THE UNIVERSITY OF SYDNEY

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - GROUP ASSESSMENT

Unit of Study: SOFT2412 - Agile Software Development Practices

Assignment name: Scrum Software Development Project

Tutorial time: 4:00pm Tuesday **Tutor name:** Nofil Khan

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. Saksham Lama	slam7511	Yes / [redacted]	Yes/[redacted]	Saksham
2. Peter McGee	pmcg3506	Yes / [redacted]	Yes / [redacted]	
3. William Wang	wwan6344	Yes / [redacted]	Yes / [redacted]	
4. Qiuda Song	qson6632	Yes / [redacted]	Yes / [redacted]	
5. Zihe Wang	zwan3215	Yes / [redacted]	Yes / [redacted]	Craig Wang
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

Report - Sprint 1

Work carried out by the team

First we assigned roles to each of the team members (Product Owner, Scrum Master, and Dev Team). The Scrum Master created the Jira account in order for all members to have access to the Scrum Board, Scrum Backlog and the various reports such as the Burndown chart to be used for the analysis of each sprint. The Product Owner then went through the assignment specs and entered the user stories into the Backlog in Jira Agile. The dev team decided which user stories will be implemented and tested based on priority. There were eleven user stories added for Sprint 1. We assigned in our team meeting the story points for each user story, based around the efforts and time taken to complete the assigned user story.

The communication channel setup for the sprint is Slack. The Scrum Master used Slack to communicate with the members to set up zoom meetings for the four daily-standups required for each sprint. The meetings lasted around 30 minutes to an hour. During this meeting we discussed the progress made so far by each member and the improvements to be made for the future. During the sprint the Product Owner wrote up Success and Acceptance criteria for each story of Sprint 1. The Product Owner kept a consistent check up on the user stories based around the current implementation of the Sprint.

We utilised CD/CD tools for our group collaboration in order to improve the automation of the steps taken to implement and test the application. GitHub was used for version control for a common The GitHub was integrated by Jenkins in order to view the test coverage of determine if the push made to the GitHub was built correctly. The tools improved the overall efficiency of the steps required to fulfil the features of the application.

After the implementation of the user stories for Sprint 1, it was time for the Sprint review. The sprint review was conducted during the demonstration of our application to the client, as an opportunity to receive feedback. The feedback allowed us to understand the weaknesses within our application so far, and the ways we can improve the process of implementing the requirements of the application by working Agile as a team. During the demonstration, we asked questions to better understand the requirements of the application for future sprints.

Lastly, our team underwent a retrospective of our progress so far with the application and the steps we should take for future sprints. During this the Scrum Master provided his feedback on how effectively and efficiently we should aim to meet the client's and the application's requirements. The artifacts used for Sprint 1 details the summary of the progress we have made so far with the application, listed below.

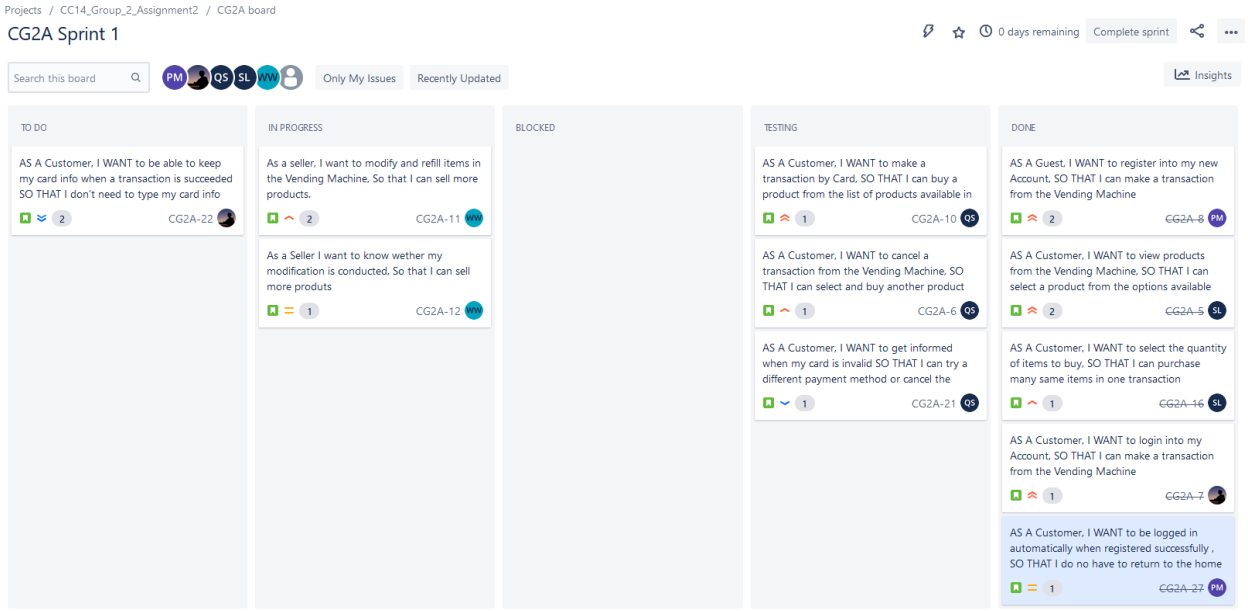
Scrum Practices

Sprint Planning:

During this stage, the Product Owner created the Product Backlog using Jira. The sprint goal was determined by the Product Owner and based on the sprint goal, the user stories for Sprint 1 was determined, with the help of the Dev Team. The process of determining the user stories was firstly determining its priority and then deciding its story estimate. The story estimates are an indication of the duration of implementing and testing the user story incorporated with the effort it will require.

At the end of each daily meeting, the Scrum Master collated everyone's work and provided a brief analysis to make improvements for the next day and to ensure everyone was up to data with Scrum methodologies.

Active Sprint Board:



This is the active sprint board that we are using to identify how each member is progressing with their assigned user stories. The board came with the TO DO, IN PROGRESS and DONE columns; during the sprint, the BLOCKED and TESTING columns were created to better show the progress of the tickets throughout the sprint.

The BLOCKED column was created, as after the initial sprint planning, it was discovered that some of the assigned user stories were dependent on the completion of another member's user stories. As a result, these were placed in the blocked column to indicate that this work had stalled. This allowed for the issue to be highlighted at the daily standup meetings where the BLOCKED, TESTING and IN PROGRESS columns were the focus.

Roles

Product Owner: (Zihe Wang)
As the Product Owner, my main responsibility is creating user stories and set priorities for them. I managed the product backlog and set priority based on functionality and difficulty. I discussed with other team members to make sure they understand and satisfied with all the user stories in the backlog and ask them to make some supplement if needed. I discussed with the team about how to determine which interface should each user story use, GUI or CLI. Since we are not familiar with Javax and swing package, I tried to learn how to implement swing GUI and finished a sample Login window at the beginning of this sprint. So that other dev team members can reuse my code to create other GUIs such as register window or transaction window. I attend every meeting during the sprint and listen to the dev team to share their approach and demonstration.

Scrum Master: (Saksham Lama)
As the Scrum Master my main responsibilities were to clear the issues all team members faced in understanding our Scrum functions and to ensure everyone remains agile in their approach throughout the sprint. I have ensured through Slack to ensure consistent communication is maintained between Product Owner, Scrum Master and Dev Team. I tried to ensure everyone remained Agile in their approach of fulfilling the application requirements. I held team meetings four times a week. Communication for me is really important and I have emphasised this to all members of the team multiple times to always communicate any issues so we can resolve them as a team. There were a few challenges faced during Sprint 1. The challenge was due to GitHub servers undergoing maintenance. This was challenging because the Dev team couldn't get access to the updated version of the code which was stored locally and some of the implementation depended on a user story which was unimplemented. The user story CG2A-22 depended on the implementation of CG2A-10. However, as the Scrum Master I suggested to my team to test their code locally and start working on the report in the meantime. Overall, I have tried to make sure the process of fulfilling client and application requirements is overall streamlined and Agile. Along with being the Scrum Master, I have been involved in the implementation of the following user stories CG2A-5 and CG2A-16.

Dev Team: (Quida Song, Weihui Wang, Peter McGee)

The dev team's responsibility is to perform the majority of the development work on the project, by completing user stories and building out the application. These stories are assigned to the dev team from the backlog as part of the sprint planning meeting. Work is assigned based on the story points of the tickets, so as to ensure the dev team (who don't have other responsibilities within the project) are able to complete.

Daily Meetings

Friday 14/10/22, 10:00-11:00pm

NAME	What did I do today?	What will I do tomorrow?	Encounter any issues?
William wang	I have completed the functionality part of the two user stories that were assigned to me. CG2A-11, CG2A-12. Being one of the development team. I created a seller page to ensure the seller can edit all items in the vending machine.	Report work and start testing , merge my work with the main branch.	Problem with reading and writing in json. Fixed by watching tutorials on json files.
Saksham Lama	I have completed the two user stories that were assigned to me. I have tested them fully. CG2A-5 and CG2A-16. Being the Scrum Master, I have made sure everyone has remained up to date with their user-stories I assigned to them. I have communicated with the Product Owner to make sure acceptance and success criteria are added for each of the Sprint 1 user stories. Created the structure of the report.	Assign myself some parts of the report to work on.	Communication between team members needs to be more strong.
Peter McGee	Completed assigned user stories. Also included refactor of already created user login process to its own class, allowing GUI and cmdline to reuse the same code.	Report work and merge PR into main - conflicts expected.	Some tests were failing on Windows builds. This was due to different end of line characters. Resolved by using .strip() on strings being compared.
Qiuda Song	I listened to other group members who demonstrated their functionalities, and asked questions on the overall coding structures.	Start writing my coding part based on the current coding structure	The usyd github server was down, so I was not able to start coding today as planned.
Zihe Wang	I have completed the user story CG2A-7 which is assigned to me. As the Product Owner, I	Waiting for other members to accomplish the	Problems related to connecting GUI with command line

	discussed with team members about dealing with existing bugs.	card transaction part which is the prerequisite of my other user story. Check members' pull requests and deal with them.	interface. Trying to find a better solution.

