

More fields Original estimate, Time tracking, Epic Link, Component...

Activity

Show: All Comments History Work log

Newest first

DD Add a comment...

Pro tip: press M to comment

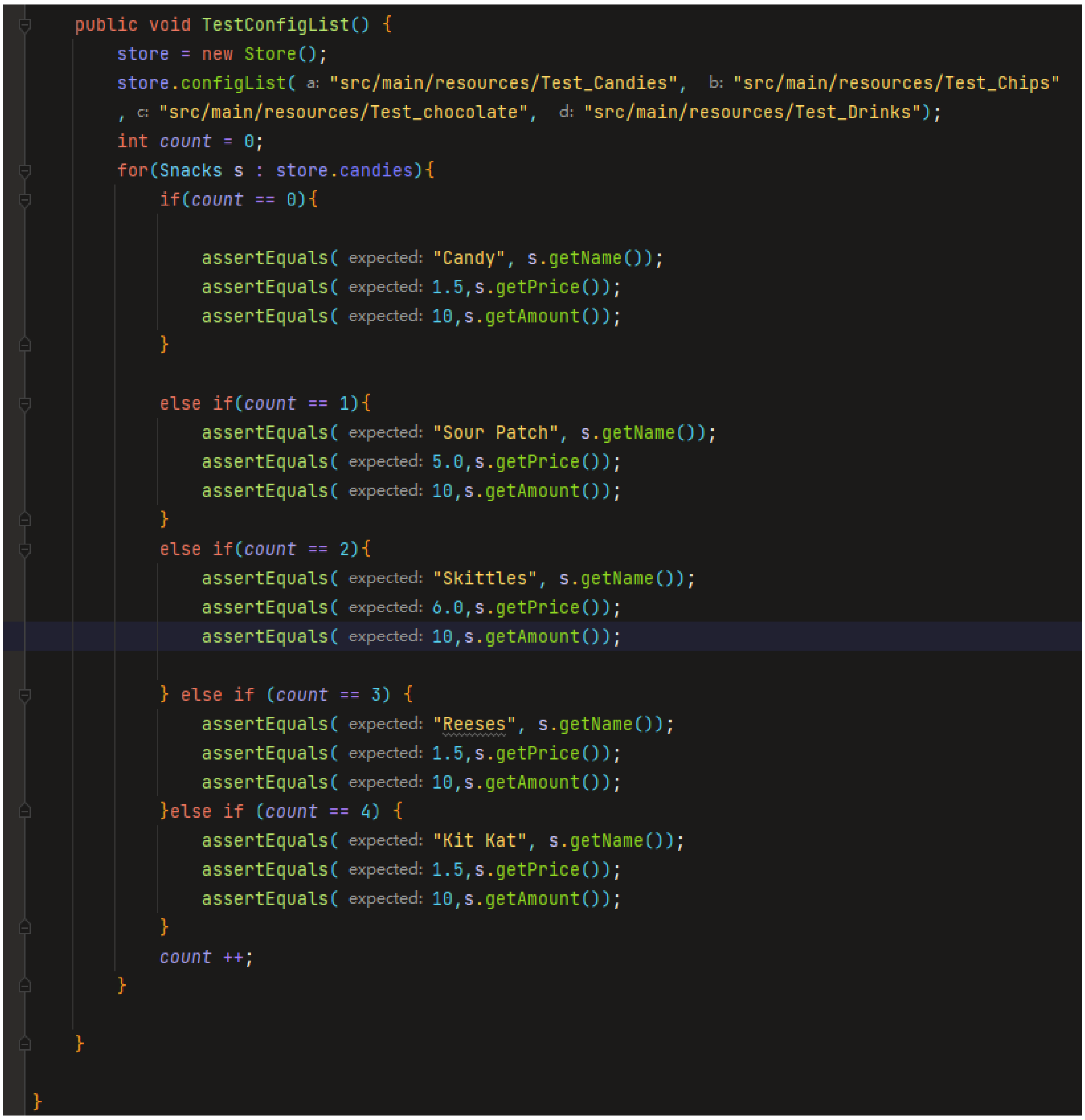
Created October 7, 2022 at 10:49 PM

Updated 1 hour ago

