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	Jin JJL
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	

Level 2, SIT Building, J12
The University of Sydney
NSW 2006 Australia

T +61 2 9351 3423
F +61 2 9351 3838
E sit.info@sydney.edu.au
sydney.edu.au/it

ABN 15 211 513 464
CRICOS 00026A

SOFT2412 Assignment2 Sprint3

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

1. Introduction and Scrum group collaboration

In the third sprint, because this is the last sprint, we need to make sure that this assignment is fully completed after this sprint, so we first need to confirm our scrum role which is similar as before.

We have made the same role distribution as the previous two sprints.
1.Xiaoqian Hu is the core team member who is responsible for the project interface development.
2.Yiheng Yang is the core team member who is responsible for the function realization and structure design.
3.Haolin Jin is the scrum master who is responsible for the scrum backlog and each sprint report.
4.Zechao Sun is the product owner who is responsible for the communication between client and the other scrum members, also the code testing part.

Haolin Jin arranged the meeting for the third sprint and our first meeting started on 29 October, we discussed the user stories and main goals for the third sprint. Because this is the last sprint so we need to sort out the user stories we have completed so far and unfinished requirements. After that, we determined the general direction and decided to improve the feedback shown in the last project. First, we added a login system to our software and then added their own login accounts for different roles. In this way, our remaining needs are not much, only the functions of seller and product owner are left.

Our second meeting started on 30 October.We completed the previous task in one day, and then we decided to summarize and discuss our next division of work and the development direction in this meeting. Our current software can provide user login and anonymous purchases, but it still cannot display the most recent purchases. So we decided to leave this requirement first and prioritize ensuring the usability of our software instead of focusing on expanding our functionality, because next Thursday is the demonstration of the project we need to make sure there are no bugs and flaws. So the main purpose of our meeting this time is

to combine our unit test and integration test with the CI/CD pipeline to ensure that our product can pass all test cases.

Our third meeting was on 31 October. We spent a day making sure our current software passed all test cases and requirements, and then we should start working with the rest of the functionality. For the third meeting, we mainly discussed the division of work for the remaining functions. Xiaoqian Hu was responsible for the seller's final report, Yiheng Yang was responsible for the realization of some functions of the product owner, and then the remaining team members were responsible for testing and ensuring that the reason for each cancellation of the order was recorded and written into a new file.

Our fourth meeting was on 2/November. Before starting this meeting, we have completed all the functions and implementation of the vending machine. Next, we will organize all the content to be demonstrated, as well as the report. So we compared the content of the user story in the meeting to make sure that we implemented all the requirements, and then sorted out our github README file, including the running of Jenkins. Finally we sorted out the rough sequence of the demonstration and then proceeded with the final project report.

GithubLink: SOT2412-COMP9412-2022S2/T22_G03_Assignment2 (sydney.edu.au)

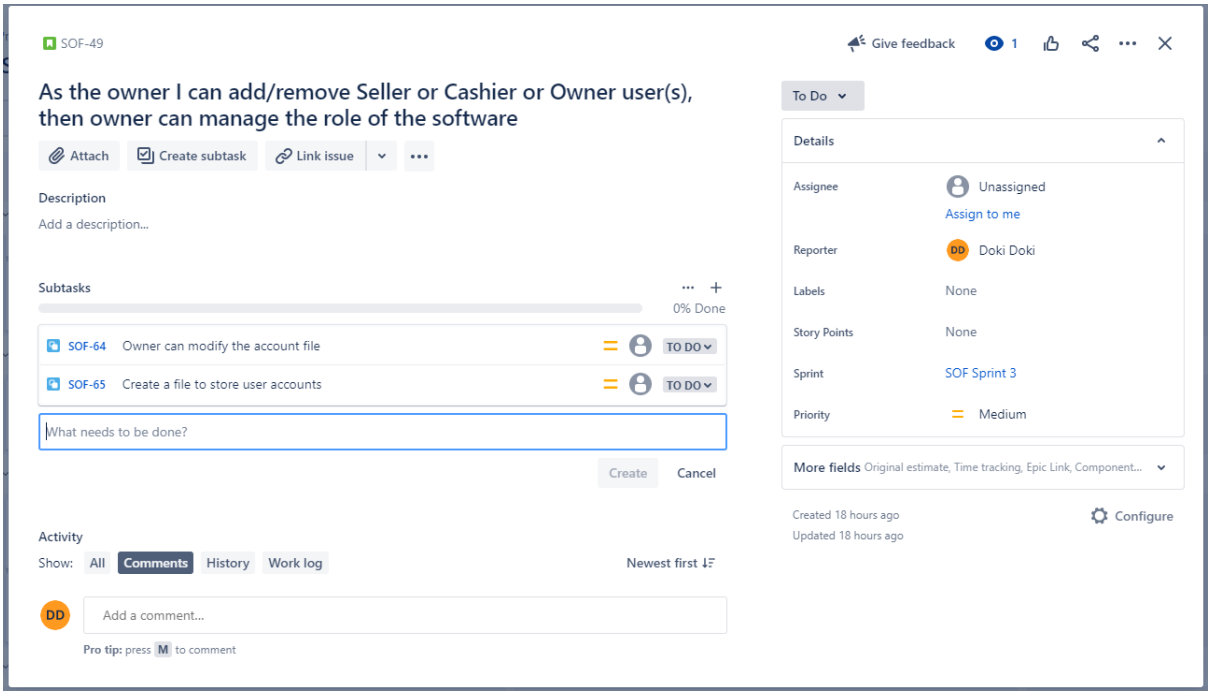
Jira Link:SOF board - Agile Board - Jira (atlassian.net)
(I will put the screenshot of the project board down below)

2. Planning and Goals

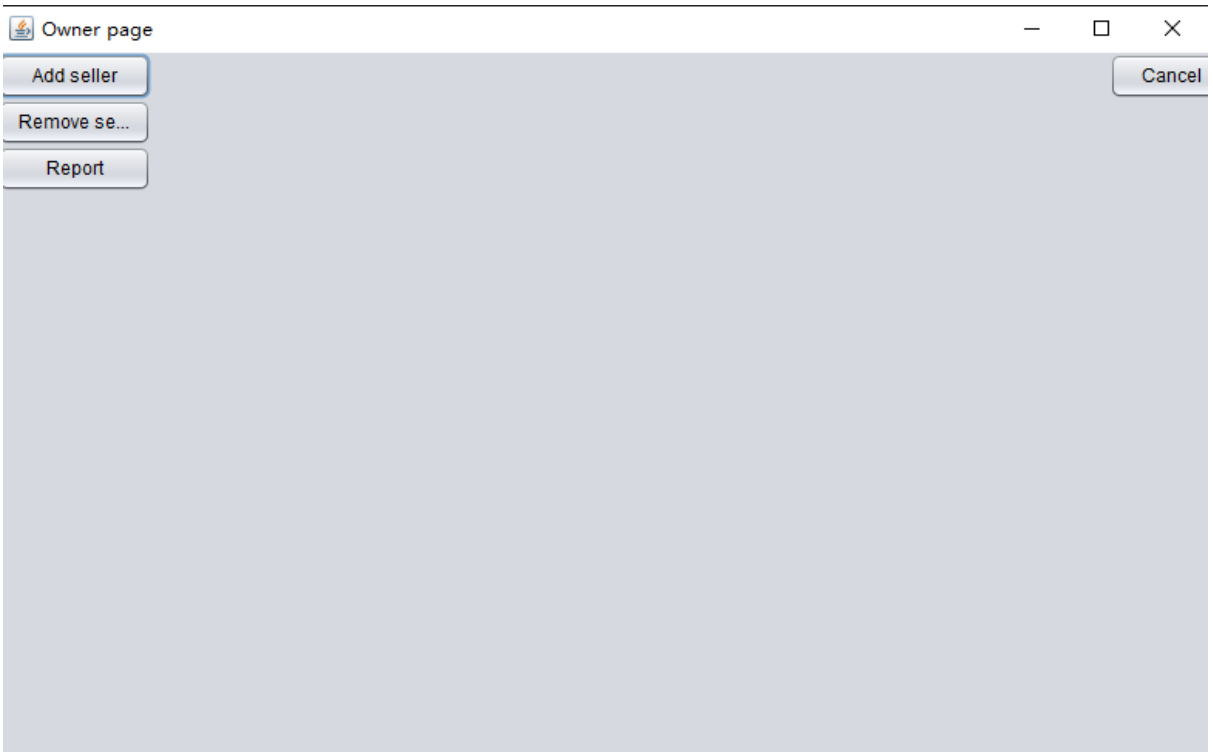
Because this sprint is the last sprint, we must make sure that we can finish every assignment requirement after this final sprint. There are basically two main parts for our work, the first part is to complete the functionality of the owner, and the second part is to catch up the previous missing function for seller and login page.

- **First user story**

The above plan is too broad, because there are so many functions that the product owner needs to implement, and even the account of each role can be controlled by the product owner, so we need to plan in more detail how to implement the product owner's function. In the requirements of the project, we know that the product owner needs to be able to change or add different roles in the project, so we need to have a separate file to store each account and its corresponding role. The product owner should have a separate page to change this file or add new accounts and their corresponding roles, so we wrote this requirement as a user story and added it to our third sprint, which will also be a part of our third sprint.



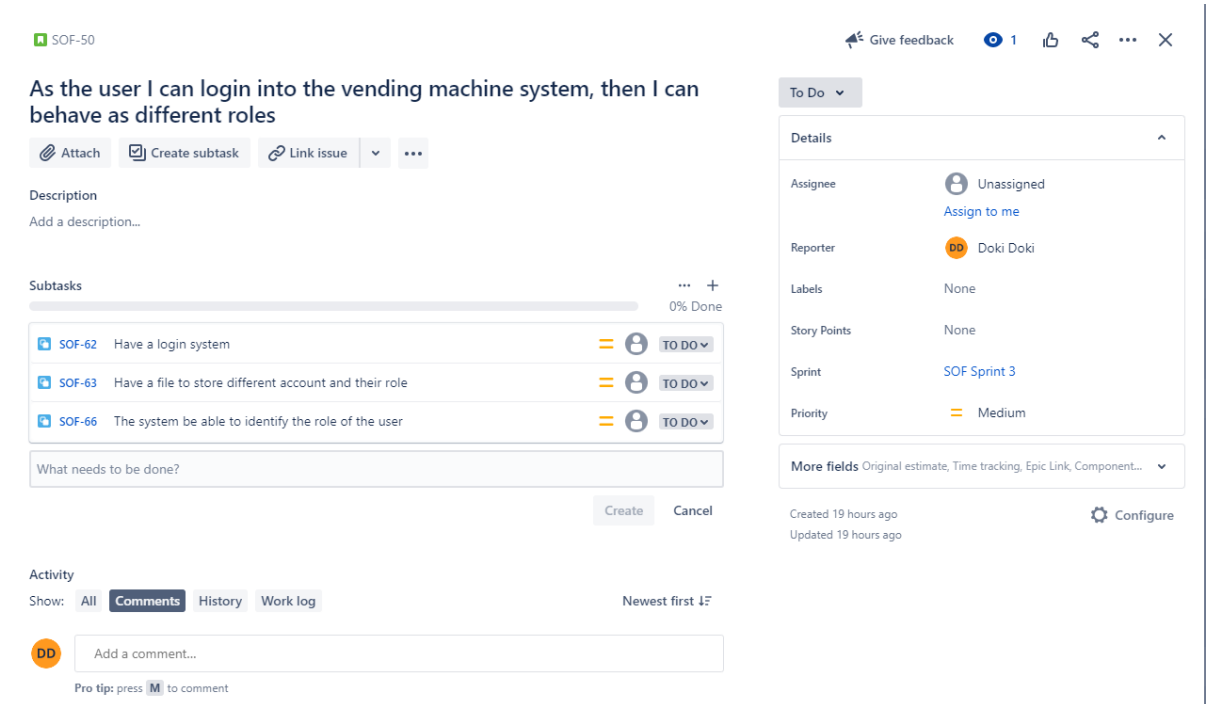
Just like the image above, this is one of the user stories of the third sprint. The product owner needs to be able to delete and add different roles other than himself, so his first task is to add a file to store account information of different roles. The second subtask is that the product owner needs to be able to access and change this file. For the specific implementation, we need to have a separate page for the product owner because we need to combine the user interface, and there needs to be a button on it. The product owner can use this button to change the content of the file.