Saturday 15/10/22, 5:00-6:00pm

NAME	**What did I do today?**	**What will I do tomorrow?**	**Encounter any issues?**
William wang	Worked on test cases for CG2A-11, CG2A-12 , have a coverage of 56 percent.	Finish the report and integrate with quida's part	None
Saksham Lama	Worked on the report section. Being a Scrum Master, I conducted the meeting and ensured everyone was on track with their user stories.	Work more on the report and ensure that everyone is up to date with their user stories.	None
Peter McGee	Finished tests and integration of the user manager module. Worked a little on the report.	Finish report section. Work on testing what is ready for demo.	Getting PRs approved can be slow and make integration harder.
Qiuda Song	Did not make to the meeting.	Work on the report section. Work on test cases.	Noticed a bug from other member's part, and reported it in the communication channel.
Zihe Wang	Excluded the GUI method from the jacoco test report by creating a new interface to mark methods as ignored. Which can be used in future commits.	Finish the report segment.	None

Sunday 16/10/22, 9:00-10:00pm

NAME	What did I do today?	What will I do tomorrow?	Encounter any issues?
William wang	Work on CG2A-11,CG2A-12, add seller page into the default page.	Finish report section.	Gui can't successfully swap pages as default page always running at the back end.
Saksham Lama	Work on completing the report sections assigned to me. I conducted the meeting and ensured everyone was on track with their user stories.	Work on completing the report sections assigned to me.	None
Peter McGee	Fixed various bugs that arose in the development of the application - multiple scanners running, some products not displaying etc	Test demo, work on report, fix bugs, assist in integration.	None
Qiuda Song	Finished CG2A-21 and other associated stories(inside the same branch), and asked my team member Weihui to test my code and merge it with his part. Demonstrated my functionality on the meeting.	Finish the report section.	None
Zihe Wang	Request changes for pull requests from team members.	Finish the report section.	None

Monday 17/10/22, 9:00-10:00pm

NAME	What did I do today?	What will I do tomorrow?	Encounter any issues?
William wang	Enabled transaction page to decrease item quantity after each purchase	Finish the report	None
Saksham Lama	Work on completing the report sections assigned to me. I conducted the meeting and ensured everyone was on track with their user stories.	Finish the report and actively take part in the demonstration of the application. Take on feedback from the client and ask questions for future sprints	None
Peter McGee	Checked and merged existing PRs. Working on report sections.	Test demo for client.	
Qiuda Song	Working with William together on integrating our functionalities together so that after each transaction, the items in the database gets decreased accordingly	Attend the demo remotely	None
Zihe Wang	Working on the report section and trying to figure out the issue found before. Finished the user story CG2A-22 with new techniques to use native pop-up windows.	Attend the demo	None

Sprint Review

Sprint goal: The user should be able to run the application, and see the products available, making a card purchase with very limited functionality.

Was the sprint goal accomplished?
The goal of the sprint was accomplished. Running the application, seeing available products and making a purchase with a card are all available functionality. In addition, the ability to login and register new users, make changes to products as a seller, and saving of card details were all implemented to some degree.

Client feedback:

- Have the default page include more details about the items displayed to the User.
- Client Preference: Use GUI instead of CLI, due to ease of implementation and visually appealing to the user.
- Lack of user commands displayed on the CLI, which would lead the user to confusion on how to use the application.

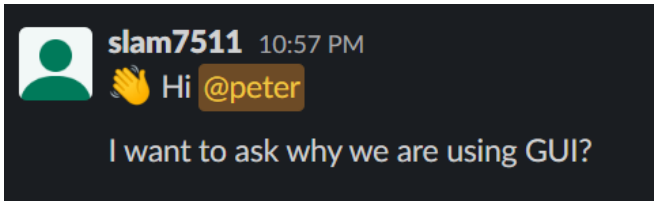
Retrospective:

Member	What went well during the Sprint?	What can be improved in the future?
William wang	Tasks are distributed evenly. The development team fully understands the requirements from the product owner.	Team communication needs to be conducted more frequently.Team members should respond as soon as possible when they see the instructions.
Saksham Lama	The application requirements chosen for Sprint 1 was built with complete functionality and testing. The tasks were equally distributed by the Scrum Master. Overall, the development is Agile.	Communication was a major issue in determining the timings of the daily stand-up meetings. This can be improved by having timings set for 4 days within the week early on.
Peter McGee	Built the basic functionality for the vending machine application. Explored the cmdline and gui interaction methods for the system. Team collaborated on various functions well, bug fixing and adapting each other's code.	Team communication and meeting organisation was difficult. As there were no set times due to team member availability, zoom meetings proved difficult to organise. We may try and set these meetings earlier in the week and keep the meeting short to ‘what i did, what’s blocking, what i’ll do’ methodology.
Zihe Wang	The development is very agile. Each pull request is reviewed by a team member, only classed done testing can	Arrange the tasks in a more logical way, make sure won’t let a team member always waiting for another member’s part to get finished and then

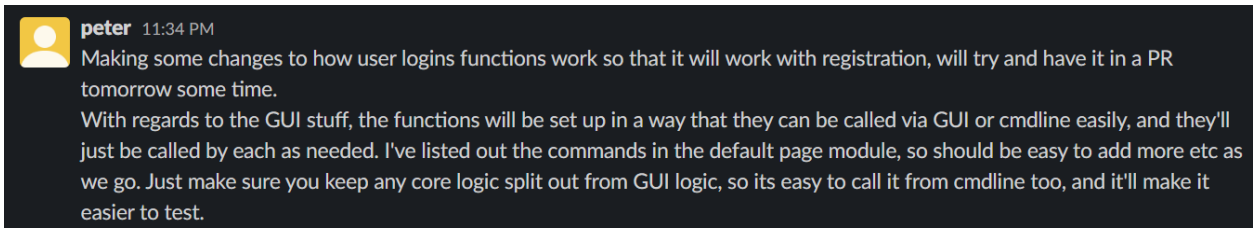
	be marked as Done on Jira. The tasks are well splited to each team member. Everyone agreed on the design pattern.	can start working. Hard to organize meetings because everyone has different available time.
Qiuda Song	Every team member had very specific tasks to be working on throughout the sprint. Each single task was light, so that we can clearly see how our team members were progressing on their assigned tasks	Because the github server was down for one day, I had some problems accessing the correct version of the code bases. I would recommend team members to pull code frequently from the main in the next sprint to avoid delay caused by such incidence, and always work on the newest version if possible.

Issue/Conflict Resolution

Initially we had planned to complete the application interaction as a command line application. However, during the implementation of certain features (login, card transactions) it was found that certain things are much simpler and clearer to use when implemented through graphical user interface elements.



As such, there was uncertainty and much discussion amongst the team in the early stage of the project, where we had developers going in two different directions. This was resolved in our online meetings, where it was decided that we would implement as much functionality as possible through command line and graphical user interfaces. This would help to keep the logic code split out from the interface elements, as the functions would need to be modular and usable in both places.

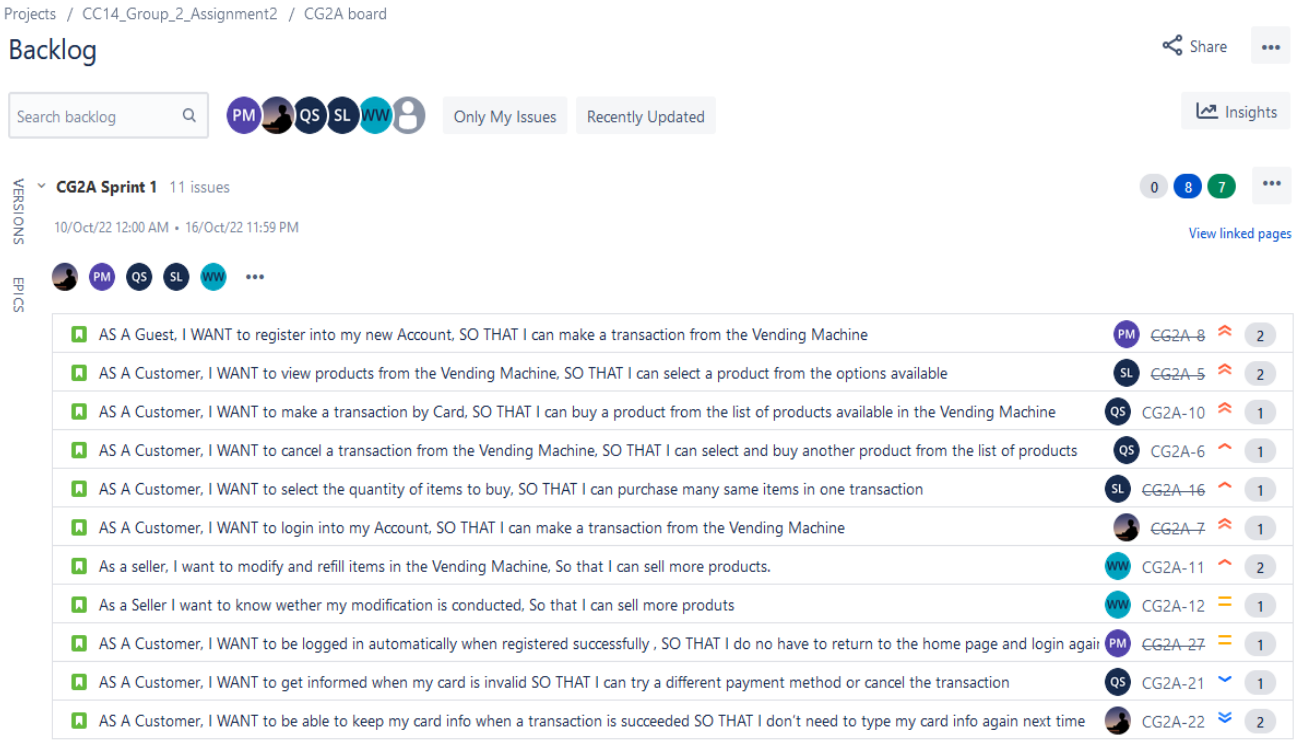


At the client demo, it was highlighted by the client that the command line interface was difficult to read. The current implementation was used as the number of items the vending machine can display is too much for a single command line window, so a cutback view was used.