Configure

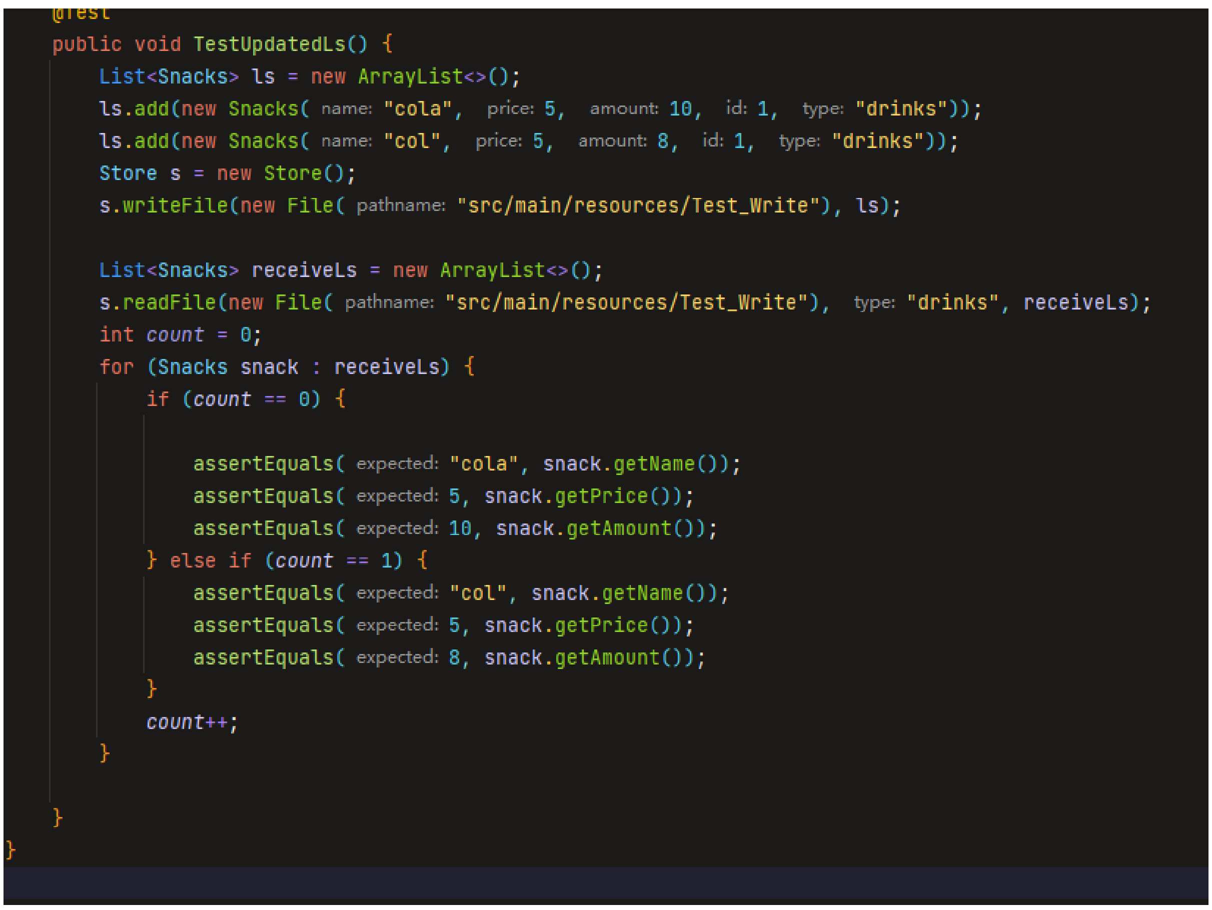
This is our first user story about the storage system of our vending machine, and there are several subtasks that also behave as the acceptance criteria. If we finish testing every subtask we can safely say we finish this user story.