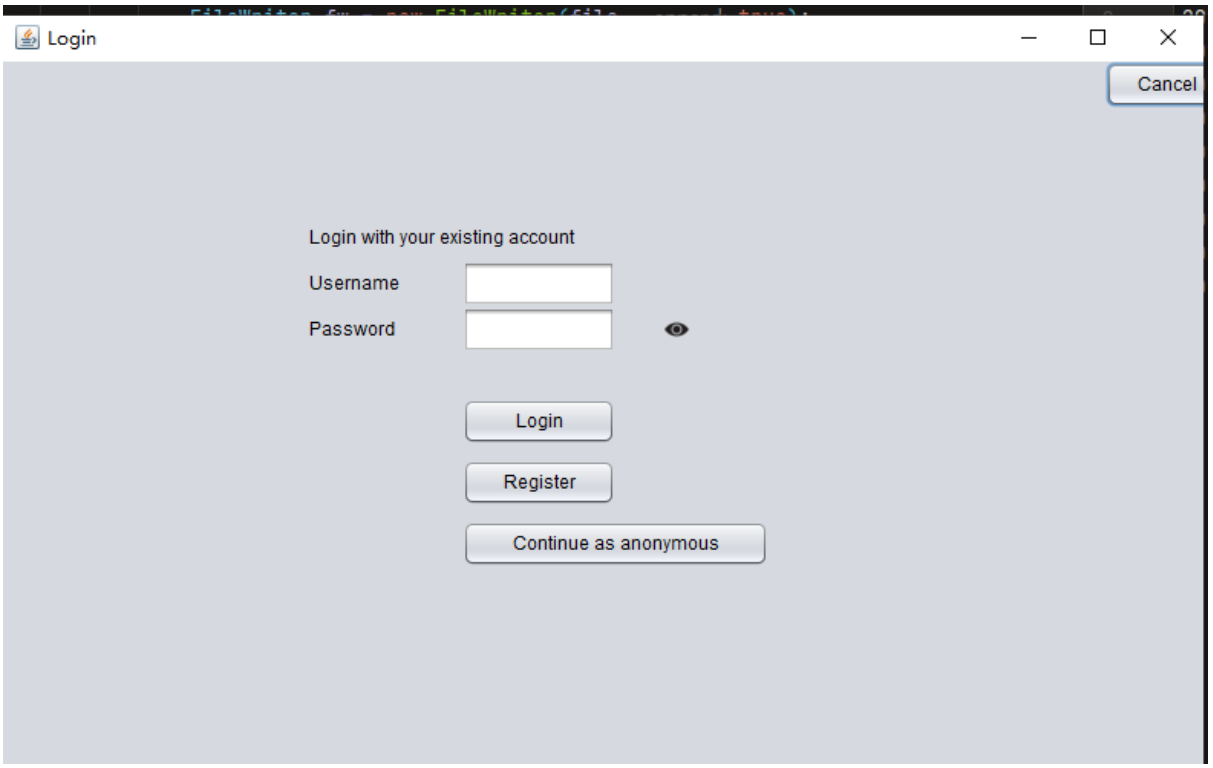
So when the product owner clicks on “Add seller” or “Remove seller” he will be able to change the account file and free to add another user into the vending machine. This work is assigned to Yiheng Yang and should be completed before the last meeting.

- **Second user story**

To implement the user story above, we first need to make sure that our program supports users to log in with different roles, so we need a file to save these contents.

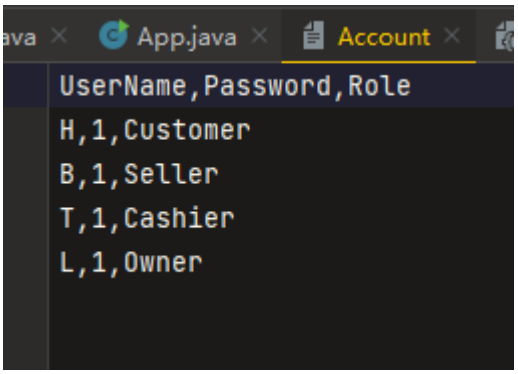


There are three different subtasks for this user story. The software needs to have a login function, so we need to make some changes to the default page, and we need to add a login button. Users can use this button to log in or register an account.



Just like the image above, if the user presses the "LOGIN" button, they will go to this page. The user can choose to log in with an existing account or register a new account, and can also purchase products as

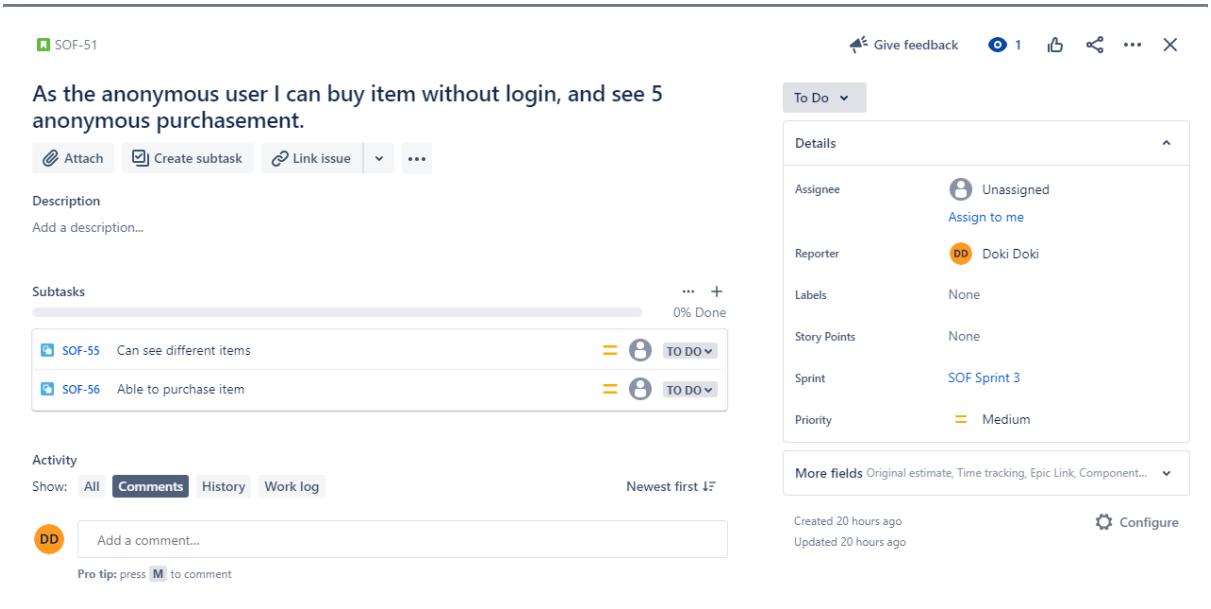
anonymous.The remaining two subtasks require developers to read and write files in the code, and then store the information in the Customer class. At this time, if the user has purchased some products and is ready to checkout, we need to pass in the customer parameter and then the system It will determine what role the purchaser plays and then store their corresponding user name in the purchase record file. So we need a new file to store each user's information. Haolin Jin responsible for the third task, and Yiheng Yang and Xiaoqian Hu responsible for the first two tasks.



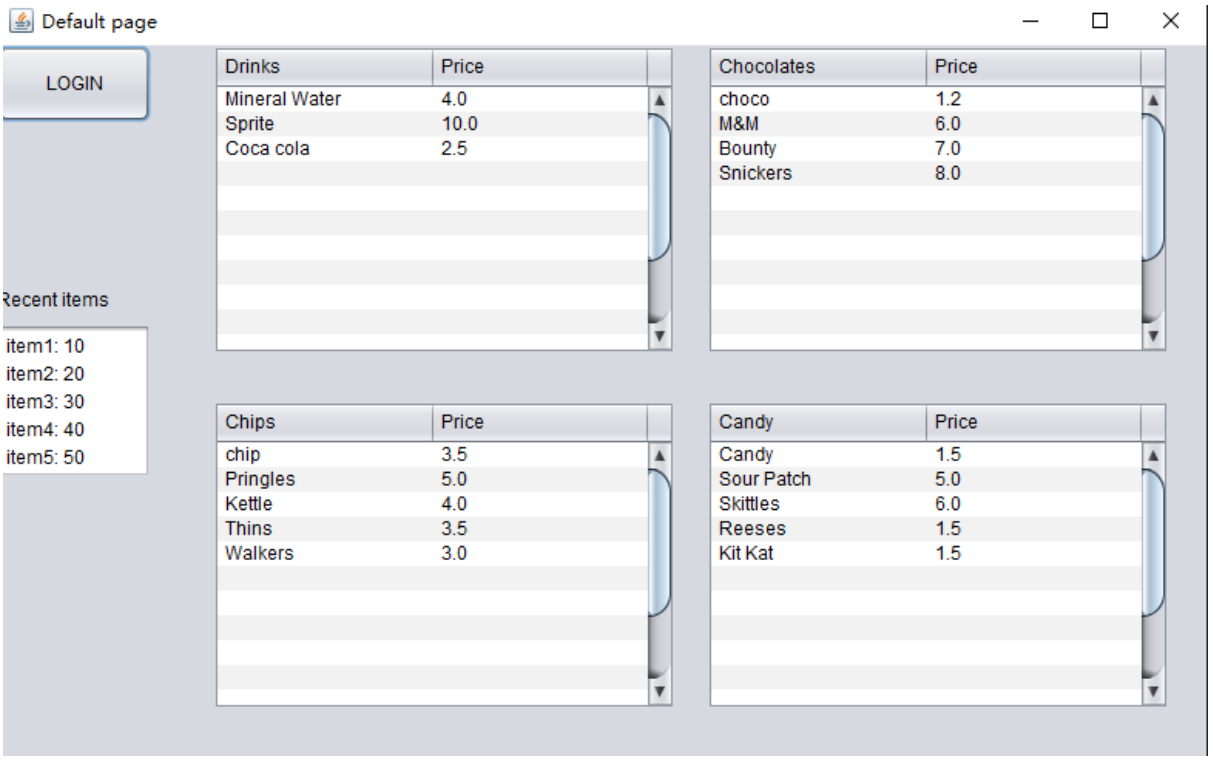
(This just a template of the Account file)

- **Third user story**

The third user story is that users can buy products without logging in, and can also see the products purchased by other anonymous users.



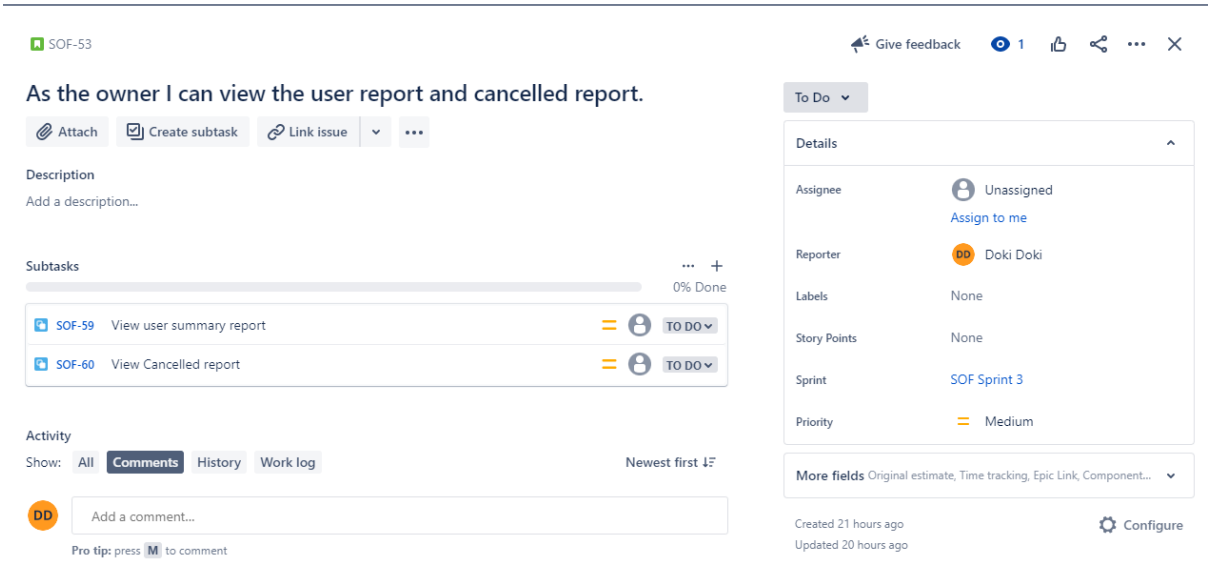
In this user story, different subtasks need to be implemented separately. The first subtask needs to display different products in the initial interface, because in the second sprint, we can only see the information of each product in the purchase of products, but after joining the login system, anonymous users should also be able to see all products , so we need to change our initial interface to allow anonymous users to see and buy items. We assigned the first subtask to Xiaoqian Hu to write, just like the picture below, users can now see different products on the initial page.