As this doesn't present a very good user experience, the decision was made to proceed with the graphical user interface, rather than using the command line interface

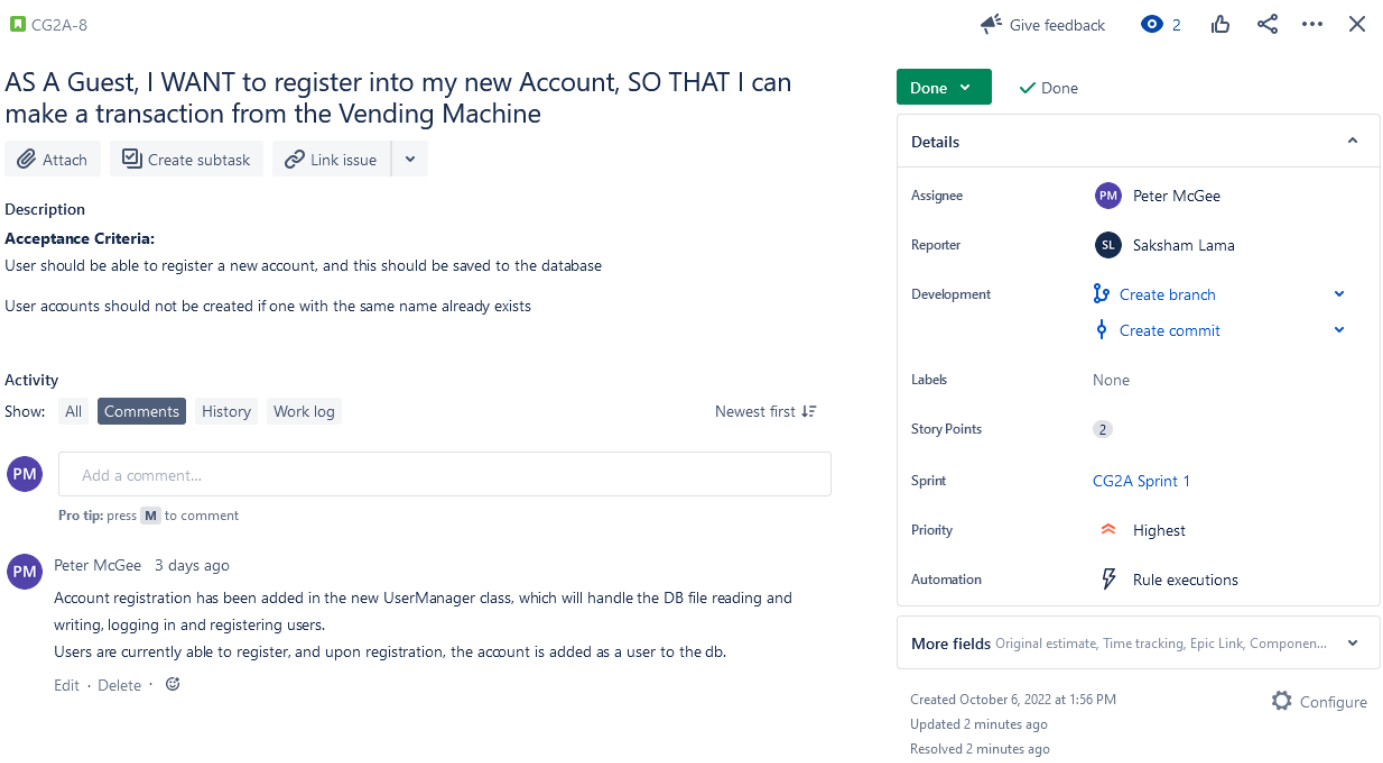
Scrum Artefacts

Project Backlog



The above image shows the user stories which were selected for the first sprint, as these were identified as getting the minimum viable product working, and a base for future development of the application. It can be seen that these are generally the highest priority tickets in the backlog (compared with below), with some lower priority tickets chosen that tie in with other development tasks.
The number on the right is the story points of each ticket, this identifies the expected workload that completion of this ticket will require. This was assigned during the sprint planning meeting, and is used to assign the tickets evenly across the team with respect to workload rather than number of tickets.

The above shows the list of user stories which have been created for the project, but were not selected to be within the first sprint. These will be reviewed and prioritised for the next sprint of the project.



Example of a user story. This user story shows the task in the form ‘AS A {} I WANT {} SO THAT {}’. Acceptance criteria are shown with the task, which were added by the Product Owner.

Comments are added to update anyone following the ticket with what has been done, or any issues that have arisen. On the right, the reporter and assignee can be seen, the assignee was assigned during the sprint planning meeting, where the priority and story points of this ticket were assigned.

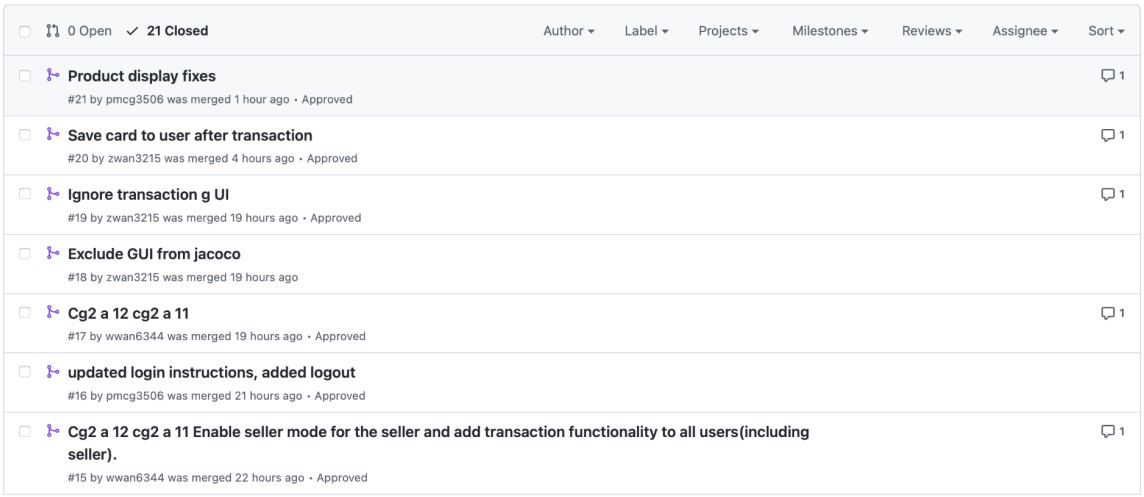
Burndown Chart:

This indicates that the rate at which tickets were completed at the beginning of the sprint were on track to be completed with the sprint, but as the sprint drew on, ticket completion slowed. This is due to the increasing complexity of systems being implemented, and blocked issues waiting on others to complete tasks. [LINK]

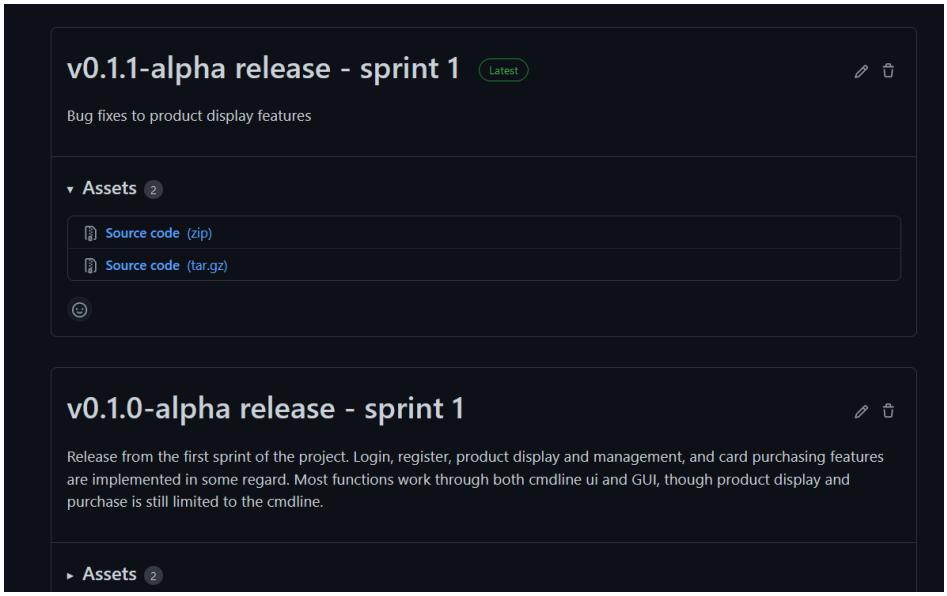
CI/CD

Git

Git is being used as the source code and collaboration platform for this project. The git repository is hosted on GitHub Enterprise, the address is https://github.sydney.edu.au/SOT2412-COMP9412-2022S2/CC14_Group_02_Assignment_2
The main branch is set to protected and each pull request must be reviewed so that it can be merged into main branch. If there are issues included in the pr, the reviewer will request the developer to change certain things and get reviewed again. We have 21 closed PRs in Sprint1.



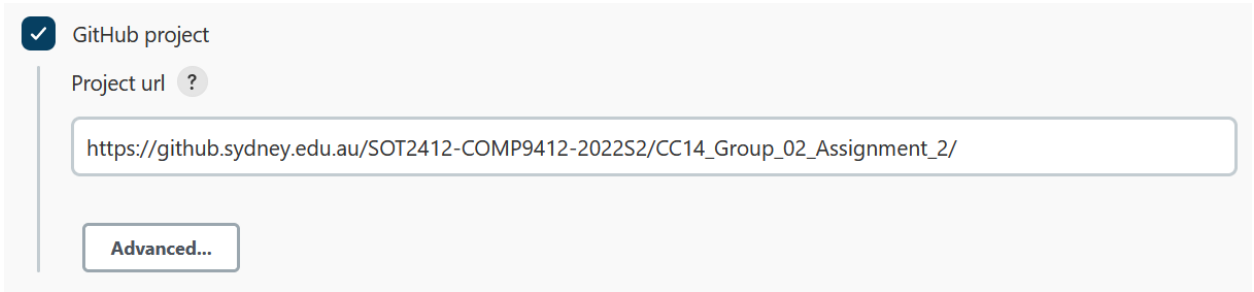
For each user story, the developer working on it is asked to create a new branch and include the code of that user story on it. At the end of Sprint1, we released an alpha version which included all the functionalities planned in sprint1.