Test cases for user story:



```
    public void TestConfigList() {
        store = new Store();
        store.configList( a: "src/main/resources/Test_Candies",  b: "src/main/resources/Test_Chips"
        , c: "src/main/resources/Test_chocolate",  d: "src/main/resources/Test_Drinks");
        int count = 0;
        for(Snacks s : store.candies){
            if(count == 0){

                assertEquals( expected: "Candy", s.getName());
                assertEquals( expected: 1.5,s.getPrice());
                assertEquals( expected: 10,s.getAmount());
            }

            else if(count == 1){
                assertEquals( expected: "Sour Patch", s.getName());
                assertEquals( expected: 5.0,s.getPrice());
                assertEquals( expected: 10,s.getAmount());
            }
            else if(count == 2){
                assertEquals( expected: "Skittles", s.getName());
                assertEquals( expected: 6.0,s.getPrice());
                assertEquals( expected: 10,s.getAmount());

            } else if (count == 3) {
                assertEquals( expected: "Reeses", s.getName());
                assertEquals( expected: 1.5,s.getPrice());
                assertEquals( expected: 10,s.getAmount());
            }else if (count == 4) {
                assertEquals( expected: "Kit Kat", s.getName());
                assertEquals( expected: 1.5,s.getPrice());
                assertEquals( expected: 10,s.getAmount());
            }
            count ++;
        }

    }

}
```

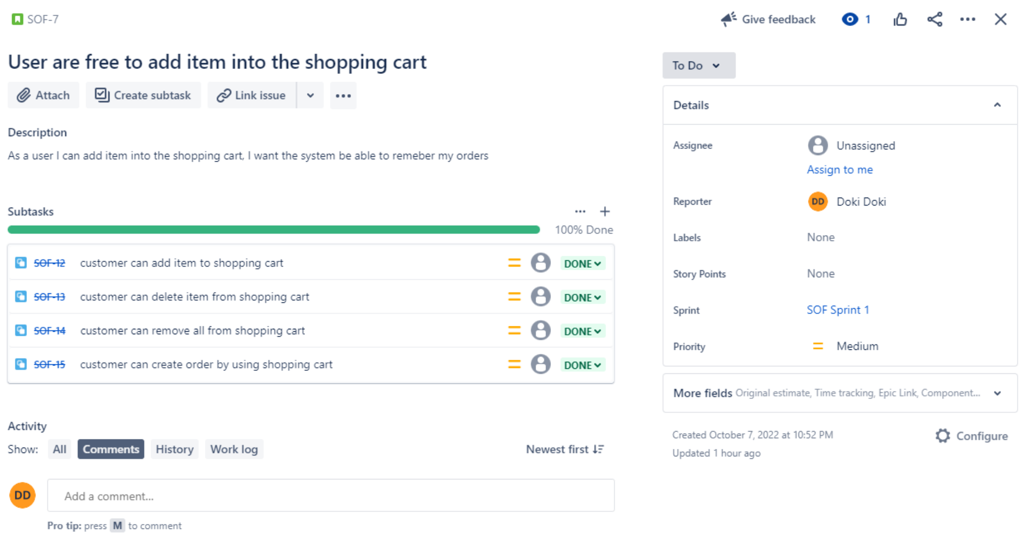
The first test case is testing the storage system of the vending machine, the test case is trying to read the file's content and put into the different category lists, and then use the junit test to confirmation the addition.



```java
    @Test
    public void TestUpdatedLs() {
        List<Snacks> ls = new ArrayList<>();
        ls.add(new Snacks( name: "cola",  price: 5,  amount: 10,  id: 1,  type: "drinks"));
        ls.add(new Snacks( name: "col",  price: 5,  amount: 8,  id: 1,  type: "drinks"));
        Store s = new Store();
        s.writeFile(new File( pathname: "src/main/resources/Test_Write"), ls);

        List<Snacks> receiveLs = new ArrayList<>();
        s.readFile(new File( pathname: "src/main/resources/Test_Write"),  type: "drinks", receiveLs);
        int count = 0;
        for (Snacks snack : receiveLs) {
            if (count == 0) {

                assertEquals( expected: "cola", snack.getName());
                assertEquals( expected: 5, snack.getPrice());
                assertEquals( expected: 10, snack.getAmount());
            } else if (count == 1) {
                assertEquals( expected: "col", snack.getName());
                assertEquals( expected: 5, snack.getPrice());
                assertEquals( expected: 8, snack.getAmount());
            }
            count++;
        }

    }
}
```

The second test case tested the read and write function of different snacks, which accomplish the second acceptance criteria.