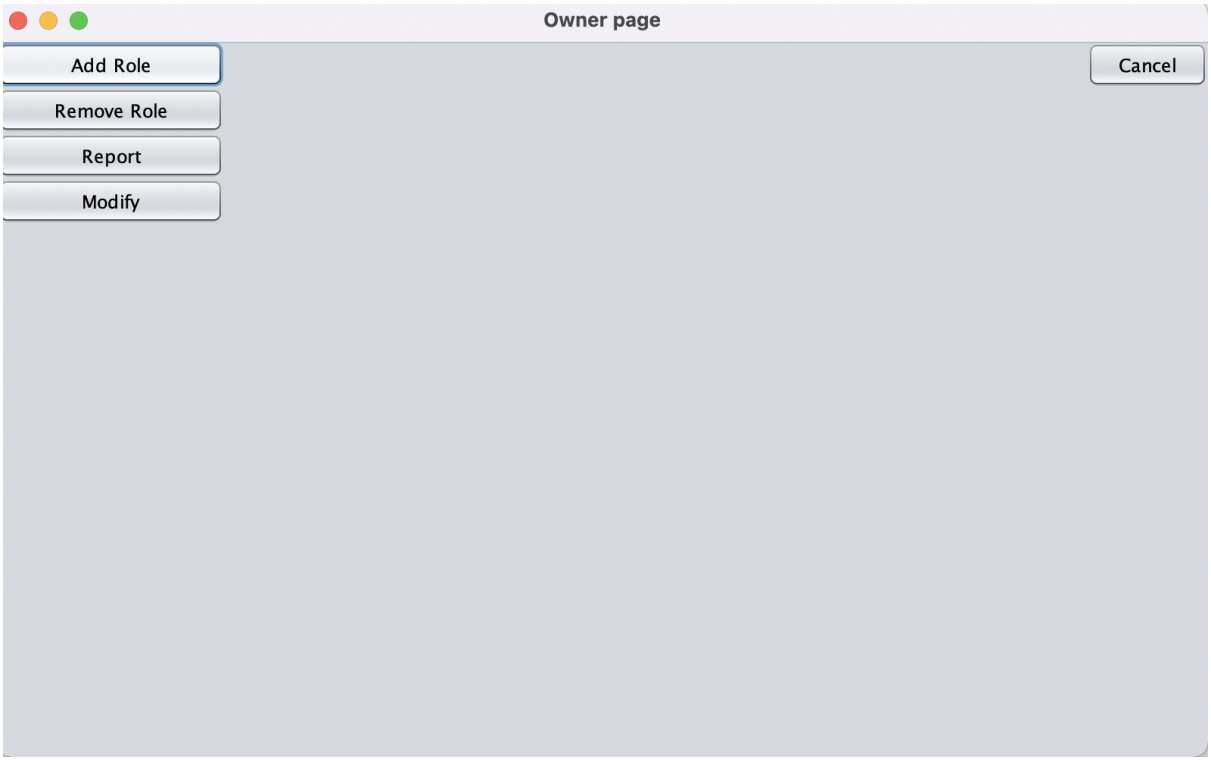
For the second task, we are not yet completed, so the above image just shows a template of the recently purchased five items on the left hand side. Because there is a file in our system that stores the purchase records of users, we plan to take the first five items of this file as the records of recent purchases. Because the records seen by anonymous users and normal users are different, we also need to decide whether the username of the current user is matched, and then put it on the UI.

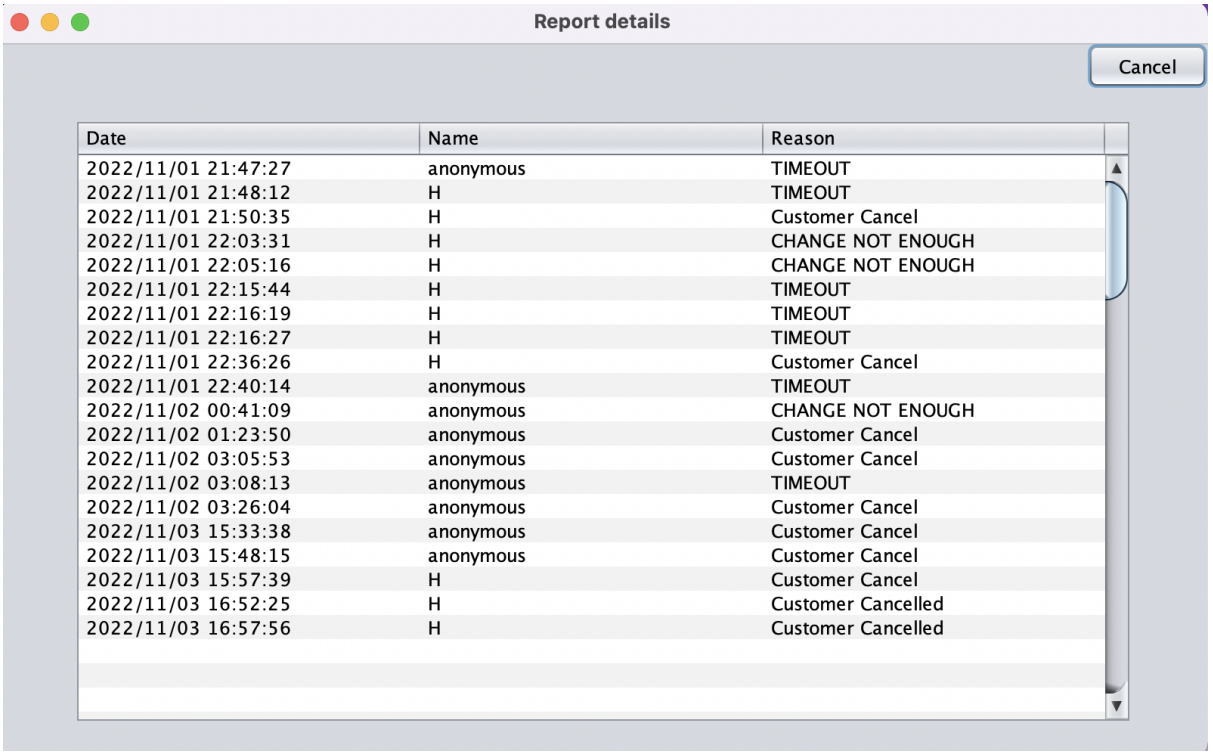
- **Fourth user story**

Another function of the project owner is to be able to see the report of all user information and the report of canceled purchase, so I divided these two requirements into two subtasks and assigned them to different team members to complete.



The first subtask is relatively easy to implement. The developer only needs to add a button and the project owner can access the information of the stored account through this button, and a table will be formed to show the specific member name and his corresponding role.





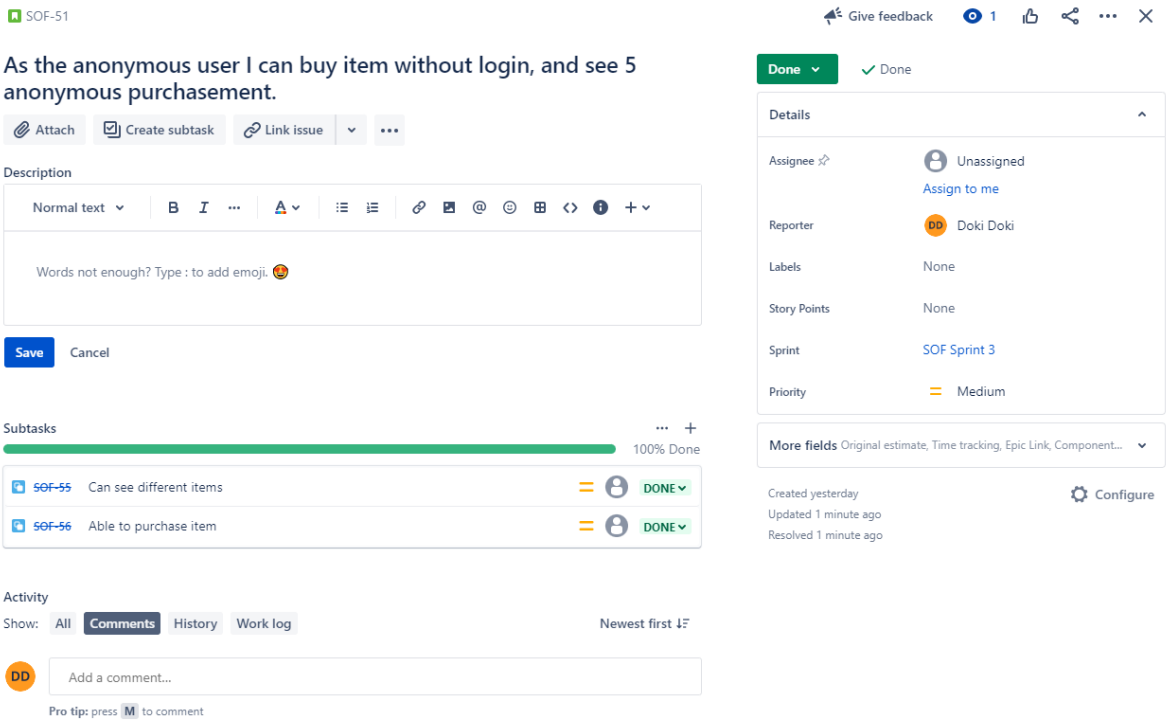
The second subtask requires the developer to add some new functions to the purchase page. There are many reasons for the user's failure to purchase. The user can actively cancel the purchase, it can also automatically exit because it exceeds 120 seconds, and because the vending machine does not have enough money. Therefore, developers need to consider a lot of situations, so it is relatively difficult to implement, which is why product owner is the last functional role to be implemented. So to sum up, every time you cancel an order, you need to record the time, the purchased item and the reason for the purchase failure. We plan to store these contents in a file. Whenever the product owner wants to access this information, the file will be recorded. The reading and writing are written into a table and then displayed through the UI.

- Conclusion

Because the report of the third sprint and the implementation of our code are happening at the same time, the above image is not the final result of our product. Before the final demonstration, we will rearrange the content of the report, which also includes a summary of client feedback.

3. User story test cases

User story one:



Test cases for user story:

```
孙泽超
@Test
public void recentFiveItems(){
    List<String[]> result = Items.show( username: "anonymous", path: "src/main/resources/TestTransactionHist");
    String[] expect1 = {"1","Mineral Water"};
    assertArrayEquals(expect1,result.get(0));

}
```

This test case test if can read TestTransactionHist file successfully, the function of this test is to make sure our software can show recentFiveItems ,our test result is can read this file successfully and can show recent five items on the screen.

```
孙泽超
@Test
public void TestCheckCard(){
    GetCardInfo getCardInfo = new GetCardInfo();
    getCardInfo.parse( path: "src/main/resources/credit_cards.json");
    Customer c = new Customer( balance: 100000, id: 1, username: "H", password: "sss");
    boolean real = c.checkCard( name: "Charles", number: "40691");
    assertTrue(real);
    boolean real2 = c.checkCard( name: "Charles", number: "40692");
    assertFalse(real2);

}
```

This test case tests if a customer can purchase an item successfully by card, this edge test case will read our card json file, and check if the input username and input password are in the file. If not in file, return false.Our test result is that the check card function is correct, and can get correct card information from the customer file.

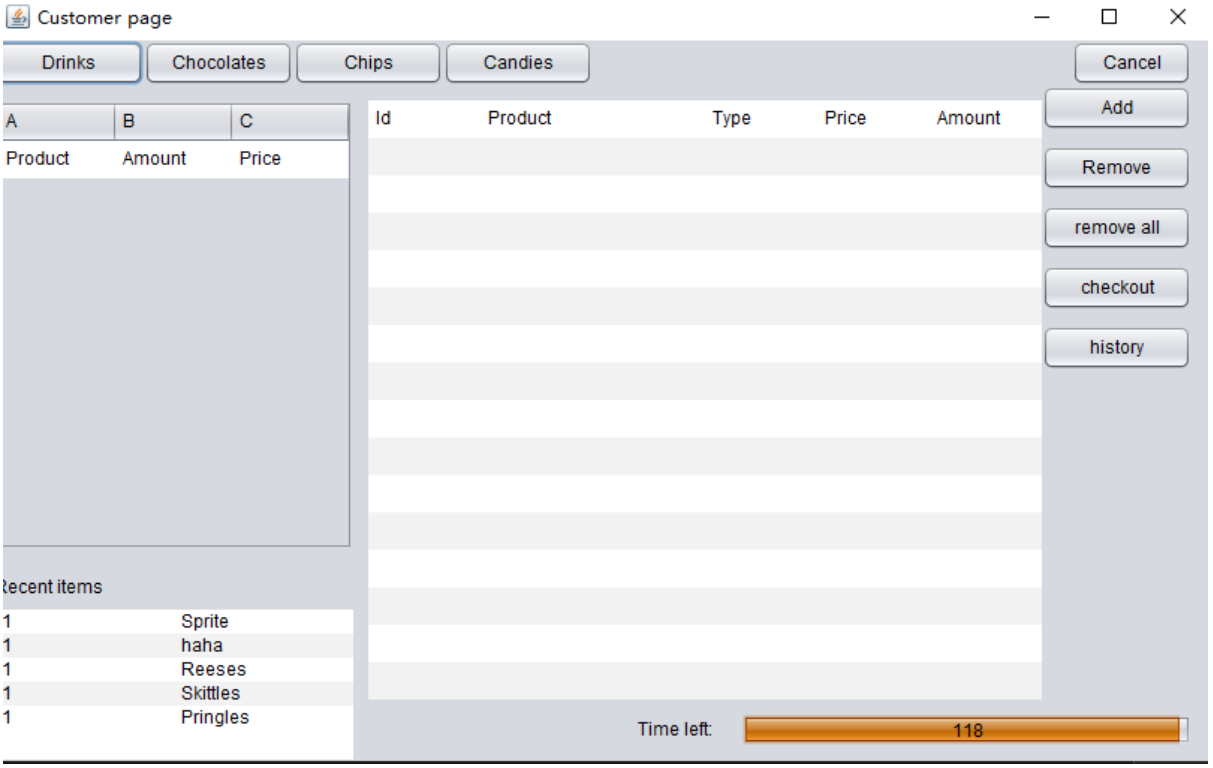
GUI display:

When the customer wants a checkout item and the customer chooses payment using a card, the system needs to check whether the card information is correct.