Jenkins

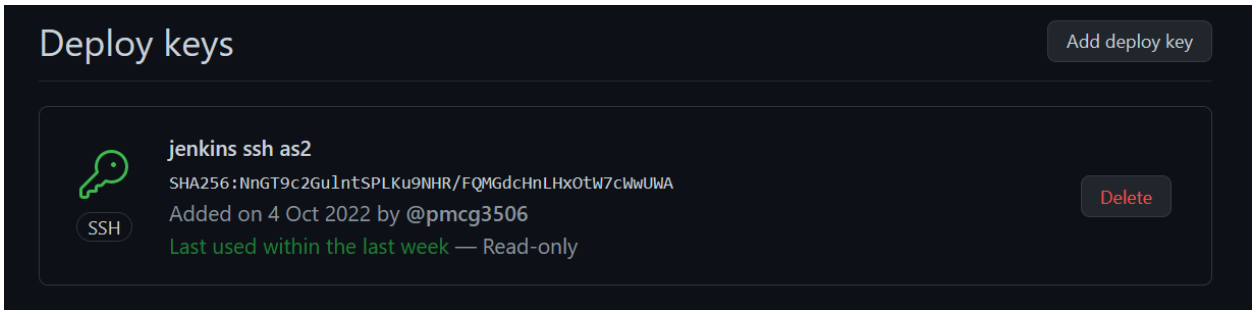
Jenkins is deployed to a web accessible Linode instance online. It is integrated with github using webhooks which trigger a build on every branch uploaded to github, or merge into main. This constitutes our continuous build process, where we can easily see that each change to the repo doesn't break existing tests or cause compilation errors.
Team members also perform these checks locally, but by doing it on jenkins a record is kept, and a consistent environment is used ensuring local configuration doesn't hide issues.

Configuration:
After the installation of jenkins, the following configuration was performed. A new jenkins project was created, and the following configuration was made.

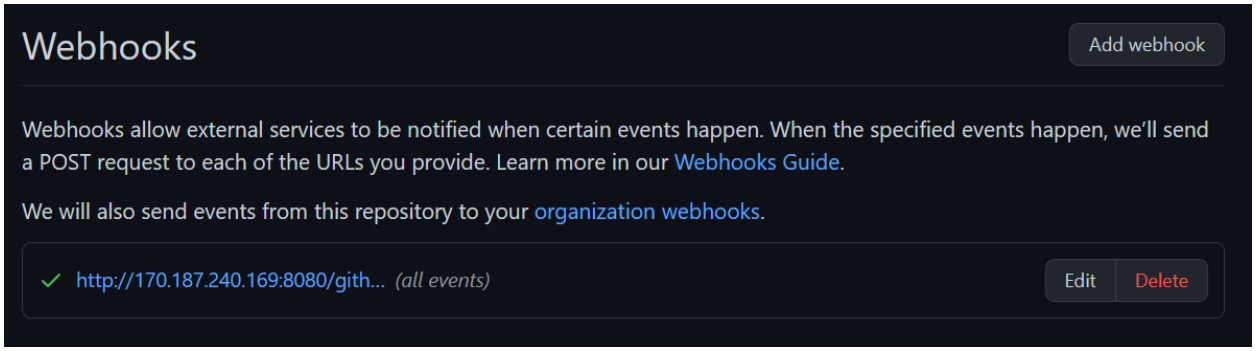


The github URL was specified, this provides a quick link to the repo on the main project page in jenkins.

This section enables Jenkins to pull down the source code from the github repository using credentials that are configured for the project. The account for this is created with jenkins using a key pair that provides access to the project.

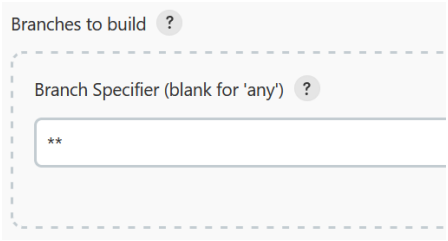


Webhooks are also created on github which hit the jenkins site, informing that a new branch is available on the repo.



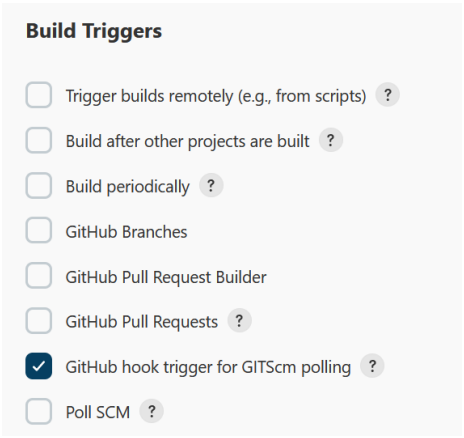
The all events specification means that github will sent notifications of all events that take place on github (new branch, commit, pull request, merge, etc), and Jenkins will be able to decide what to do with that information.

By setting the branch specifier to blank, all branches that are updated on the online repository get built by jenkins:

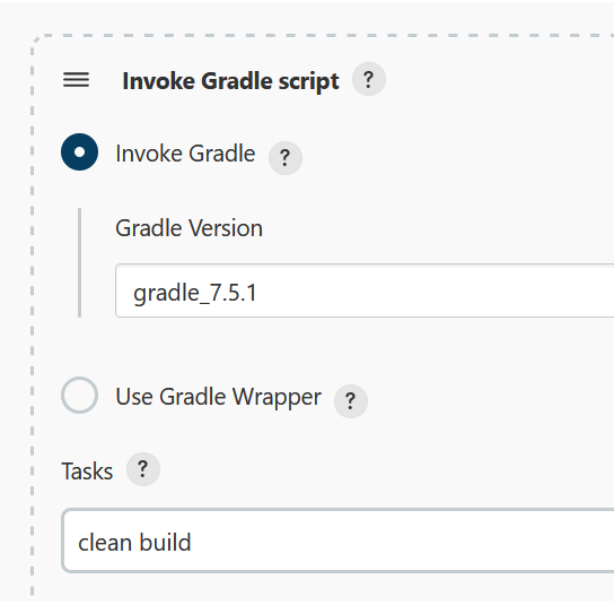


This allows our developers to ensure that their branch is building correctly before it is merged into main, or even before a pull request is created.

These builds are triggered by the github hook triggers below.

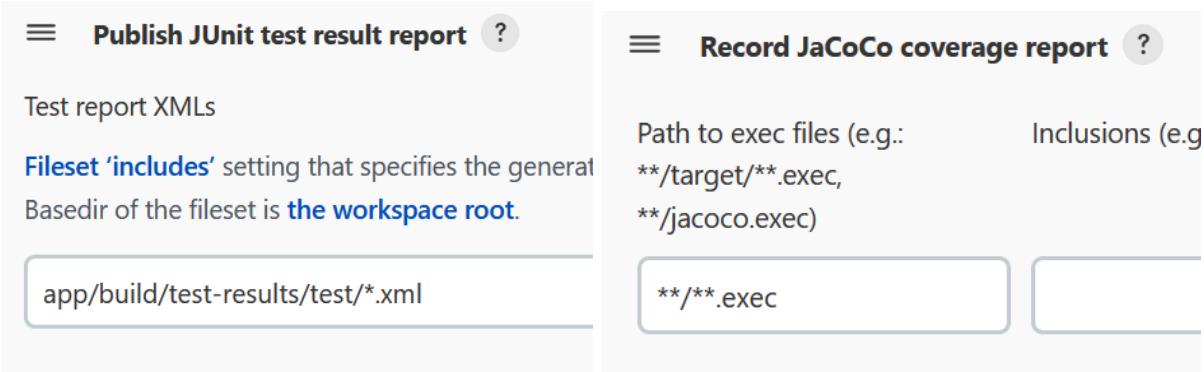


The configuration is set to use a specific version of gradle which stay consistent throughout the project, and the ‘gradle clean build’ command is used to ensure that previous build files are cleaned before each build.

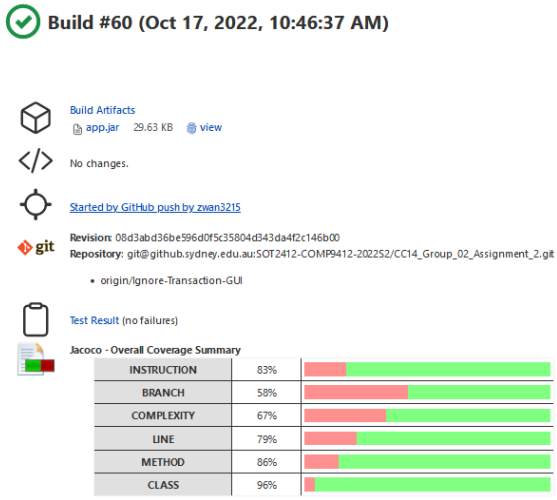


Artefacts are saved by Jenkins to be kept with the build by configuring the below settings:

Saves the final built app.jar file, and the build reports.
Junit tests and jacoco reports are saved with the following.



Builds:

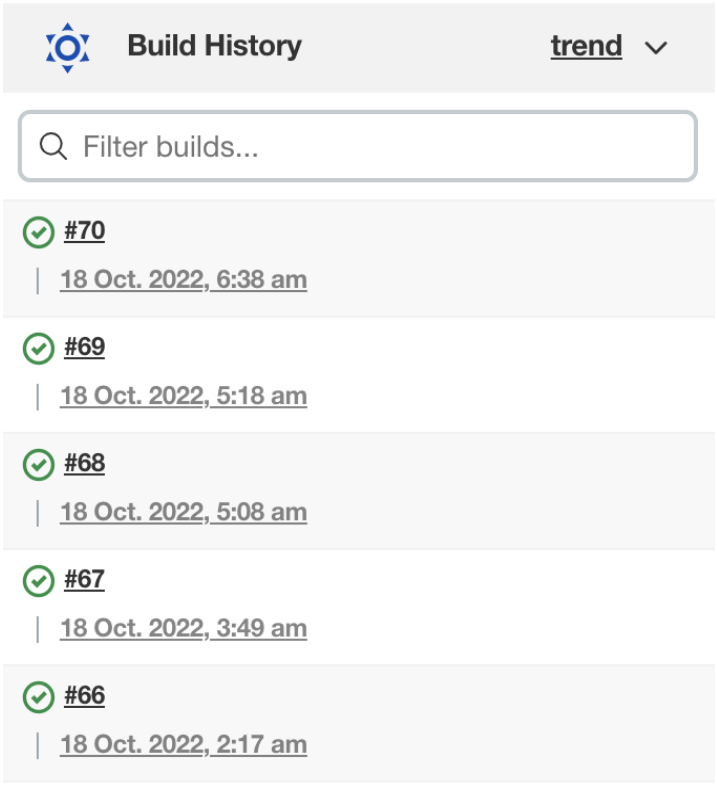


Here a successful build can be seen, with links to the artefacts stored with the build. This includes the built app.jar file, the jacoco test report, and the junit test results.
The test report gives a code coverage summary for the tests, and the test report will highlight any errors in the tests with the application. An example of a failed build is shown in the JUnit section.