User story two:



SOF-7

User are free to add item into the shopping cart

Attach | Create subtask | Link issue

Give feedback 1

To Do

Description

As a user I can add item into the shopping cart, I want the system be able to remeber my orders

Subtasks

100% Done

~~SOF-12~~	customer can add item to shopping cart	DONE
~~SOF-13~~	customer can delete item from shopping cart	DONE
~~SOF-14~~	customer can remove all from shopping cart	DONE
~~SOF-15~~	customer can create order by using shopping cart	DONE

Details

Assignee	Unassigned Assign to me
Reporter	DD Doki Doki
Labels	None
Story Points	None
Sprint	SOF Sprint 1
Priority	Medium

More fields Original estimate, Time tracking, Epic Link, Component...

Created October 7, 2022 at 10:52 PM
Updated 1 hour ago

Configure

Activity

Show: All | Comments | History | Work log

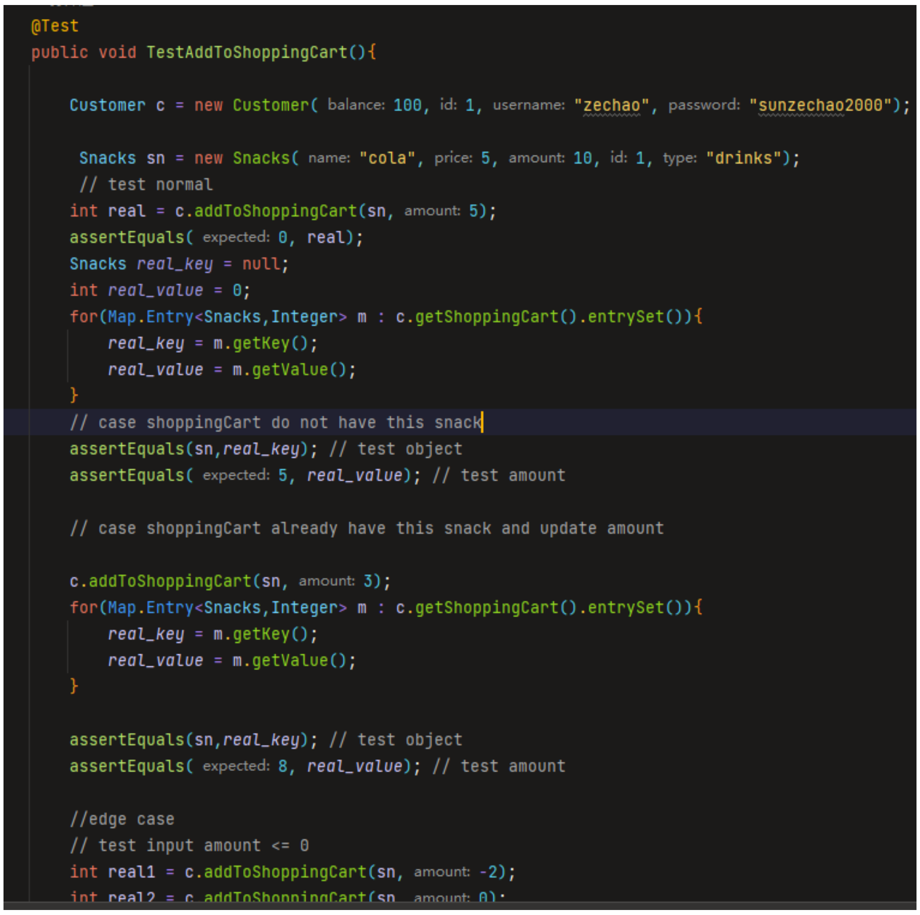
Newest first

DD Add a comment...