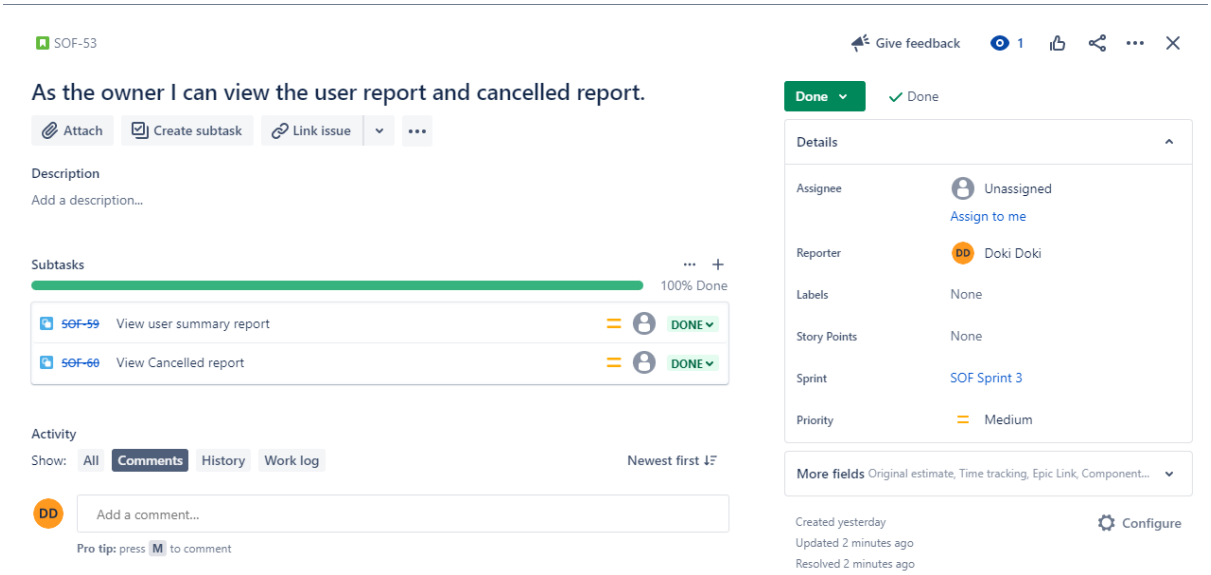
```
孙泽超
@Test
public void TestGetCardInfo(){
    GetCardInfo getCardInfo = new GetCardInfo();
    getCardInfo.parse( path: "src/main/resources/credit_cards.json");
    Customer c = new Customer( balance: 100000, id: 1, username: "Q", password: "sss");
    String[] result = c.getCardInfo( path: "src/main/resources/CustomerInfo");
    String[] expect = {"Patricia","30690"};
    assertArrayEquals(expect,result);
    Customer c1 = new Customer( balance: 100000, id: 1, username: "l", password: "sss");
    String[] result1 = c1.getCardInfo( path: "src/main/resources/CustomerInfo");
    assertNull(result1);
}
```

This test case tests whether our system can get card information from the customer file correctly. It will read the customer file and get card information about the customer by customer name . Our test result can get the card information correctly from the file. There is an edge case where the input username does not exist in the CustomerInfo file. Return null if does not exist.

GUI display:


Customer page
Drinks
Chocolates
Chips
Candies
Cancel
A
B
C
Product
Amount
Price
Id
Product
Type
Price
Amount
Add
Remove
remove all
checkout
history
ecent items
1 Sprite
1 haha
1 Reeses
1 Skittles
1 Pringles
Time left:
118

User story two:

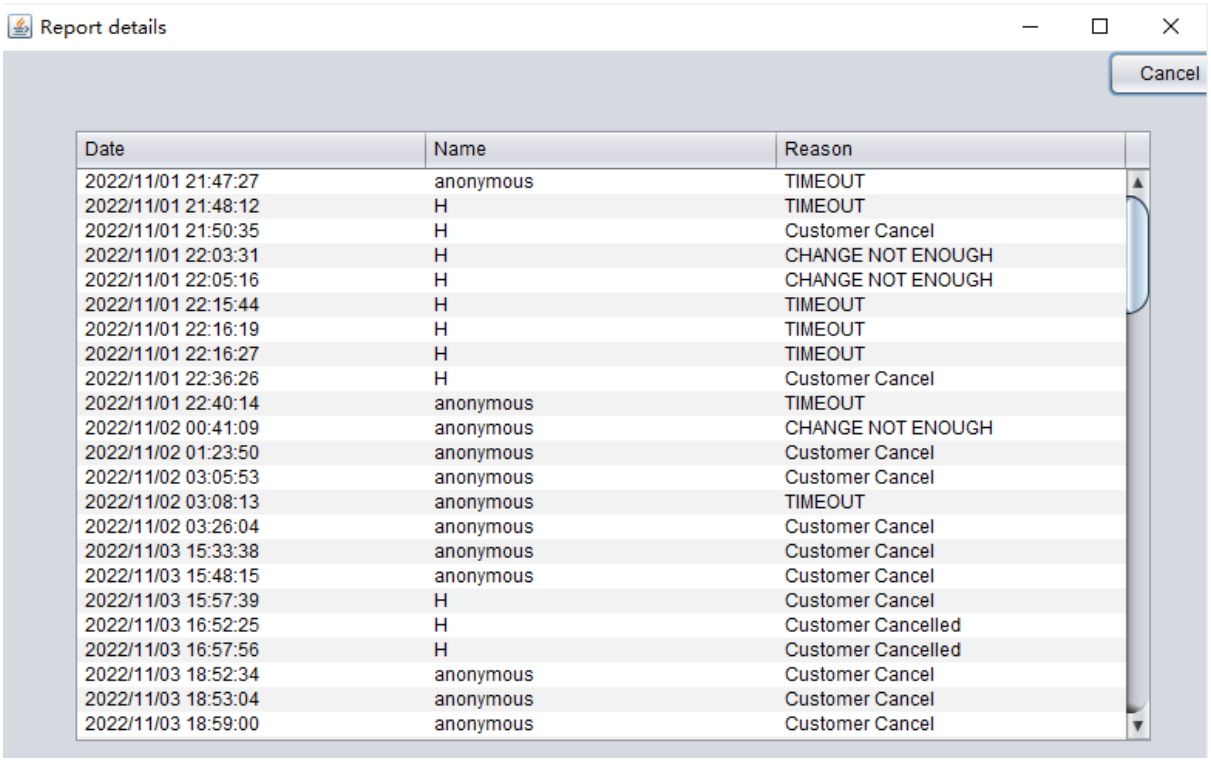


Test cases for user story:

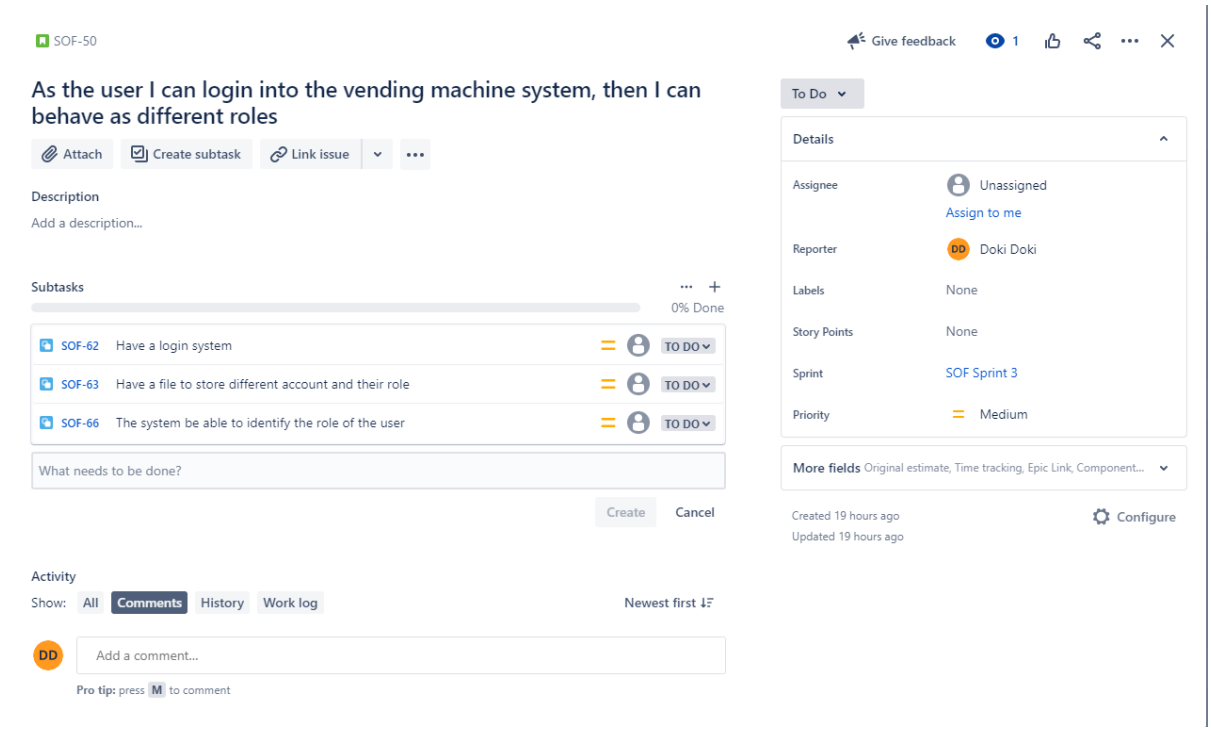
```
孙泽超
@Test
public void TestViewCancelReport(){
    Owner O = new Owner( username: "H", password: "1");

    String[] s1 = {"2022/11/01 21:47:27","anonymous","TIMEOUT"};
    String[] s2 = {"2022/11/01 21:48:12","H","TIMEOUT"};

    List<String[]> result = new ArrayList<>(O.getCanceled( path: "src/main/resources/TestCancel"));


    assertArrayEquals(s1,result.get(0));
    assertArrayEquals(s2,result.get(1));

}
```

This test case tests if our system can read the cancel file and view the cancel Report. The test result is that our system can read the cancel file correctly, and also can show a Cancel report in the front end.

GUI Display:



User story three:



Test cases for user story:

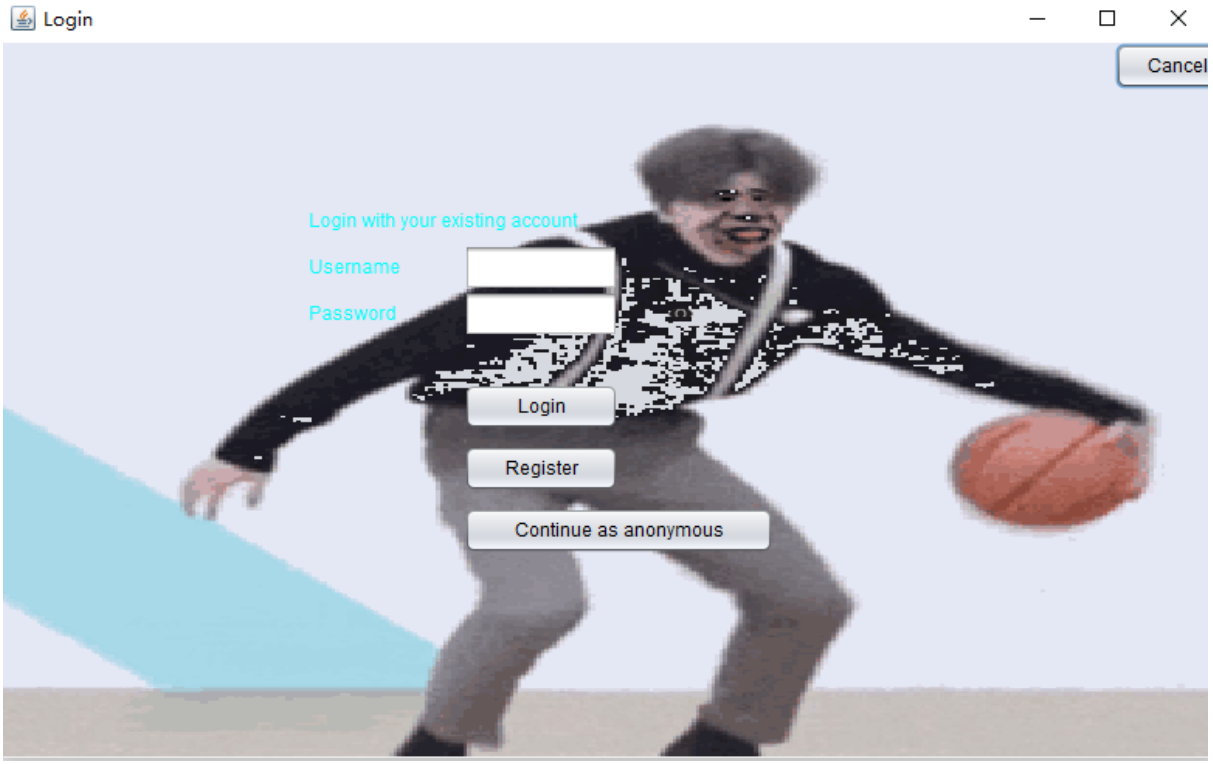
```java
// test login SOF-62, SOF-63 SOF - 65
孙泽超
@Test
public void TestLOGIN(){

    Login L = new Login( path: "src/main/resources/Account");
    String real = L.checkLogin( username: "L", password: "1");
    assertEquals( expected: "Owner", real);


    //edge case, file not found
    Login L1 = new Login( path: "src/main/resources/Account1");
    String real1 = L1.checkLogin( username: "L", password: "1");
    assertNull(real1);


    // edge case, username or password does not exist
    Login L2 = new Login( path: "src/main/resources/Account");
    String real2 = L2.checkLogin( username: "LeoYang1", password: "1");
    assertNull(real2);
}
```

This test case test if our system can check login, our test was passed, it can handle files that do not exist and successfully login. There are two edge cases to test what happens if we put a wrong username or password, and wrong Account file path. If the input username or password is wrong, it will return null.