Gradle

The primary purposes of Gradle are to automate the construction, run, and testing of programming. Users of this build software have flexibility. It is the process of cleaning up and compacting the code during compilation. It is the most agile way to create a project using a variety of technologies. From modest to large industry-based projects, the tool can be used.

It has the ability to run several files simultaneously. The source code can be added, removed, or edited with flexibility. By concentrating primarily on the project's functioning, it helps to maximise its worth. It aids in enhancing the functionality of the currently running code.

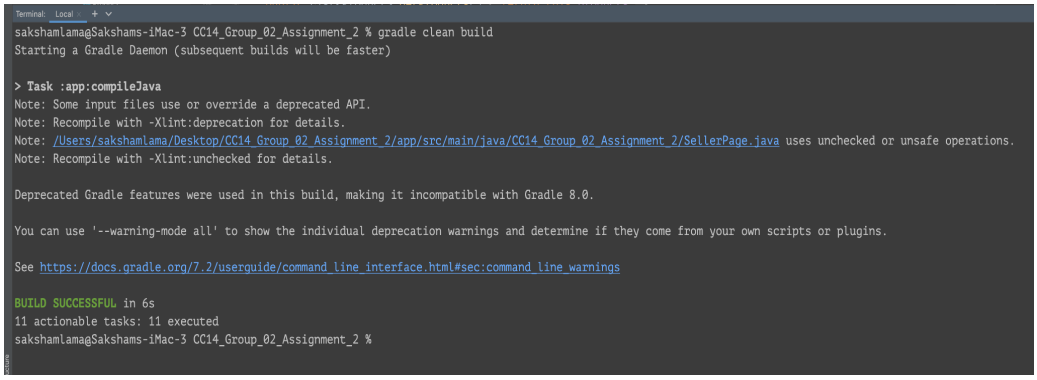


[List of the latest gradle build: Jenkins]

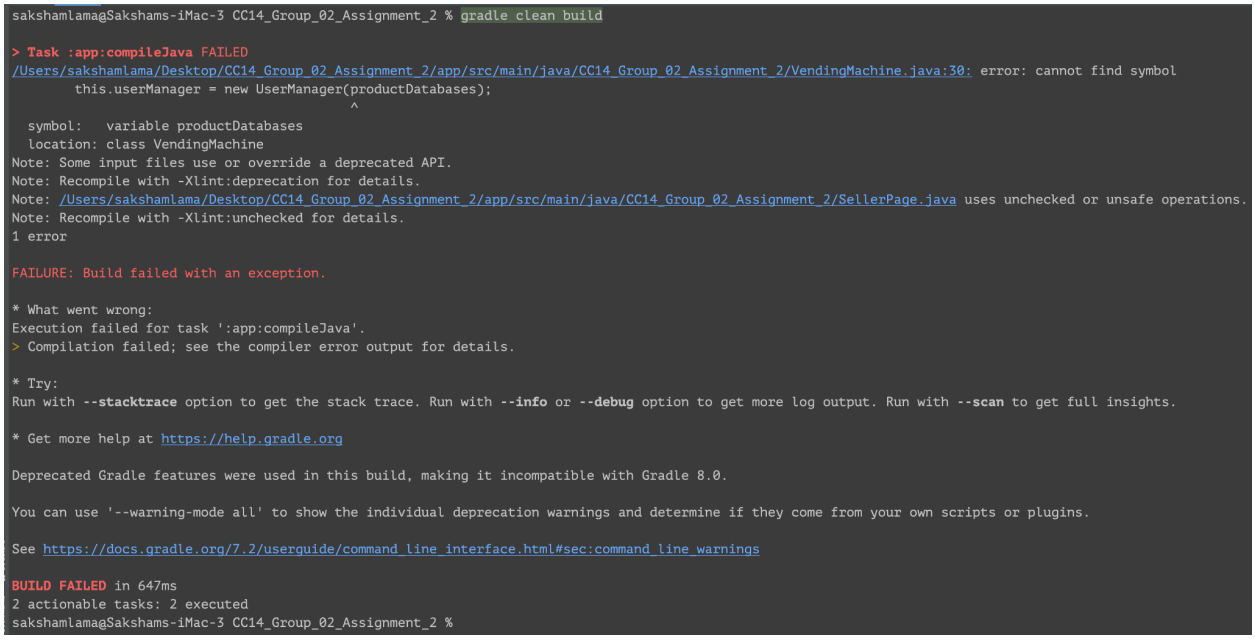
Relevant Gradle commands used:

gradle clean build: this command deletes the directory, where all the classes, files will be removed to ensure the program when it builds, is compiled using fresh data. This command behaves like the gradle build command, except for the deleting of the directory makes it different.

If the command executes all the tasks in order to build the project, the message at the end will be “BUILD SUCCESSFUL” in a certain number of seconds. Otherwise, the failed build will output, “FAILURE: Build failed with an exception”.



[gradle clean build: successful]



[gradle clean build: failed]

gradle test: This command will test all the test cases which are located in the test directory. If all the test cases present within the test suite succeed, the output will be displayed as “BUILD SUCCESSFUL”. Otherwise, the failed build will output, “FAILURE: Build failed with an exception”.

```
sakshamlama@Sakshams-iMac-3 CC14_Group_02_Assignment_2 % gradle test

Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own scripts or plugins.

See https://docs.gradle.org/7.2/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 1s
6 actionable tasks: 3 executed, 3 up-to-date
sakshamlama@Sakshams-iMac-3 CC14_Group_02_Assignment_2 %
```

[gradle test: successful]

```
sakshamlama@Sakshams-iMac-3 CC14_Group_02_Assignment_2 % gradle test

> Task :app:test FAILED

DefaultPageTest > TestProductToDisplayOutOfBoundsValue() FAILED
    org.opentest4j.AssertionFailedError at DefaultPageTest.java:77

36 tests completed, 1 failed

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':app:test'.
> There were failing tests. See the report at: file:///Users/sakshamlama/Desktop/CC14_Group_02_Assignment_2/app/build/reports/tests/test/index.html

* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output. Run with --scan to get full insights.

* Get more help at https://help.gradle.org

Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own scripts or plugins.

See https://docs.gradle.org/7.2/userguide/command_line_interface.html#sec:command_line_warnings

BUILD FAILED in 1s
6 actionable tasks: 2 executed, 4 up-to-date
sakshamlama@Sakshams-iMac-3 CC14_Group_02_Assignment_2 %
```

[gradle test: failed]

gradle --console plain run: This command will run the application. After the application is run, the program will display the command line interface showing the Vending Machine default page.

```
sakshamlama@Sakshams-iMac-3 CC14_Group_02_Assignment_2 % gradle clean run

> Task :app:compileJava
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: /Users/sakshamlama/Desktop/CC14_Group_02_Assignment_2/app/src/main/java/CC14_Group_02_Assignment_2/SellerPage.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

> Task :app:run
########## Welcome to Vending Machine ##########

===== Category: Drinks -> [0] =====
1. Mineral Water
2. Sprite
3. Coca cola
4. Pepsi
5. pp

===== Category: Chocolates -> [1] =====
1. Mars
2. M&M
3. Bounty
4. Snickers

===== Category: Chips -> [2] =====
1. Smiths
2. Pringles
3. Kettle
4. Thins
```

[gradle clean run: successful]

```
sakshamlama@Sakshams-iMac-3 CC14_Group_02_Assignment_2 % gradle clean run

> Task :app:compileJava FAILED
/Users/sakshamlama/Desktop/CC14_Group_02_Assignment_2/app/src/main/java/CC14_Group_02_Assignment_2/PurchasePage.java:38: error: cannot find symbol
                    vm.getInvalid();
                       ^
  symbol:   method getInvalid()
  location: variable vm of type VendingMachine
/Users/sakshamlama/Desktop/CC14_Group_02_Assignment_2/app/src/main/java/CC14_Group_02_Assignment_2/PurchasePage.java:42: error: cannot find symbol
                vm.getInvalid();
                  ^
  symbol:   method getInvalid()
  location: variable vm of type VendingMachine
/Users/sakshamlama/Desktop/CC14_Group_02_Assignment_2/app/src/main/java/CC14_Group_02_Assignment_2/DefaultPage.java:105: error: cannot find symbol
                vm.getInvalid();
                  ^
  symbol:   method getInvalid()
  location: variable vm of type VendingMachine
/Users/sakshamlama/Desktop/CC14_Group_02_Assignment_2/app/src/main/java/CC14_Group_02_Assignment_2/DefaultPage.java:110: error: cannot find symbol
            vm.getInvalid();
              ^
  symbol:   method getInvalid()
  location: variable vm of type VendingMachine
/Users/sakshamlama/Desktop/CC14_Group_02_Assignment_2/app/src/main/java/CC14_Group_02_Assignment_2/ProductPage.java:36: error: cannot find symbol
                    vm.getInvalid();
                       ^
  symbol:   method getInvalid()
  location: variable vm of type VendingMachine
/Users/sakshamlama/Desktop/CC14_Group_02_Assignment_2/app/src/main/java/CC14_Group_02_Assignment_2/ProductPage.java:40: error: cannot find symbol
                vm.getInvalid();
                  ^
  symbol:   method getInvalid()
  location: variable vm of type VendingMachine
/Users/sakshamlama/Desktop/CC14_Group_02_Assignment_2/app/src/main/java/CC14_Group_02_Assignment_2/TransactionPage.java:57: error: cannot find symbol
            vm.getInvalid();
              ^
```