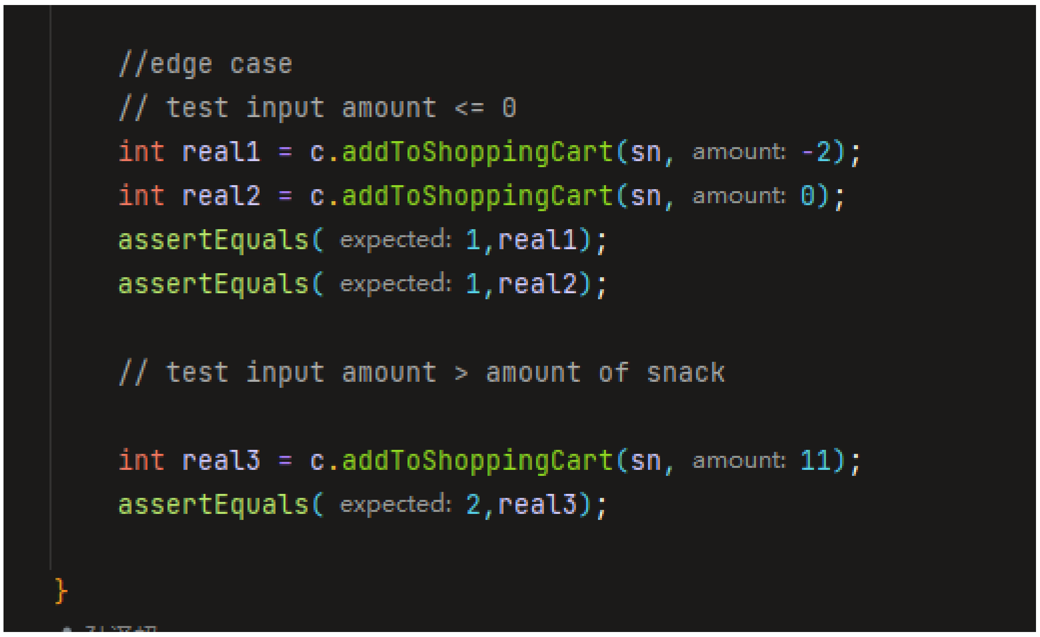
Pro tip: press M to comment

Our second user story is able to store the user's order into the shopping cart, and we have four different acceptance criteria for this.

Test cases for user story:

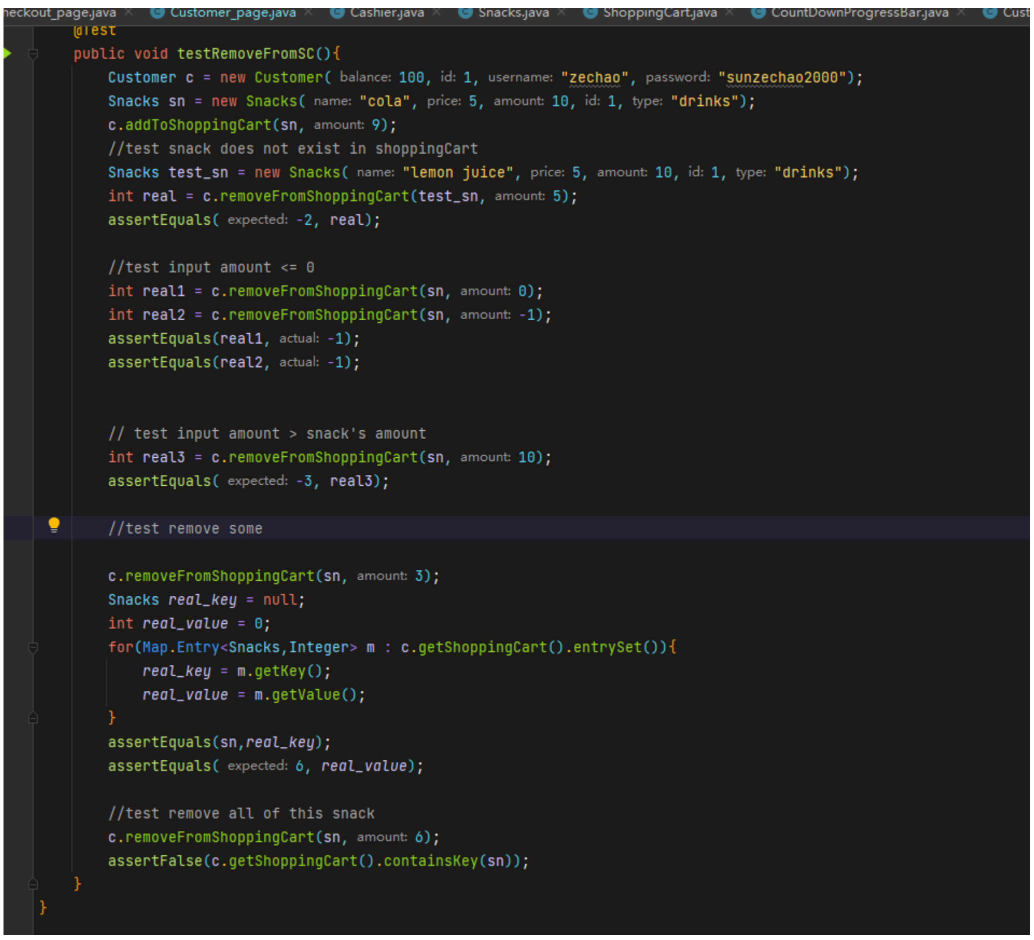


```
@Test
public void TestAddToShoppingCart(){

    Customer c = new Customer( balance: 100, id: 1, username: "zechao", password: "sunzechao2000");

     Snacks sn = new Snacks( name: "cola", price: 5, amount: 10, id: 1, type: "drinks");
     // test normal
    int real = c.addToShoppingCart(sn, amount: 5);
    assertEquals( expected: 0, real);
    Snacks real_key = null;
    int real_value = 0;
    for(Map.Entry<Snacks,Integer> m : c.getShoppingCart().entrySet()){
        real_key = m.getKey();
        real_value = m.getValue();
    }
    // case shoppingCart do not have this snack
    assertEquals(sn,real_key); // test object
    assertEquals( expected: 5, real_value); // test amount

    // case shoppingCart already have this snack and update amount

    c.addToShoppingCart(sn, amount: 3);
    for(Map.Entry<Snacks,Integer> m : c.getShoppingCart().entrySet()){
        real_key = m.getKey();
        real_value = m.getValue();
    }

    assertEquals(sn,real_key); // test object
    assertEquals( expected: 8, real_value); // test amount

    //edge case
    // test input amount <= 0
    int real1 = c.addToShoppingCart(sn, amount: -2);
    int real2 = c.addToShoppingCart(sn, amount: 0);
```