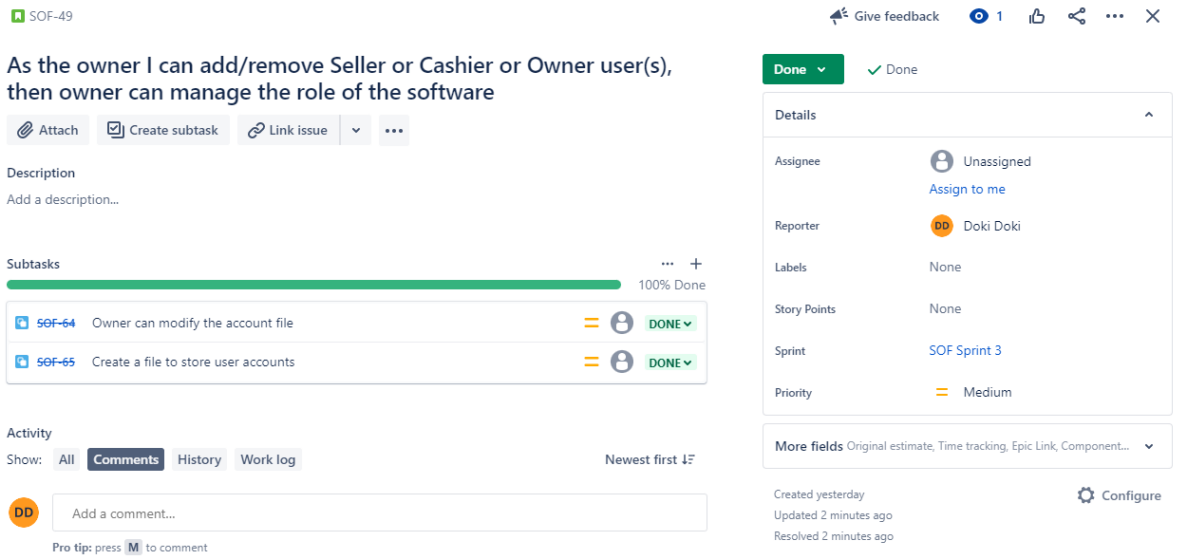
```java
//Test identify user
Customer c = new Customer( balance: 1000000, id: 1, username: "H", password: "1");

Roles r = new Roles( username: "L",  password: "1");
String real1 = r.getUsername();
assertEquals( expected: "L", real1);

String realPassWord = r.getPassword();
assertEquals( expected: "1",realPassWord);


assertEquals(c.getUsername(), Roles.roles.get(0).getUsername());
assertEquals(c.getPassword(), Roles.roles.get(0).getPassword());
```

This test case tests whether our system can get the correct username and password.

GUI Display:



User story Four:



Test cases for user story:

```
        //TestRemoveRole
        Owner n = new Owner( username: "H",  password: "1");
        n.removeRole( username: "H",  path: "src/main/resources/TestAccount");
        Owner.addRole( username: "H",  password: "1", type: "Customer", path: "src/main/resources/TestAccount");
        assertEquals( expected: "H",Owner.roles.get(9).getUsername());

        n.removeRole( username: "ss", path: "src/main/resources/TestAccount");

    }
```

This test case test if the owner can remove Role successfully from file, this test was passed, our owner can remove Role successfully.

```
@Test
public void TestModifyAccountFileAndIdentifyUser(){
    int real = Owner.addRole( username: "s", password: "h", type: "Seller", path: "src/main/resources/TestAccount");
    int real2 = Owner.addRole( username: "s1", password: "h", type: "Customer", path: "src/main/resources/TestAccount");
    int real3 = Owner.addRole( username: "s2", password: "h", type: "Owner", path: "src/main/resources/TestAccount");
    int real4 = Owner.addRole( username: "s3", password: "h", type: "Cashier", path: "src/main/resources/TestAccount");
    assertEquals( expected: 1,real);
    assertEquals( expected: 1,real2);
    assertEquals( expected: 1,real3);
    assertEquals( expected: 1,real4);
    int real5 = Owner.addRole( username: "s", password: "h", type: "Cashier", path: "src/main/resources/TestAccount");
    assertEquals( expected: 0,real5);
    List<String> result = new ArrayList<>();
    File file = new File( pathname: "src/main/resources/TestAccount");
    Scanner scan;
    int count = 0;
    try {
        scan = new Scanner(file);
        while (scan.hasNextLine()) {
            if (count > 0) {
                String[] tmp = scan.nextLine().split( regex: ",");
                switch (tmp[2]) {
                    case "Seller" -> result.add(tmp[0]);
                }
            }
            count++;
        }
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }

    assertEquals( expected: "B",result.get(1));
```

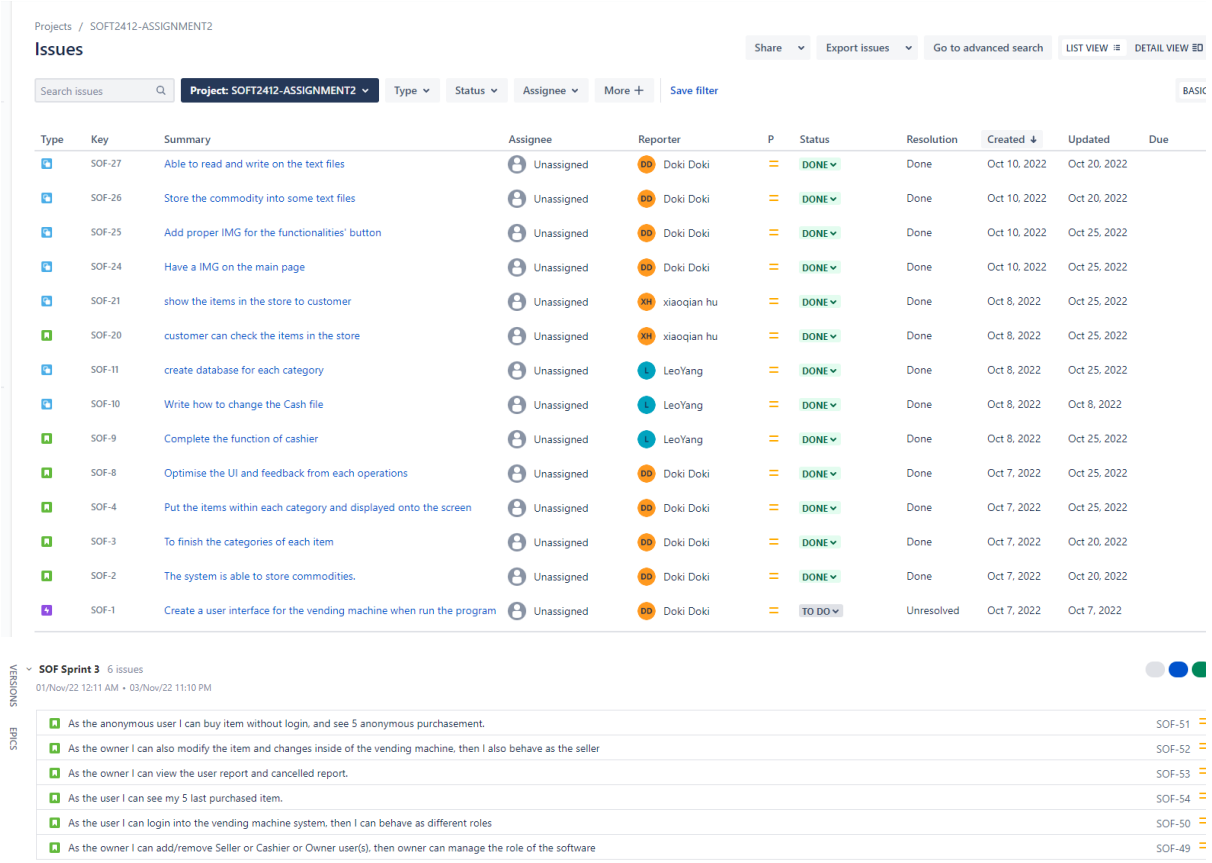
This test case tests if our owner can add a role to the file successfully, and if our test was passed, our owner can add the role successfully. There is an edge test which tests what happens if the owner adds a role where the user's name exists in the role list. return 0 if exist.

GUI Display:


Owner page
Add Role
Remove Role
Report
Modify
Cancel

4. Scrum Artifacts

- Product backlog



Projects / SOFT2412-ASSIGNMENT2

Issues

Share | Export issues | Go to advanced search | LIST VIEW | DETAIL VIEW

Search issues | Project: SOFT2412-ASSIGNMENT2 | Type | Status | Assignee | More | Save filter

Type	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
	SOF-27	Able to read and write on the text files	Unassigned	Doki Doki	=	DONE	Done	Oct 10, 2022	Oct 20, 2022	
	SOF-26	Store the commodity into some text files	Unassigned	Doki Doki	=	DONE	Done	Oct 10, 2022	Oct 20, 2022	
	SOF-25	Add proper IMG for the functionalities' button	Unassigned	Doki Doki	=	DONE	Done	Oct 10, 2022	Oct 25, 2022	
	SOF-24	Have a IMG on the main page	Unassigned	Doki Doki	=	DONE	Done	Oct 10, 2022	Oct 25, 2022	
	SOF-21	show the items in the store to customer	Unassigned	xiaoqian hu	=	DONE	Done	Oct 8, 2022	Oct 25, 2022	
	SOF-20	customer can check the items in the store	Unassigned	xiaoqian hu	=	DONE	Done	Oct 8, 2022	Oct 25, 2022	
	SOF-11	create database for each category	Unassigned	LeoYang	=	DONE	Done	Oct 8, 2022	Oct 25, 2022	
	SOF-10	Write how to change the Cash file	Unassigned	LeoYang	=	DONE	Done	Oct 8, 2022	Oct 8, 2022	
	SOF-9	Complete the function of cashier	Unassigned	LeoYang	=	DONE	Done	Oct 8, 2022	Oct 25, 2022	
	SOF-8	Optimise the UI and feedback from each operations	Unassigned	Doki Doki	=	DONE	Done	Oct 7, 2022	Oct 25, 2022	
	SOF-4	Put the items within each category and displayed onto the screen	Unassigned	Doki Doki	=	DONE	Done	Oct 7, 2022	Oct 25, 2022	
	SOF-3	To finish the categories of each item	Unassigned	Doki Doki	=	DONE	Done	Oct 7, 2022	Oct 20, 2022	
	SOF-2	The system is able to store commodities.	Unassigned	Doki Doki	=	DONE	Done	Oct 7, 2022	Oct 20, 2022	
	SOF-1	Create a user interface for the vending machine when run the program	Unassigned	Doki Doki	=	TO DO	Unresolved	Oct 7, 2022	Oct 7, 2022	

VERSIONS

EPICS

SOF Sprint 3 6 issues

01/Nov/22 12:11 AM • 03/Nov/22 11:10 PM

As the anonymous user I can buy item without login, and see 5 anonymous purchasement.	SOF-51
As the owner I can also modify the item and changes inside of the vending machine, then I also behave as the seller	SOF-52
As the owner I can view the user report and cancelled report.	SOF-53
As the user I can see my 5 last purchased item.	SOF-54
As the user I can login into the vending machine system, then I can behave as different roles	SOF-50
As the owner I can add/remove Seller or Cashier or Owner user(s), then owner can manage the role of the software	SOF-49

(The above image includes the product and sprint backlog.)