[gradle clean run: failed]

build.gradle

An automation tool to build configurations, which will help in the project for creating the application.

plugins

- java: gain access java, which will provide libraries.
- jacoco: to obtain coverage reports .
- application: aids in creating a JVM executable. During the development phase, this executable will assist in launching the application locally.

repository

- mavenCentral: finds the dependencies present in the build.gradle file.

dependencies

- implementation: uses a json library (json-simple:1.1.1) to process json files for the implementation of the application
- testImplementation: uses JUnit Jupiter API (5.6.0), during the testing phase of the project.
- testRuntimeOnly: uses JUnit Jupiter (5.6.0) Engine also for testing.

run

- standardInput = System.in: takes user input via scanner.

jacocoTestReport

- Creates the coverage report and testing report on which test cases have passed or failed.

test

- useJUnitPlatform() - indicates to build.gradle that JUnit tool is being used for the testing of the application.
- Test.finalizedBy jacocoTestReport - indicates to build.gradle to use jacoco for testing in order to get the coverage report of the tests executed

application

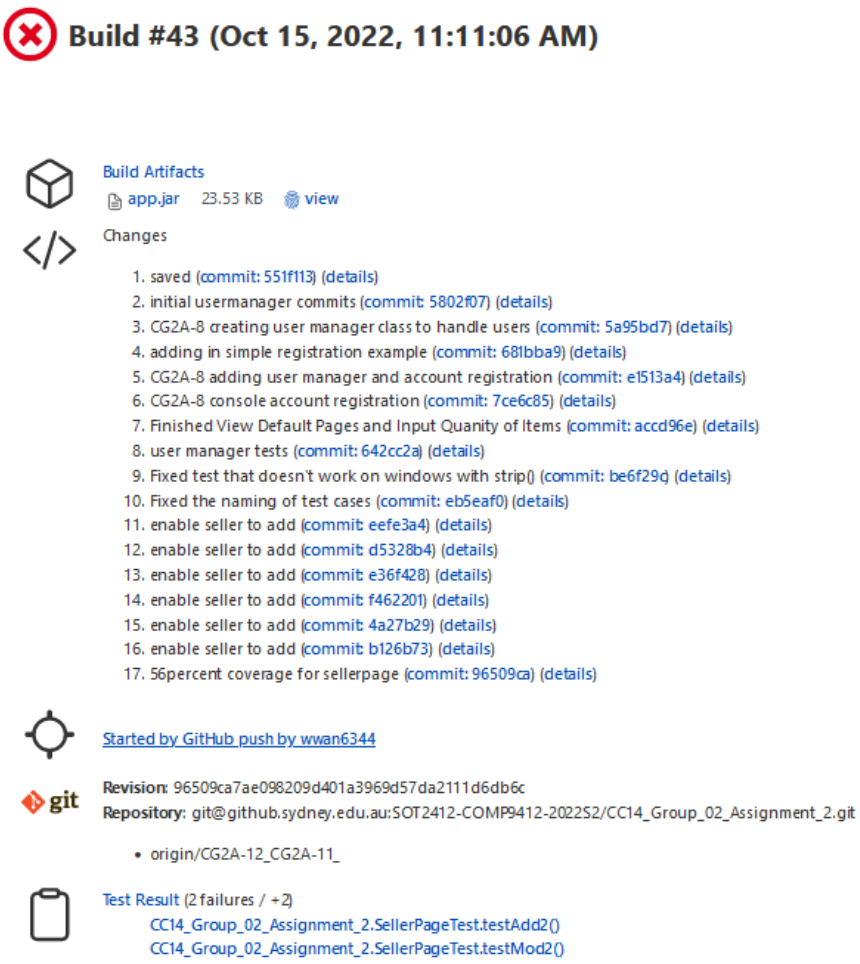
- mainClassName = CC14_Group02_Assignment_2.App’ - indicates to build.gradle on where the main method is located.

Junit:

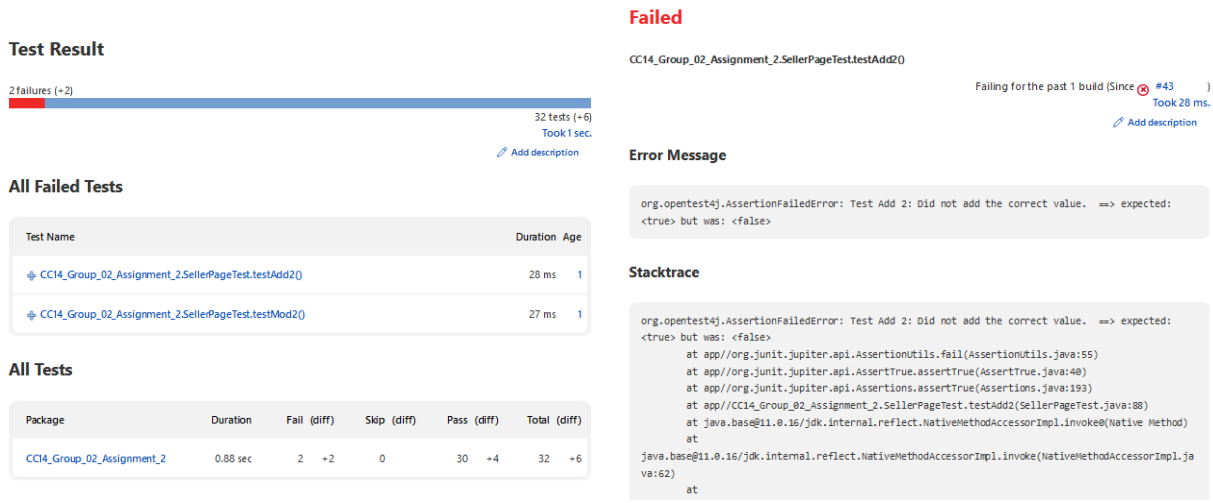
Junit is used as our code testing platform. Tests are created as per the table below, and are intended to be an indication that functions, methods and classes are working as intended throughout the continued development of the application

During the project, we had various tests fail at certain points. These were due to changed and incorrect file names for database files, changes to the number in inputs for certain functions, and string outputs changing across build environments.

A successful build is shown above in the Jenkins section, but here is an example of a failed build.



At the bottom, the tests which indicate the cause of the build to fail are seen.
Digging into this report, we can see the details on why these tests failed.



Test Result

2 failures (+2)

32 tests (+6)

Took 1 sec.

Add description

All Failed Tests

Test Name	Duration	Age
CC14_Group_02_Assignment_2.SellerPageTest.testAdd2()	28 ms	1
CC14_Group_02_Assignment_2.SellerPageTest.testMod2()	27 ms	1

All Tests

Package	Duration	Fail (diff)	Skip (diff)	Pass (diff)	Total (diff)
CC14_Group_02_Assignment_2	0.88 sec	2 +2	0	30 +4	32 +6

Failed

CC14_Group_02_Assignment_2.SellerPageTest.testAdd2()

Failing for the past 1 build (Since #43)

Took 28 ms.

Add description

Error Message

```
org.opentest4j.AssertionFailedError: Test Add 2: Did not add the correct value.  ==> expected:
<true> but was: <false>
```

Stacktrace

```
org.opentest4j.AssertionFailedError: Test Add 2: Did not add the correct value.  ==> expected:
<true> but was: <false>
        at app//org.junit.jupiter.api.AssertionUtils.fail(AssertionUtils.java:55)
        at app//org.junit.jupiter.api.AssertTrue.assertTrue(AssertTrue.java:40)
        at app//org.junit.jupiter.api.Assertions.assertTrue(Assertions.java:193)
        at app//CC14_Group_02_Assignment_2.SellerPageTest.testAdd2(SellerPageTest.java:88)
        at java.base@11.0.16/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at
java.base@11.0.16/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.ja
va:62)
        at
```

This error message and stack trace was used to identify the cause of failure for these tests, which was investigated and fixed for the next push to the repository.

A failed build in a CI/CD pipeline is not specifically a negative thing, as things change in software development, it merely indicates that something needs to be investigated, either resulting in changes to the code, or changes to the tests… As sometimes the tests are no longer working as needed due to code updates.

A full list of the current JUnit tests we have in place are below. Most of the tests are to confirm the usual required functioning of the method or class, but there are also several tests, where edge cases are tested to ensure that the application functions correctly at these times.
The test functions also make use of structures such as BeforeEach and AfterEach to setup and teardown test structures, or reset the testing suite to a specific state for each test

Code Coverage

Junit Test Cases:

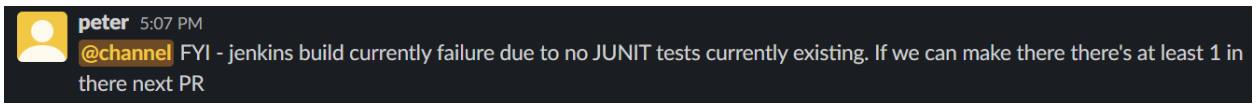
Test File	Test Name	Reason for Test
Apptest	testInvalidStringToFile	Test invalid string handling
	testValidStringToFile	Test valid string handling
CardManagerTest	TestCardDetails	Check that cards reported as valid/invalid correctly
DefaultPageTest	TestProductToDisplay	Test defaultpage cmdline interface handles good input correctly
	TestProductDisplayNegativeValue	Test defaultpage cmdline interface handles bad input correctly
	TestProductToDisplayOutOfBoundsValue	Test defaultpage cmdline interface handles bad input correctly
	TestProductToDisplayInvalidInput	Test defaultpage cmdline interface handles bad input correctly
ItemTest	ItemTests	Test that item class can instantiate, and be changed as expected.
ProductPageTest	TestCancelDisplayingProductDetails	Test that product page can be cancelled as expected
	TestPurchaseItemFromProductPage	Test purchasing item through cmdline product interface
	TestPurchaseItemFromProductPageOutOfBounds	Test product page handled bad input correctly.
	TestPurchaseItemFromProductPageNegativeValue	Test product page handled bad input correctly.
	TestPurchaseItemFromProductPageInvalidValue	Test product page handled bad input correctly.
ProductTest	TestParseProductDatabaseInvalid	Test invalid filepath handling for product DB
	TestViewDetailValid	Test products display as expected, have correct details from DB
	TestViewDetailInvalidProduct	Test attempt to display invalid product
	TestViewSimple	Test that product displaying item works as expected
	TestViewSimpleInvalidArguments	Test that viewing invalid product fails as expected