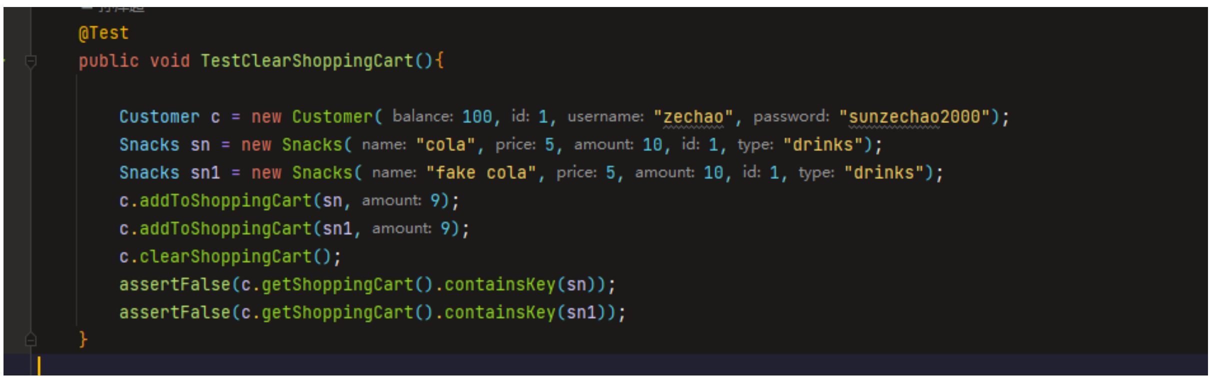
```
    //edge case
    // test input amount <= 0
    int real1 = c.addToShoppingCart(sn, amount: -2);
    int real2 = c.addToShoppingCart(sn, amount: 0);
    assertEquals( expected: 1,real1);
    assertEquals( expected: 1,real2);

    // test input amount > amount of snack

    int real3 = c.addToShoppingCart(sn, amount: 11);
    assertEquals( expected: 2,real3);

}
```