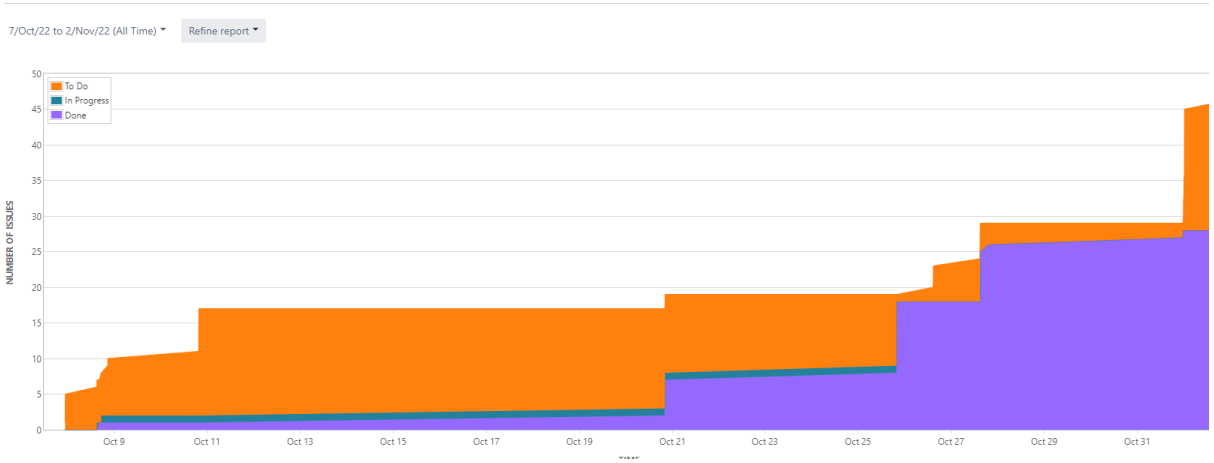
We use Jira to help with our scrum method, and we can check the previous backlog and the product backlog inside of the website.

- Project estimate

We form the project estimate by group discussions, because this is the last sprint to complete the whole project, so we need to spend quite a long time working through all the requirements and do some further error handling. During the discussion we use some charts to help us with the estimation, like cumulative charts, we can see that the amount of “Done” has increased from sprint one to sprint three, so there is a clear increase of our working efficiency, so we consider to decrease the time of the project estimation.

Not only that, we also need to ask for some feedback from each scrum member, especially for some core team members who are working

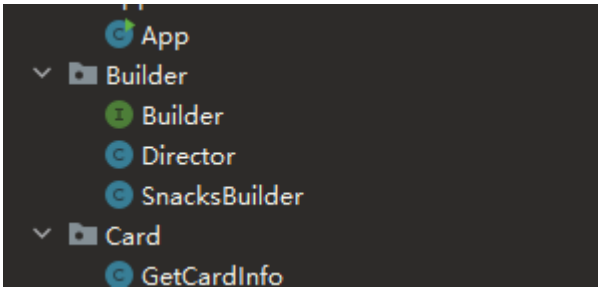
mainly on the code part, and combine the difficulty of the new user stories.



(Flow chart)

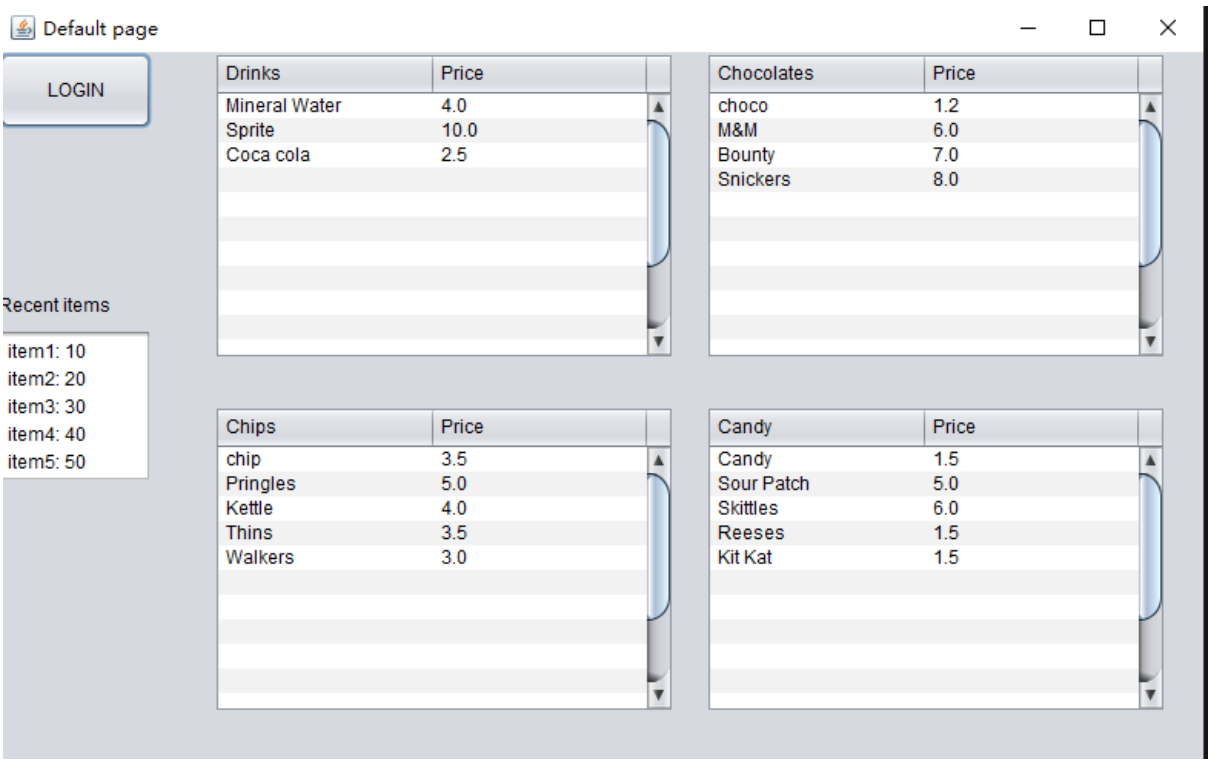
The result of our final discussion is divided into three parts, one part is the report, the second part is the time required for the test case, and the last part is the time spent in the implementation of the code part. We think the third sprint needs to spend about 8 hours on the code, 2 hours for the test case, and 4 hours for the final report. Because we may encounter bugs that we have not seen before in the process of code implementation, we need to fix them before extending new functions, so it takes a lot of time. So to sum it up we need a total of 14 hours to complete the third sprint.

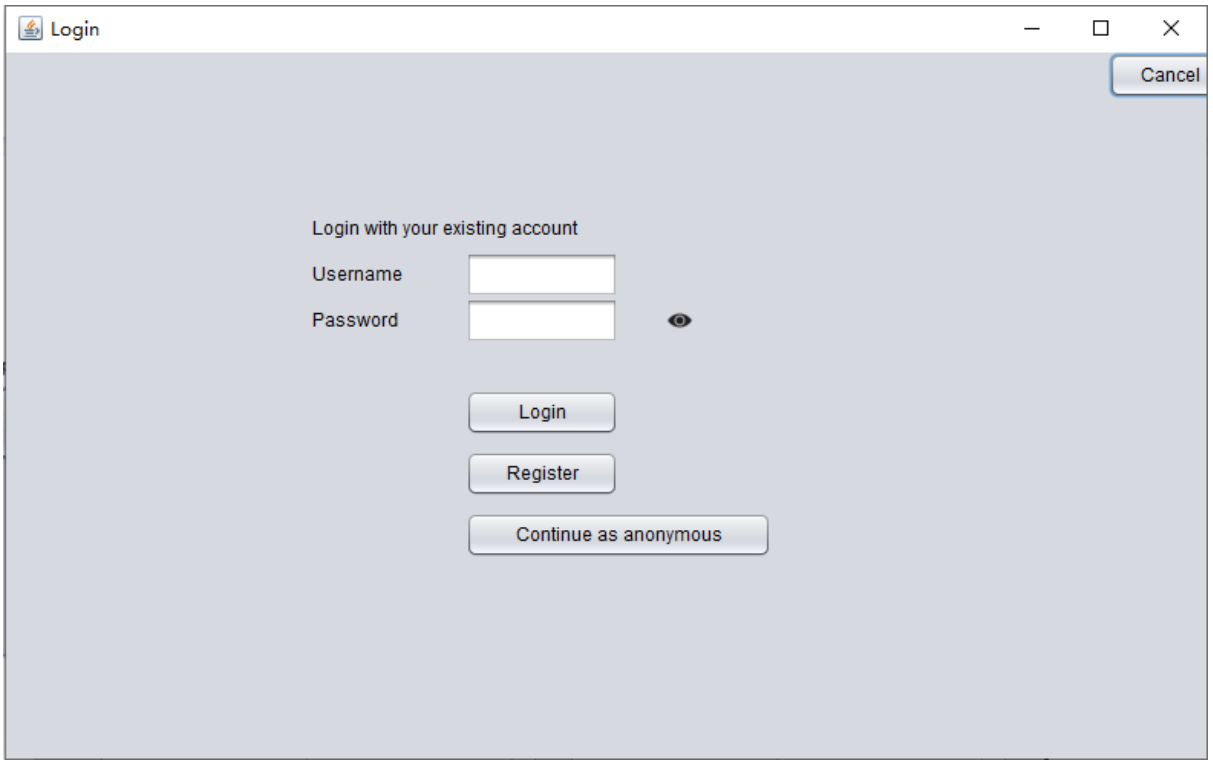
In Conclusion, reference the previous sprints because the increase of our development skill our estimation may be way more than we actual expense, it also depend on our previous codes whether it's easy to extend or not, because we implement the design pattern inside of our code so we think the accuracy might be correct but also may be less than we expected.



(Builder pattern)

- UI diagram





I think our Login UI is good, because it can hide the password to “*”, and the user can see their actually entered integer by the right side “eyes” icon

5. Scrum retrospective

From the last sprint, there is one subtask we didn't complete which is the card purchase implementation, so we spent some time completing this requirement just after the project demonstration. We took some time to discuss the reason for this situation. We think the main reason is we didn't spend enough time in the second sprint, and need to do some time estimation for every sprint, so that is why we have project estimation inside of this sprint.

We did pretty good on the project discussion, every day there was someone inside of the wechat group to make some discussion about the functionality we need to implement, and we communicate frequently during each sprint.

Generally speaking, our progress was pretty good, but we still need some improvement on the scrum method, then we can make sure that in the future development, we can allocate the time perfectly and complete all the user stories on time.

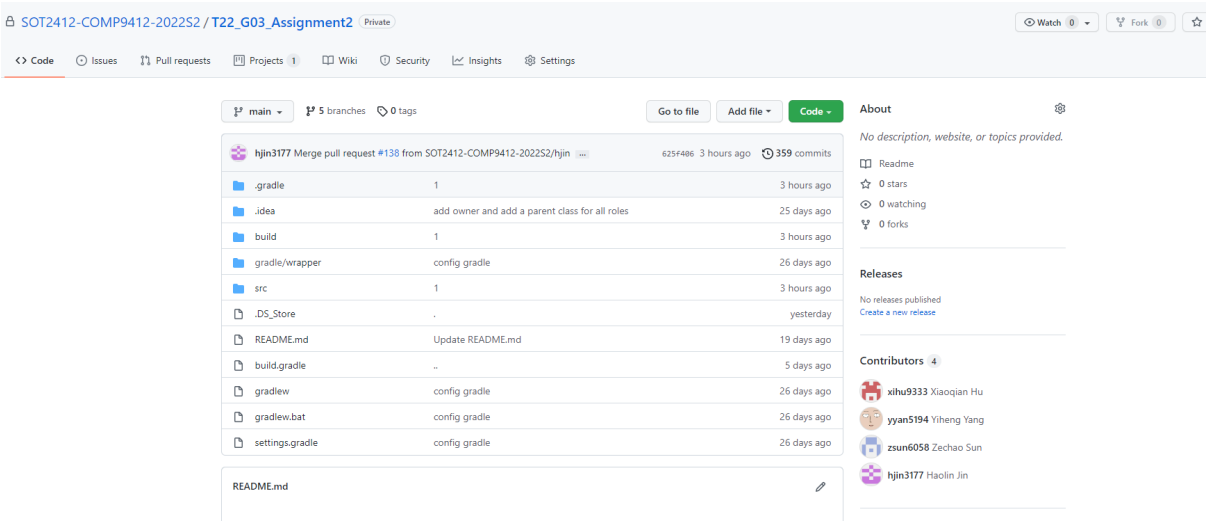
6. Daily Scrum

Everyday we will have a 15-minute time-boxed event for the development team to synchronize activities and check whether we need to merge any conflict and plan for the next day. Just like the previous sprints we usually did this through the zoom meeting which created by Haolin Jin who is the scrum master of this project and we found it quite useful, because we are responsible for the different part and it's important to have around 15 minutes chat to reconnect our idea together and to make sure no one get out of the track.