PurchasePageTest	TestCancelPurchasingItem	Test that purchase page can be cancelled as expected
	TestPurchaseItemFromProductPageLow	Test purchase page cmdline UI success
	TestPurchaseItemFromProductPageHigh	Test purchase page cmdline UI success
	TestPurchaseItemFromProductPageOutOfBounds	Test purchase page cmdline ui fails as expected
	TestPurchaseItemFromProductPageNegativeValue	Test purchase page cmdline ui fails as expected
	TestPurchaseItemFromProductPageInvalidValue	Test purchase page cmdline ui fails as expected
SellerPageTest	testDelete1	Test that products will not delete with invalid details
	testDelete2	Test that products delete with correct details
	testAdd1	Test that incorrect format for add fails as expected
	testAdd2	Test that add works as expected with correct format
	testMod1	Test that modifying product fails with incorrect format
	testMod2	Test that modifying product works with correct format
UserManagerTest	testLogin	Test that users can login with correct details
	registerUser	Test that users can be registered and then logged in
	failuretests	Test that registering doesnt work with existing user, login fails for invalid pw
	testRemoveAndRestore	Test that users can be removed from database and db restored from disk
VendingMachineTest	TestEmptyProductsListToDisplay	Test that empty product list displays error
	TestInvalidProductsListToDisplay	Test that invalid product list displays error
CardManagerTest	CardManagerTest	Test that the card and password checking function is working properly

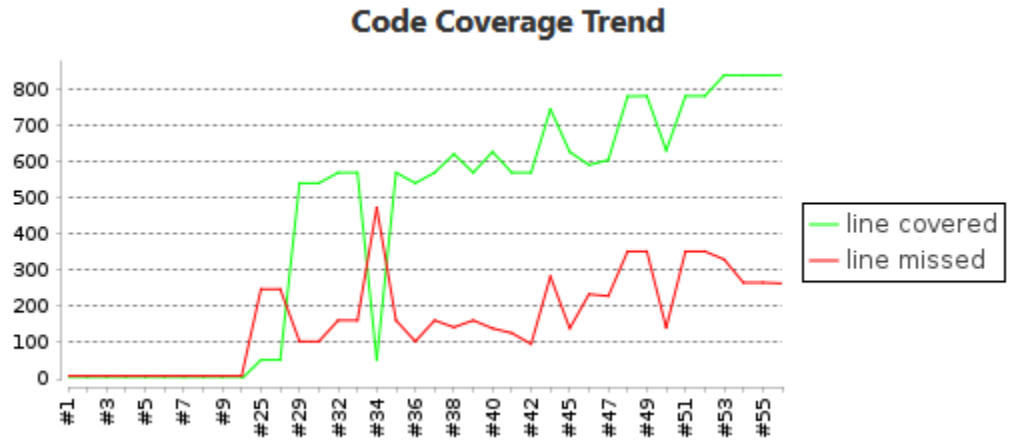
JaCoCo (Testing)

Initially, our builds were failing on Jenkins due to a lack of test cases. This can be seen here in our log:

```
BUILD SUCCESSFUL in 43s
10 actionable tasks: 10 executed
Build step 'Invoke Gradle script' changed build result to SUCCESS
Archiving artifacts
Recording test results
ERROR: Step ‘Publish JUnit test result report’ failed: No test report files were found. Configuration error?
Finished: FAILURE
```



Once some tests were implemented, we began to see reports of our test code coverage trends.



Here it shows the overview of our code coverage during testing, and how this has changed over the life of the project. Keeping this coverage high allows for us to know that each build of the application is still working as intended, as all important functions are being tested to ensure they are still running as expected.

name	instruction	branch	complexity	line	method	class
CC14_Group_02_Assignment_2	M: 954 C: 3628 79%	M: 103 C: 137 57%	M: 86 C: 170 66%	M: 263 C: 836 76%	M: 23 C: 113 83%	M: 2 C: 22 92%

Some of the classes and branches that are missed in our code coverage are user interface elements, and will be excluded as the project moves on.

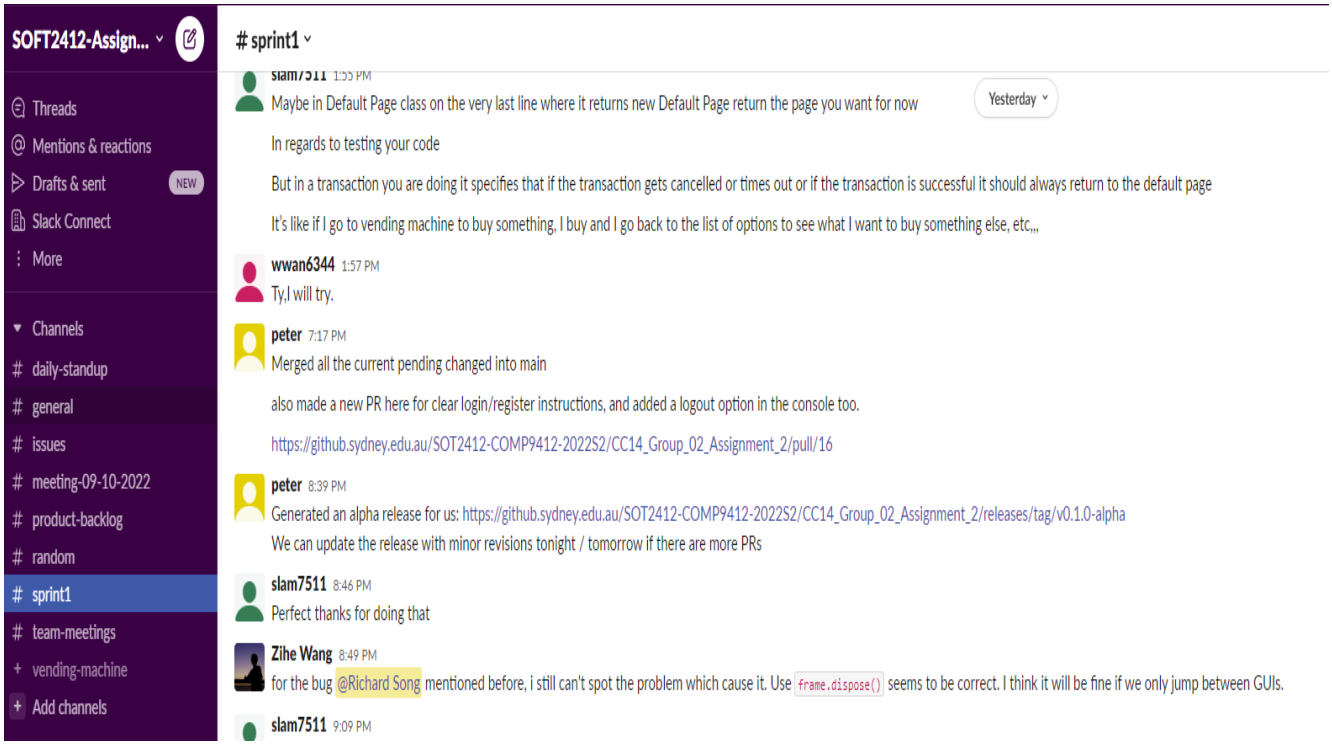
name	instruction	branch	complexity	line	method	class
App	**M:** 11 **C:** 14 56%	**M:** 2 **C:** 0 0%	**M:** 2 **C:** 3 60%	**M:** 5 **C:** 4 44%	**M:** 1 **C:** 3 75%	**M:** 0 **C:** 1 100%
AppTest	**M:** 0 **C:** 24 100%	**M:** 0 **C:** 0 100%	**M:** 0 **C:** 4 100%	**M:** 0 **C:** 10 100%	**M:** 0 **C:** 4 100%	**M:** 0 **C:** 1 100%
CardManager	**M:** 68 **C:** 74 52%	**M:** 4 **C:** 4 50%	**M:** 4 **C:** 5 56%	**M:** 23 **C:** 20 47%	**M:** 2 **C:** 3 60%	**M:** 0 **C:** 1 100%
CardManagerTest	**M:** 0 **C:** 40 100%	**M:** 0 **C:** 0 100%	**M:** 0 **C:** 2 100%	**M:** 0 **C:** 9 100%	**M:** 0 **C:** 2 100%	**M:** 0 **C:** 1 100%
DefaultPage	**M:** 118 **C:** 106 47%	**M:** 18 **C:** 12 40%	**M:** 12 **C:** 7 37%	**M:** 27 **C:** 32 54%	**M:** 0 **C:** 4 100%	**M:** 0 **C:** 1 100%
DefaultPageTest	**M:** 0 **C:** 176 100%	**M:** 0 **C:** 0 100%	**M:** 0 **C:** 7 100%	**M:** 0 **C:** 49 100%	**M:** 0 **C:** 7 100%	**M:** 0 **C:** 1 100%
ExcludeFromJacocoGeneratedReport	**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%
Item	**M:** 0 **C:** 77 100%	**M:** 0 **C:** 0 100%	**M:** 0 **C:** 12 100%	**M:** 0 **C:** 28 100%	**M:** 0 **C:** 12 100%	**M:** 0 **C:** 1 100%
ItemTest	**M:** 0 **C:** 95 100%	**M:** 0 **C:** 0 100%	**M:** 0 **C:** 2 100%	**M:** 0 **C:** 24 100%	**M:** 0 **C:** 2 100%	**M:** 0 **C:** 1 100%
LoginGUI	**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%