This test case is used to check whether the first criteria is achieved or not. The test case will try to assert the amount of items inside of the shopping cart and judge whether we add successfully.



```java
    @Test
    public void testRemoveFromSC(){
        Customer c = new Customer( balance: 100, id: 1, username: "zechao", password: "sunzechao2000");
        Snacks sn = new Snacks( name: "cola", price: 5, amount: 10, id: 1, type: "drinks");
        c.addToShoppingCart(sn, amount: 9);
        //test snack does not exist in shoppingCart
        Snacks test_sn = new Snacks( name: "lemon juice", price: 5, amount: 10, id: 1, type: "drinks");
        int real = c.removeFromShoppingCart(test_sn, amount: 5);
        assertEquals( expected: -2, real);

        //test input amount <= 0
        int real1 = c.removeFromShoppingCart(sn, amount: 0);
        int real2 = c.removeFromShoppingCart(sn, amount: -1);
        assertEquals(real1, actual: -1);
        assertEquals(real2, actual: -1);


        // test input amount > snack's amount
        int real3 = c.removeFromShoppingCart(sn, amount: 10);
        assertEquals( expected: -3, real3);

        //test remove some

        c.removeFromShoppingCart(sn, amount: 3);
        Snacks real_key = null;
        int real_value = 0;
        for(Map.Entry<Snacks,Integer> m : c.getShoppingCart().entrySet()){
            real_key = m.getKey();
            real_value = m.getValue();
        }
        assertEquals(sn,real_key);
        assertEquals( expected: 6, real_value);

        //test remove all of this snack
        c.removeFromShoppingCart(sn, amount: 6);
        assertFalse(c.getShoppingCart().containsKey(sn));
    }
}
```

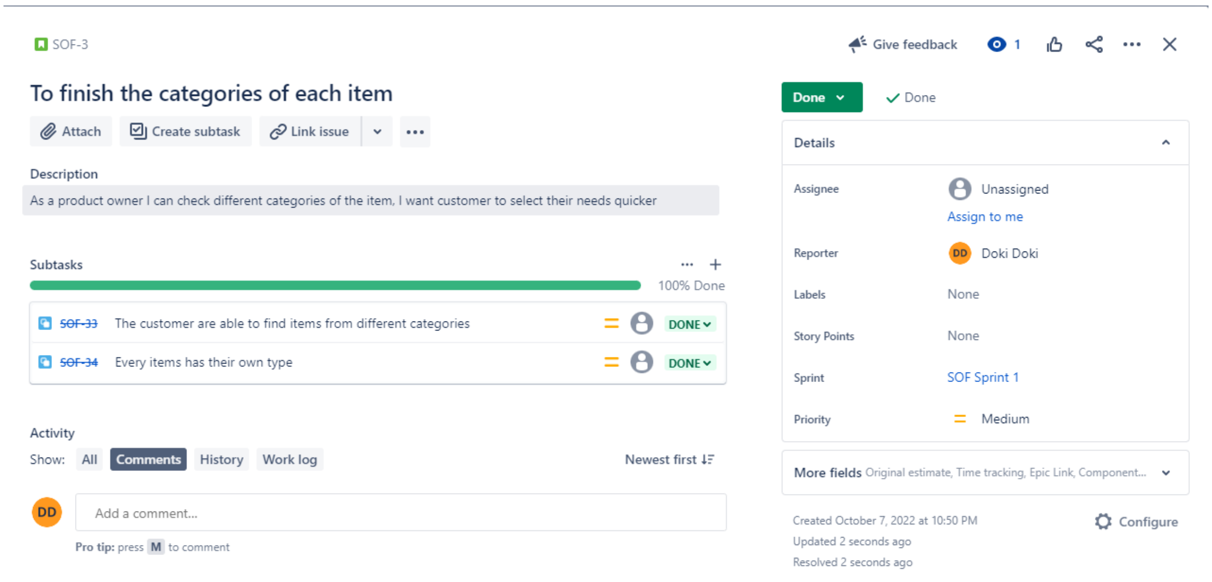
This test is testing the remove function of the system.



```java
    @Test
    public void TestClearShoppingCart(){

        Customer c = new Customer( balance: 100, id: 1, username: "zechao", password: "sunzechao2000");
        Snacks sn = new Snacks( name: "cola", price: 5, amount: 10, id: 1, type: "drinks");
        Snacks sn1 = new Snacks( name: "fake cola", price: 5, amount: 10, id: 1, type: "drinks");
        c.addToShoppingCart(sn, amount: 9);
        c.addToShoppingCart(sn1, amount: 9);
        c.clearShoppingCart();
        assertFalse(c.getShoppingCart().containsKey(sn));
        assertFalse(c.getShoppingCart().containsKey(sn1));
    }
```

The third method is responsible for the remove all function of the shopping cart which is the third acceptance criteria of this user story.

User story three:



SOF-3

Give feedback 1

To finish the categories of each item

Attach | Create subtask | Link issue | ...