7. Agile Artefact

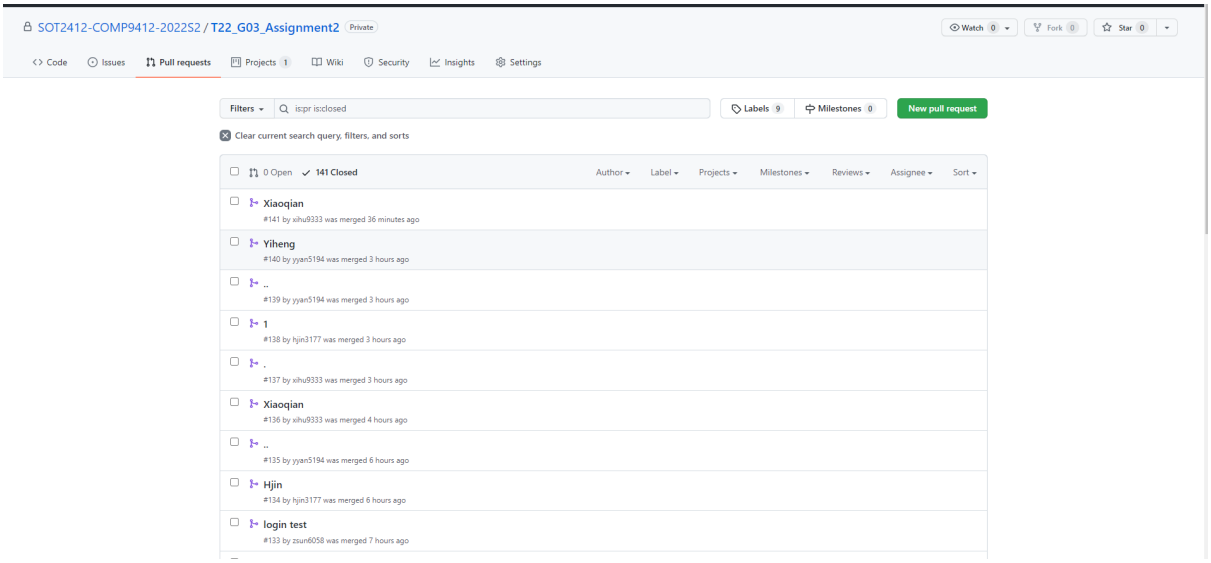
- **Github Collaboration**

By using Github, team members can update the codebase at any time, set detailed goals, record detailed data, save a lot of time, and be very efficient for team projects. For this project, we create a new repository on github and add a README file. Each member then uses git clone and link to create a clone of the target repository. Then each of us creates a new branch that allows us to break away from the mainline of development and continue working without affecting the mainline. Every time we write code in our own branch, we use the command **"git add". "git commit -m "text" then "git push"** to push to the repository. We use **"git fetch" and "git pull"** to fetch the code. It's normal to sometimes have conflicts when merging. Via Github, the team Members can create new pull requests, and Github will check if there is anything that needs to be corrected manually. Most of the time, Github will automatically correct and merge branches. However, in some cases, when the merge cannot be done automatically, Github will flag the need for manual correction. Manually modified files so that team members can manually correct them on Github before merging.

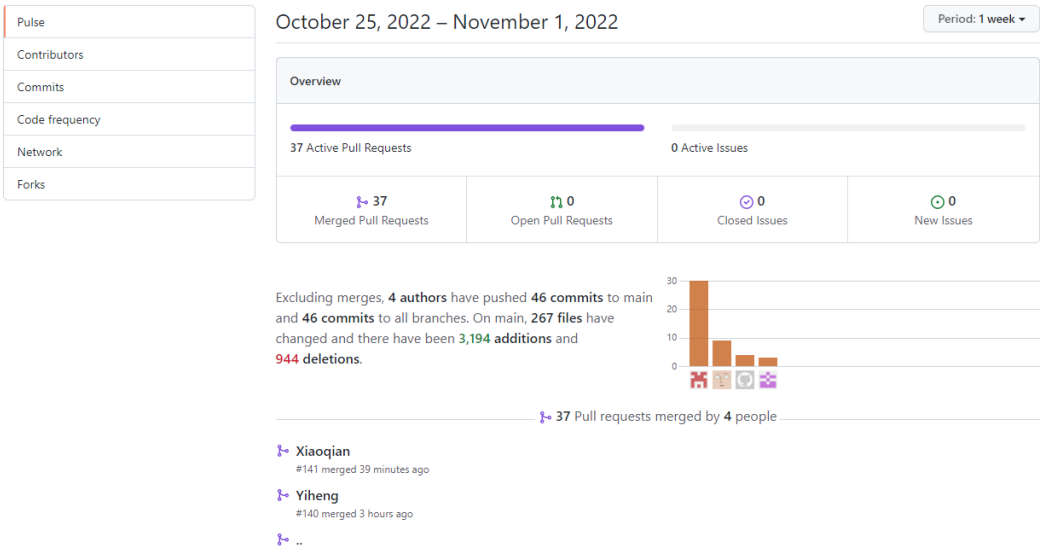


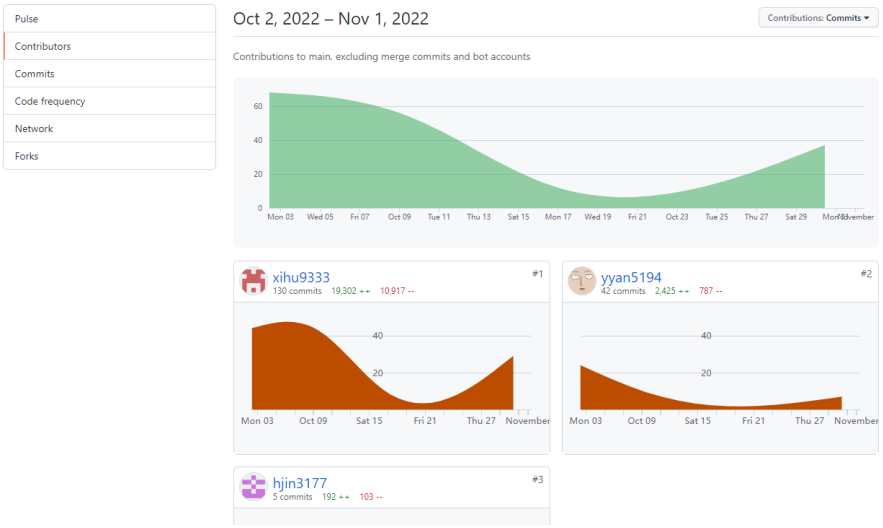
Each collaborator can view all pull requests (regardless of whether they are closed), and can select any pull request to approve, comment on, add to the project board, or modify comments. For each pull request, team members will write the purpose of the pull request, such as modifying parameters, adding features, and writing tests. By viewing pull

requests, team members can gain a clearer picture of project implementation and changes, as well as the progress and tasks of others.



Project insight:





- **README**

30 lines (22 sloc) | 1.28 KB

Raw Blame

T22_G03_Assignment2

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

introduction about how to run the project

```
For this project, using gradle above 7.4 and using gradle build to build the project, and using gradle run to run the project.
Our main class is 'App', we will use Gradle to compile and run this class, calling the main function in the App and
performing a series of operations.
```

introduction about how to test the project

```
We use jacoco to test the code which implemented by gralde, then store the test result in the build/reports/jacoco/test/html/default/
```

introduction for each class

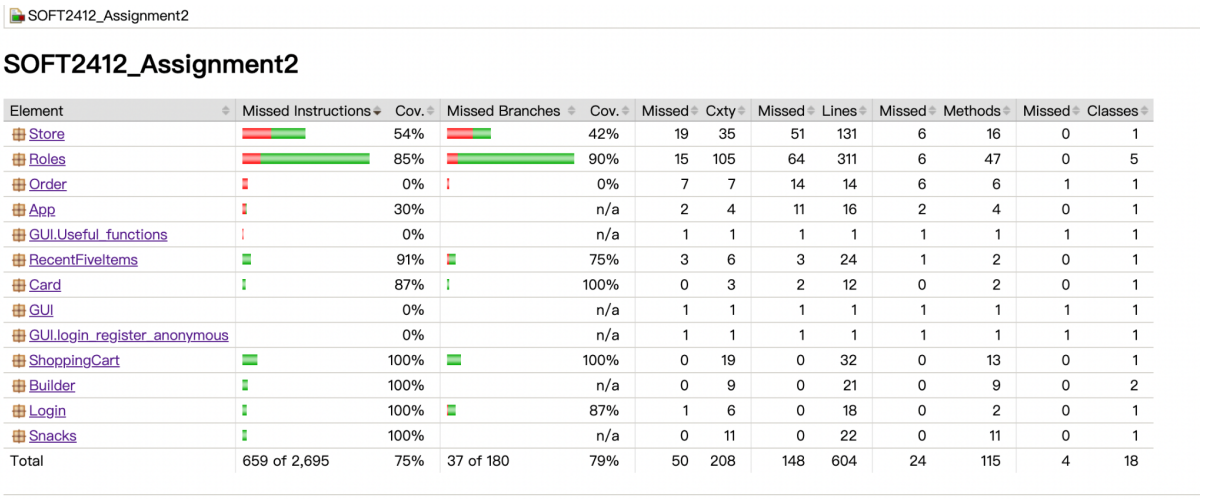
```
1. Main class: App
2. UI class: All classes in the GUI package
3. Functional and Attribute class: All classes in the Builder, Card, Login, Order, RecentFiveItems, Roles, ShoppingCart,
Snacks, Store packages.
```

Reminders

```
1. Please turn down the volume of your speaker will you run the project.
2. The vending machine do not have too many cash, so please don't fill too many cash when you pay for your orders.
3. The input of your cash will be spearate automatically, eg: enter 12 dollars will be a seperate as 1x10 dollars and
   2x1 dollars automatically.
```

- **Jacoco and Junit test**

Because the report has already talked about the test cases above, this section will be mainly focused on the Jacoco report and how to run the junit test.



SOFT2412_Assignment2

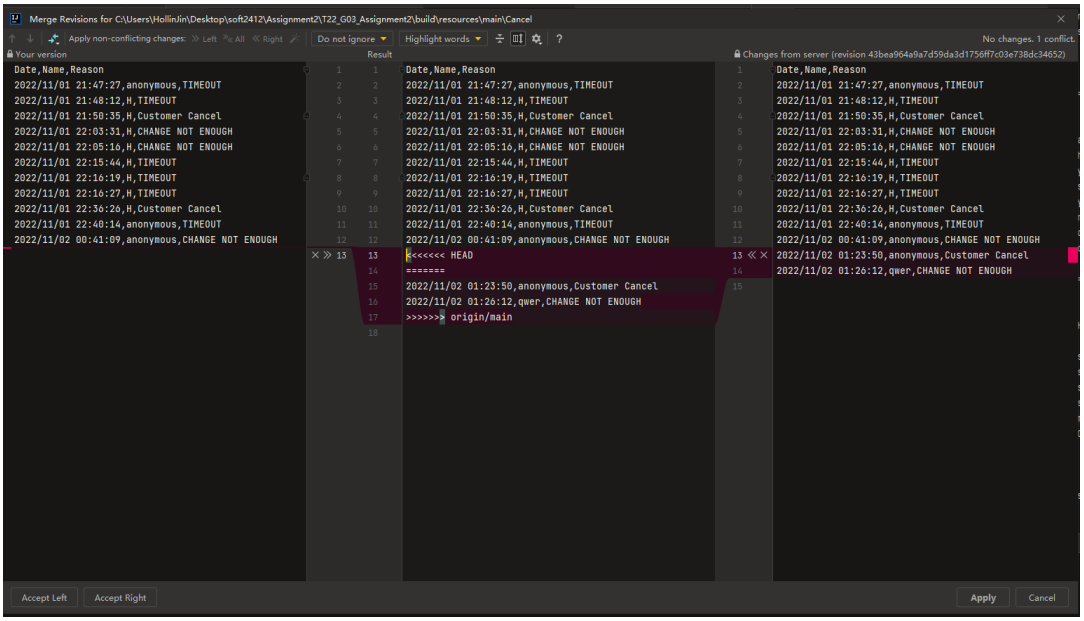
SOFT2412_Assignment2

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Store		54%		42%	19	35	51	131	6	16	0	1
Roles		85%		90%	15	105	64	311	6	47	0	5
Order		0%		0%	7	7	14	14	6	6	1	1
App		30%		n/a	2	4	11	16	2	4	0	1
GUI.Useful_functions		0%		n/a	1	1	1	1	1	1	1	1
RecentFiveItems		91%		75%	3	6	3	24	1	2	0	1
Card		87%		100%	0	3	2	12	0	2	0	1
GUI		0%		n/a	1	1	1	1	1	1	1	1
GUI.login_register_anonymous		0%		n/a	1	1	1	1	1	1	1	1
ShoppingCart		100%		100%	0	19	0	32	0	13	0	1
Builder		100%		n/a	0	9	0	21	0	9	0	2
Login		100%		87%	1	6	0	18	0	2	0	1
Snacks		100%		n/a	0	11	0	22	0	11	0	1
Total	659 of 2,695	75%	37 of 180	79%	50	208	148	604	24	115	4	18

After the tester runs the “gradle test”, it will automatically test every test case and generate the jacoco report.