The report above shows which classes are currently being tested, and the below output shows exactly which lines are being hit by the currently implemented test cases. As can be seen below, the filenotfound exception has not yet been tested in our test cases. By ensuring high code coverage, it can be known that as the application changes, the application is well tested and breaking changes are identified quickly.

```
7:     private final File cardList;
8:     private final File cardSaver;
9:     private HashMap<String, String> currentCards;
10:    public CardManager(String cardListPath, String cardSaverPath) {
11:        this.cardList = new File(cardListPath);
12:        this.cardSaver = new File(cardSaverPath);
13:        this.currentCards = new HashMap<>();
14:        this.loadCardFromDb();
15:    }
16:    public void loadCardFromDb(){
17:        HashMap<String, String> cardDetail = new HashMap<>();
18:
19:        try {
20:            Scanner scanner = new Scanner(cardList);
21:•           while (scanner.hasNextLine()) {
22:                String line = scanner.nextLine();
23:                String[] cardLine = line.split(",");
24:                cardDetail.put(cardLine[0],cardLine[1]);
25:            }
26:            scanner.close();
27:
28:        } catch (FileNotFoundException e) {
29:            System.out.println("Unable to refresh DB from file. File not found.");
30:            //throw new RuntimeException(e);
31:        }
32:
33:        this.currentCards = cardDetail;
```

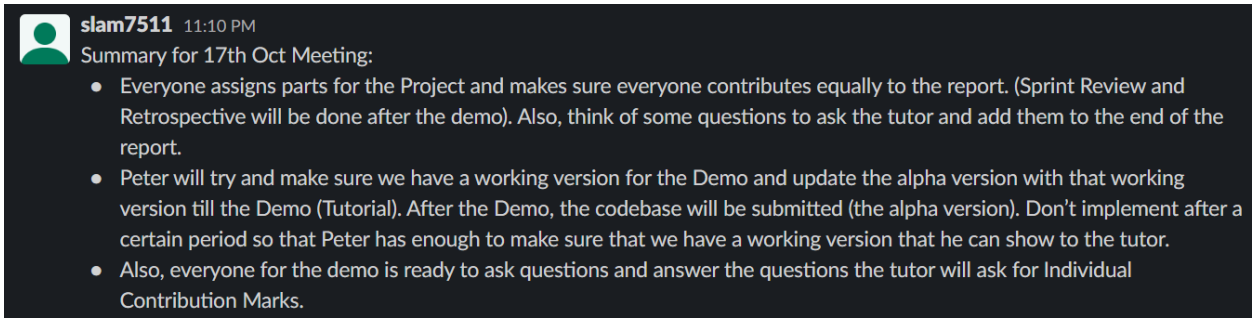
The use of tools to improve Group Collaboration – Communication(Qiuda):

In Assignment 1, we used Messager mainly as the communication tool. Because our messenger group chat only had one channel(the group chat itself), we tended to miss messages from our members from time to time because we crammed all information in there; in addition, having only one group channel also made it less efficient if we want to find what we discussed a few days ago on a specific issue. In Assignment 2, we improved our efficiency in communication by choosing ‘’Slack as our core collaborative tool. Slack allows us to open up channels for specific issues, and if we are all in the same group, we will automatically join the channels when created.

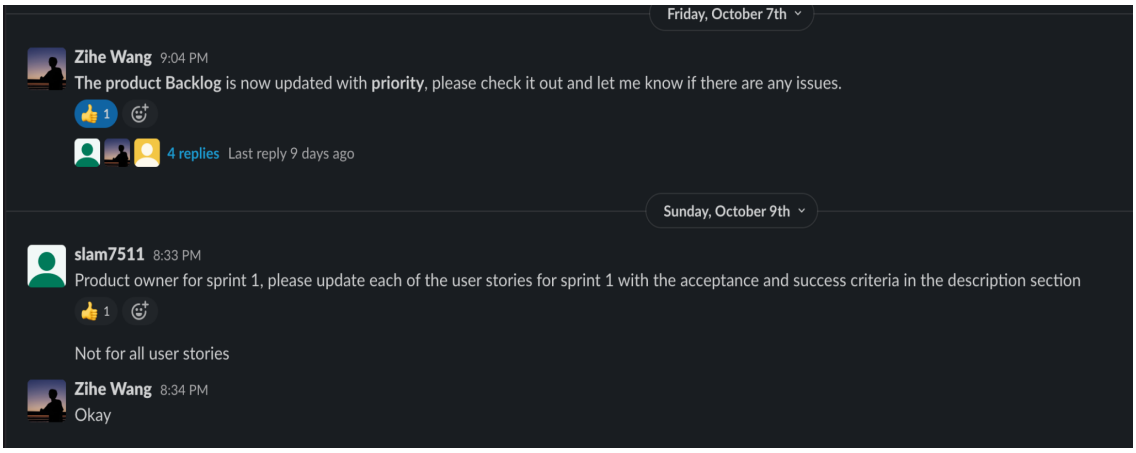


We have opened a number of channels, each having a specific purpose:

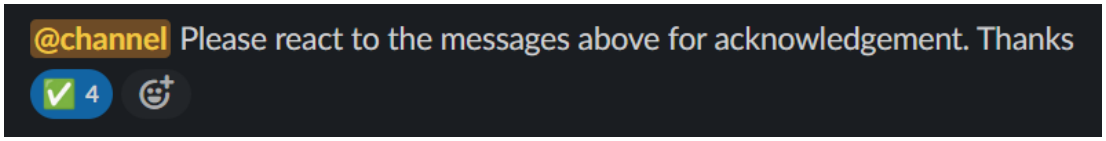
- Daily stand-up: the scrum master will make a note on the summary of each meeting and remind us what will need to be implemented afterwards.



- Product backlog: where the product owner and the scrum master will notify us of the assignment of team members onto tasks for each user story (This feature was mainly done on Jira, this channel only served as a reminder to make sure team members can get notified, and as a place where we can discuss where the current tasks allocation is reasonable etc…).



- **Team meetings:** where we arrange the times for team meetings and send zoom links. In addition to this channel, we used lettucemeet to fill in our availabilities before each meeting, so that we will what time best fit all of us efficiently by looking at the colour overlap. The Scrum Master was responsible for sending out links to each team member to fill out their availability for the team meetings.

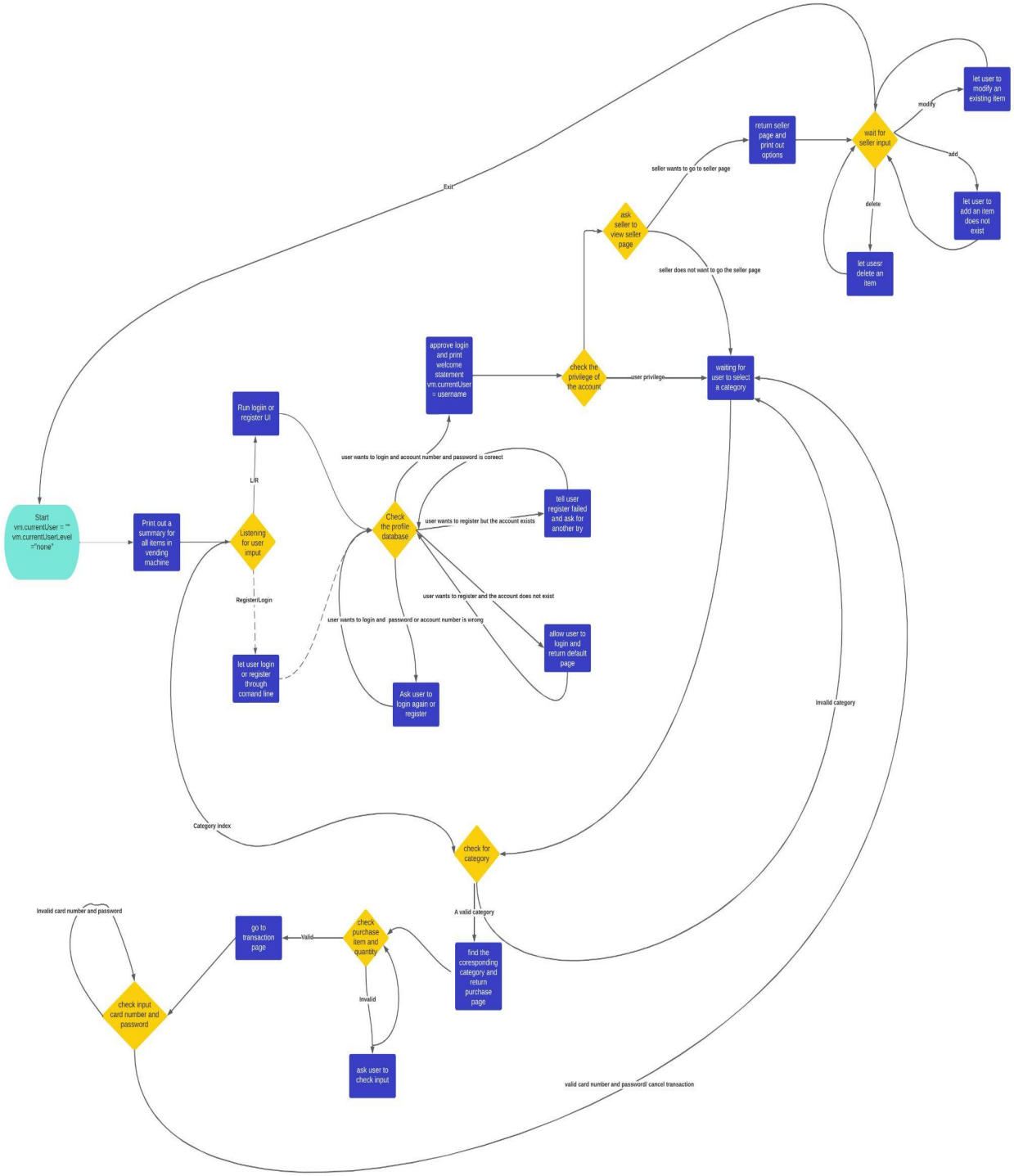


Message reactions were used to show that users had received the notification to fill out their availability.

Sprint1: where we ask questions in regards to the coding part for sprint1. In the future, we will open up a new channel for sprints 2 and 3 so that we will not get everything filled in one channel.

System Design (william)

Flowchart:



Questions for Client

The following questions were asked of the client during the demo. Answers to these questions will be integrated into decisions about the project moving forward.

1. Should the user logout automatically after a successful transaction?
2. Should the user password stored in the Vending Machine be encrypted?
3. Does the client prefer to use GUI or CLI or a mix of both?
4. In the seller page, should we show all items again after or before the modification is made?