Done | ✓ Done

Description

As a product owner I can check different categories of the item, I want customer to select their needs quicker

Subtasks

100% Done

~~SOF-33~~	The customer are able to find items from different categories	DONE
~~SOF-34~~	Every items has their own type	DONE

Details

Assignee	Unassigned Assign to me
Reporter	Doki Doki
Labels	None
Story Points	None
Sprint	SOF Sprint 1
Priority	Medium

More fields Original estimate, Time tracking, Epic Link, Component...

Activity

Show: All | Comments | History | Work log   Newest first

Add a comment...

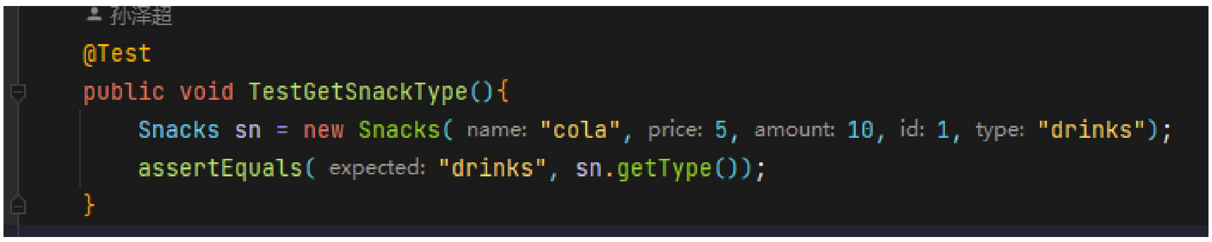
Pro tip: press M to comment

Created October 7, 2022 at 10:50 PM
Updated 2 seconds ago
Resolved 2 seconds ago

Configure

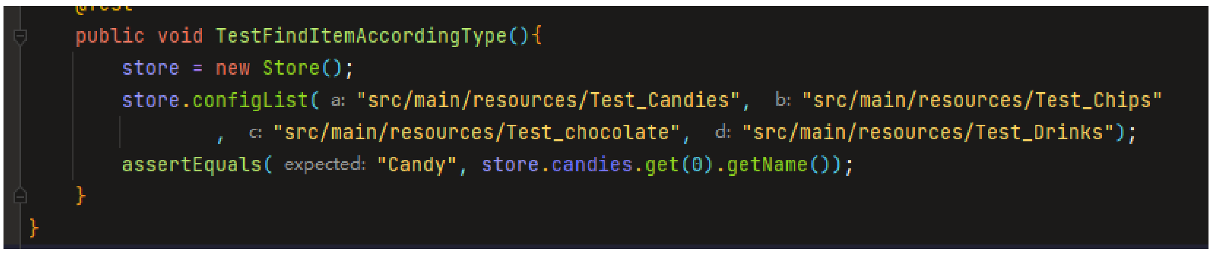
Our third user story is that the user can see the items from different categories, and the other one is that the user is able to view the item's type.

Test cases for user story:



```
孙泽超
@Test
public void TestGetSnackType(){
    Snacks sn = new Snacks( name: "cola", price: 5, amount: 10, id: 1, type: "drinks");
    assertEquals( expected: "drinks", sn.getType());
}
```

This test case is testing if the type of the item is correctly setted, then we can confirm that the user can identify the category of that item.



```
    public void TestFindItemAccordingType(){
        store = new Store();
        store.configList( a: "src/main/resources/Test_Candies",  b: "src/main/resources/Test_Chips"
                , c: "src/main/resources/Test_chocolate",  d: "src/main/resources/Test_Drinks");
        assertEquals( expected: "Candy", store.candies.get(0).getName());
    }
}
```

This test case the system will read the text file and set the store items and then the user can find the item’s name by their type, then this user story is finished and tested by test cases.

4. Client feedback

We received several feedback during the first sprint demonstration, because in the first sprint we haven't done a lot of functionalities so the feedback is generally about the test cases and the scrum method.

- Clearly state the acceptance criteria for each user story.

For each user story we didn't have a clear acceptance criteria before, and we should use this list to check whether we can say we complete this user story.

- Different user stories should have their own test cases.

In our current test cases we didn't have a specific user story which related to it, so we made some changes for this part then every acceptance criterias of the user story will have a test case, then we can prove we complete the user story.