- **Merge Conflict**

Because during our software development, we may encounter a lot of conflicts when trying to pull out code from the github, or push them into the github. The reason for the conflict is because each team member may change the code onto the same part, so the git doesn’t know he should accept which person’s code, now it will have conflict occurring. We can solve it either by github or inside of our IDEA.

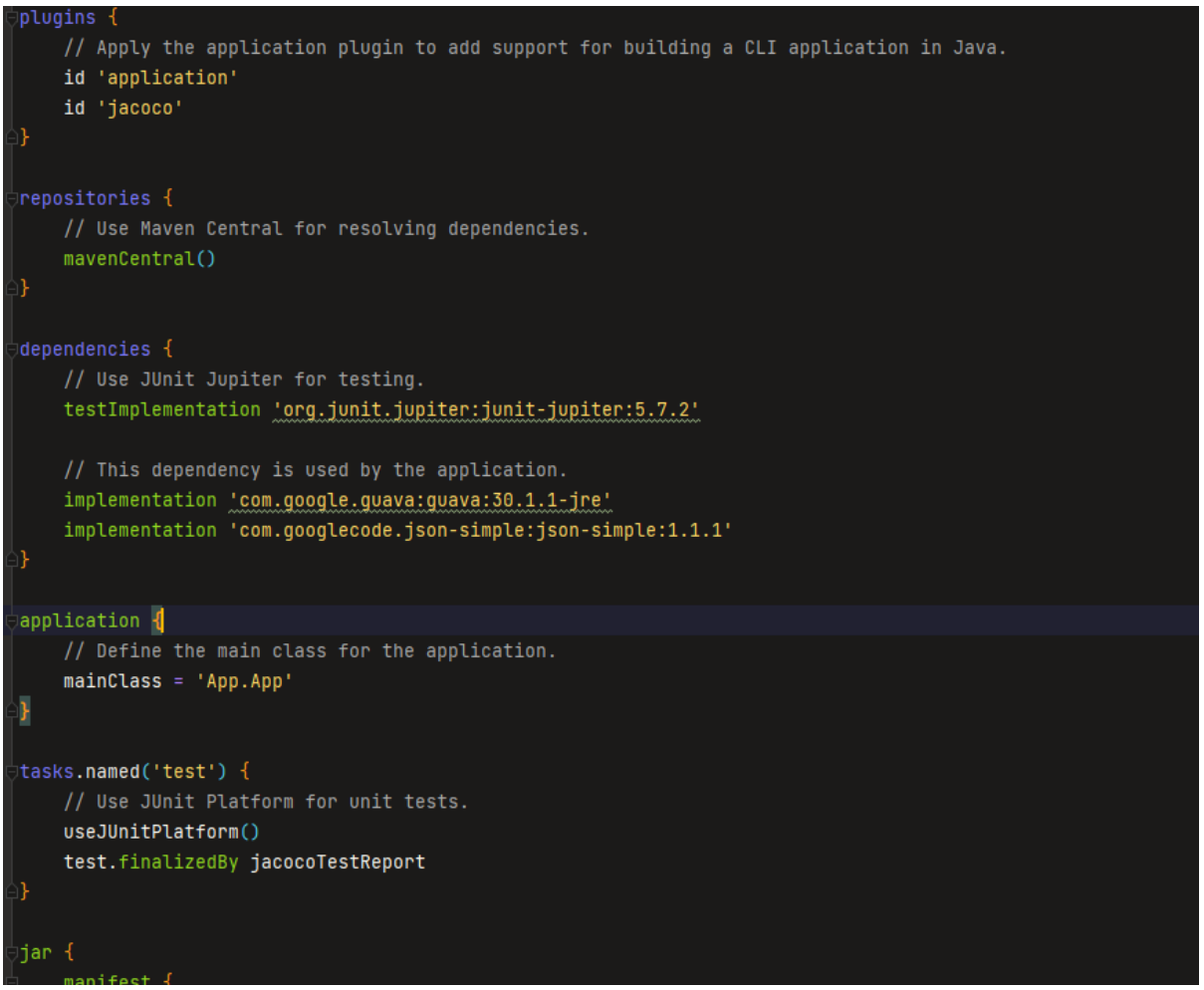


The above example is the way we merge our conflict, we will manually edit the middle part, or simply accept the left or right version of the code. After we finish editing, we click on “Apply” and it will replace our version with the new version which you just edited.

This conflict happens quite oftenly, because different people are working on the same code base, and we need to merge our conflict when this problem happens, but it’s quite straightforward.

- **Gradle**

in the build.gradle file, we add the dependency ‘org.junit.jupiter:junit-jupiter: 5.7.2’



```
plugins {
    // Apply the application plugin to add support for building a CLI application in Java.
    id 'application'
    id 'jacoco'
}

repositories {
    // Use Maven Central for resolving dependencies.
    mavenCentral()
}

dependencies {
    // Use JUnit Jupiter for testing.
    testImplementation 'org.junit.jupiter:junit-jupiter:5.7.2'

    // This dependency is used by the application.
    implementation 'com.google.guava:guava:30.1.1-jre'
    implementation 'com.googlecode.json-simple:json-simple:1.1.1'
}

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

tasks.named('test') {
    // Use JUnit Platform for unit tests.
    useJUnitPlatform()
    test.finalizedBy jacocoTestReport
}

jar {
    manifest {
```

Apply the application plugin to add support for building a CLI application in java and mavenCentral use Maven Central for resolving dependencies. In the application part, mainClass = ‘App.App’ means the first App is the folder and the second App is the main class of this program.

```
jar {
    manifest {
        attributes(
                "Main-Class": 'App'
        )
    }
}

test {
    useJUnitPlatform()
    test.finalizedBy jacocoTestReport // <- add this line
}

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

    }
}
```

Specifies that JUnit Platform (useJunitPlatform()) (JUnit 5) should be used to execute the tests. And we set jacoco in build.gradle which can generalize report after run the program that can open in browser included html file and csv file, because we set html.enabled = true and csv.enabled = true.

- **Jenkins Intergration**

  Why we need Jenkins:

We use Jenkins in our project to auto build and test our code and behave as CI pipeline. The reason we need Jenkins in our project is because we are using agile development in this project but there are some differences between this project and the previous project. In the last project we didn't implement the scrum method when working on the project, but this time we need to use scrum method and also agile development to work on the project. So the CI pipeline will save us a lot

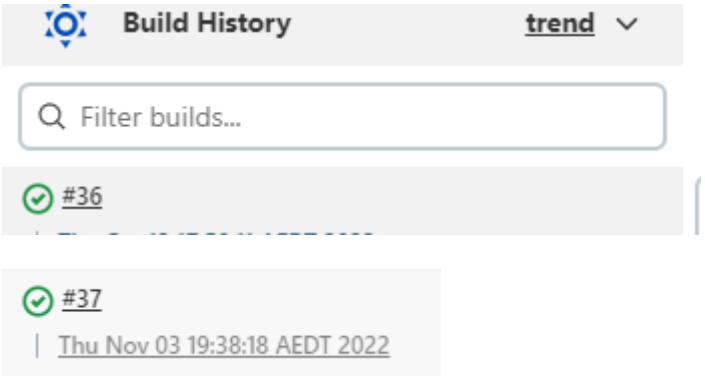
of time, it will automatically build and test our code for each commit and push commands, and we do not need to run the ‘gradle clean test’ over and over again after every commit, the Jenkins will notice each commits and build run our code if we pass all the test it will generate the Jacoco report automatically without and manually operations.

Setup the webhook from the github:

The setup of the Jenkins is pretty much the same as the first project, so in this report we will not talk about the basic setup deeply.

In this part I use ngrok to continuously integrate with Jenkins. Inside the terminal we need to manually run the command “ngrok.exe http 8080” to generate a temporary URL for our local host address, we need this for the github webhook to catch every commit.

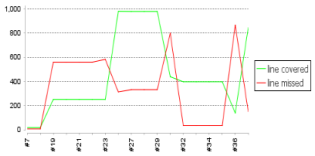
Inside of the github project, we can find the webhook inside of the setting button, then we need add above URL with “/github-webhook/” into the Payload URL then the github will send post request to our Jenkins for every commit we made. Now, everytime we make some changes and push onto the github, the webhook will notify our Jenkins, and the Jenkins behave as the CI pipeline and automatically pull our new version of the project and build to test our Junit and also form a Jacoco report at the end. We can find this inside of the build history in the Jenkins.



Jacoco result:

JaCoCo Coverage Report

Download `jacoco.exec` binary coverage file

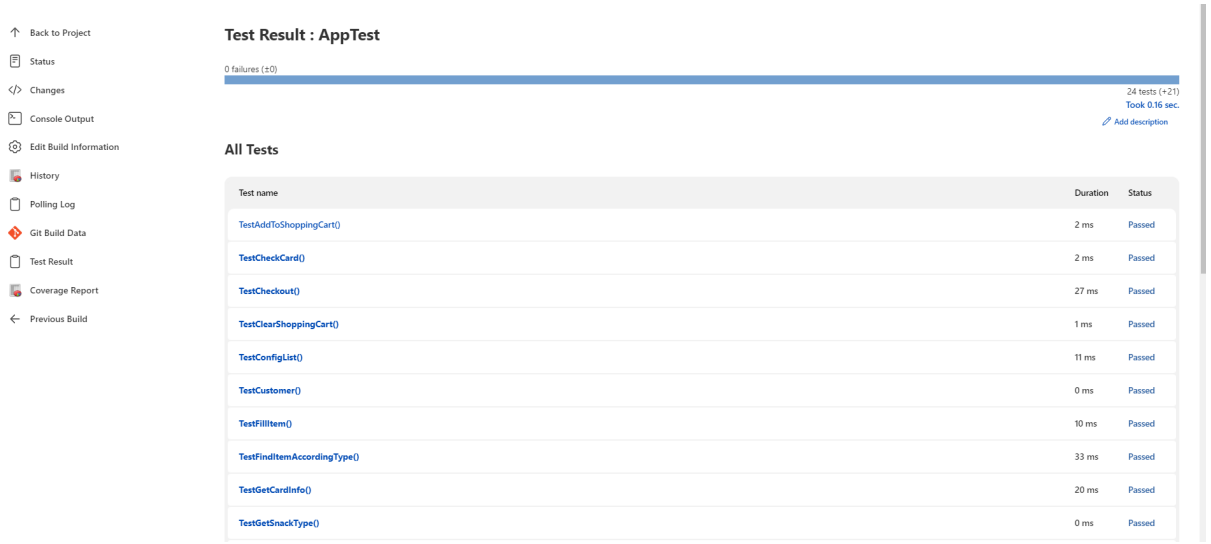


Overall Coverage Summary

name	instruction	branch	complexity	line	method	class
all classes	86% **M:** 669 **C:** 3994	81% **M:** 40 **C:** 172	79% **M:** 53 **C:** 197	85% **M:** 150 **C:** 845	83% **M:** 24 **C:** 117	79% **M:** 4 **C:** 15

Coverage Breakdown by Package

name	instruction	branch	complexity	line	method	class
(default)	**M:** 10 **C:** 1958 99%	**M:** 3 **C:** 29 91%	**M:** 3 **C:** 39 93%	**M:** 2 **C:** 389 99%	**M:** 0 **C:** 26 100%	**M:** 0 **C:** 1 100%
App	**M:** 38 **C:** 17 31%	**M:** 0 **C:** 0 100%	**M:** 2 **C:** 2 50%	**M:** 11 **C:** 5 31%	**M:** 2 **C:** 2 50%	**M:** 0 **C:** 1 100%
Builder	**M:** 0 **C:** 67 100%	**M:** 0 **C:** 0 100%	**M:** 0 **C:** 9 100%	**M:** 0 **C:** 21 100%	**M:** 0 **C:** 9 100%	**M:** 0 **C:** 2 100%
Card	**M:** 6 **C:** 43 88%	**M:** 0 **C:** 2 100%	**M:** 0 **C:** 3 100%	**M:** 2 **C:** 10 83%	**M:** 0 **C:** 2 100%	**M:** 0 **C:** 1 100%
GUI	**M:** 6 **C:** 0 0%	**M:** 0 **C:** 0 100%	**M:** 1 **C:** 0 0%	**M:** 1 **C:** 0 0%	**M:** 1 **C:** 0 0%	**M:** 1 **C:** 0 0%
GUI.MainPages	**M:** 0 **C:** 0 100%	**M:** 0 **C:** 0 100%	**M:** 0 **C:** 0 100%	**M:** 0 **C:** 0 100%	**M:** 0 **C:** 0 100%	**M:** 0 **C:** 0 100%



Above are two images showing the Jenkins test result, and the coverage for the Jacoco report which was generated by Jenkins itself after running the auto building and auto testing.

8. Client feedback and review

We received feedback during the third and the last sprint demonstration, we need to pay attention to the error handling part, then the rest of the project is pretty good.

Overall the client thinks our product is pretty good, and we finish the development and meet all the requirements successfully.