Agile Software Development Practices (SOFT2412/COMP9412)

School of Computer Science

Group Project Assignment 1 – Tools for Agile Software Development

S2, 2024

Background

Agile software development is centred around teams and teamwork. In Agile team environments, a set of practices and principles should be followed to guide teams through the development process to build quality software products efficiently and effectively. Implementing Agile practices and principles in a software development project requires developing skills that fall in two main areas namely, technical tooling, and social interactions. In this unit, you should work in Agile software development teams to experience and develop both technical and social skills.

The goal of this group project assignment is to work as a team and use and apply Agile software development tools in a software development project context.

Main Requirements

In this group project, teams of 4 students (5 students might be allowed to deal with exceptions). Students *must be in the same tutorial/lab.* Instructions on how to register your group on Canvas will be announced later on. As a group, you will be required to build **a currency converter software application in Java**. All teams are required to use this software project as the starting point of their work. As a team, you will be required to develop specific features/functionalities of the currency converter application.

Following Agile practices, each team is required to work collaboratively and carry out software development activities to build a currency converter application. During the development, each team is required to use the set of software tools covered in this course following Agile development practices. Below are more details of the project requirements and deliverables for the assignment.

Part 0 - GitHub Repository Setup

Please log in to GitHub USYD Enterprise: https://github.sydney.edu.au/ using your UniKey and password to ensure that your GitHub USYD account is activated (i.e. you can log in successfully). After this, you need to log out and wait for a teaching assistant to invite you to our organisation (SOFT2412 - 2024S2). Further instructions will be posted in an Ed Announcement. Please follow the instructions to create your team and repository on GitHub USYD and create your own repository.

Part 1 - Agile Tools Setup

Each group must work collaboratively to set up a set of software tools that will help them prepare for the agile development environment. All these tools have been/will be introduced in the weekly labs including GitHub, Gradle, JUnit (including code coverage) and Jenkins. In your tool setup, you will need to meet the following requirements:

1. 1. There must be a single shared GitHub repository in the supplied GitHub organisation, for the entire group. Every member must contribute to this repository.
   2. Gradle must be used for build automation, and JUnit for automated testing. JUnit must be integrated with Gradle and a code coverage tool. For JUnit testing, code coverage must be at least 75%.
   3. Continuous integration should be done with Jenkins, which must be hooked up to GitHub. Both Gradle and JUnit must be integrated into Jenkins and run automatically for **every new commit** in a particular GitHub branch after Jenkins has been set up correctly.
   4. Jenkins must be integrated with GitHub via webhooks. You may also use NGrok for continuous integration with Jenkins. Alternatively, Jenkins may be hosted on public cloud service providers if your team is happy with that solution. Polling on Jenkins is also allowed instead of a webhook setup. Jenkins must show test reports for the application. **Only one member is required to set up Jenkins. As Jenkins will be covered in Week 5, you are only required to incorporate Jenkins into your project in Week 5.**
   5. Using an IDE like IntelliJ or Eclipse is recommended to assist you throughout the software development process.

Part 2 - Building the Currency Exchange Application using the Agile Tools

Each group will be required to develop a simple currency converter application in JAVA. All team members must collaboratively build this application using the agile tools they have setup in the previous part. All teams will have to implement following application requirements:

1. There are two user types in the system: admin and normal user (or user). The admin user can use all the functions of the application as specified below. The normal user can use certain functions as specified below.
2. The application should allow normal users to convert money from one currency to another. The application should allow normal users to input money amount and choose its currency symbol and the desired currency they want to convert to. The application should then carry out the conversion and display it with the current currency symbol.
3. Only the admin user can maintain the currency types and its exchange rates. Initially, your application must start with 6 currencies and its exchange rates. Exchange rates must be loaded along with the date when the application first runs. This can be a simple in-memory database (e.g., file). You may find the exchange rates from online sources such as XELinks to an external site..
4. The application should allow the normal user to display most popular currencies in a table of the currencies in which every cell represents the exchange rate

between the currencies (as shown in table 1). Most popular currencies must be 4 currencies which are specified and updated by the admin user. For example, in the below table, the conversion rate from AUD to SGD is 0.99. Exchange rates must be the most up-to-date rates as specified in the currency rates file/database.

From/To	**AUD**	**SGD**	**US**	**EU**
AUD	**-**	**0.99 ↓ (D)**		
SGD	**1.01 ↑(I)**	**-**		
US			**-**	
EU				**-**

Table 1. Exchange rates of most popular 4 currencies

5. The admin user can do all the above operations as well, i.e., convert currencies and display the popular currencies table.

6. The admin user can add new exchange rates daily by entering the date and exchange rate for that date of all currencies stored in the file. A complete history of the change exchange rates must be persistent including the rate and its date of addition. For example, if AUD to SGD was 0.99 and was added on September 4, 2020 and the admin added another exchange rate (e.g., 0.97 on September 5, 2020), these rates must be persisted for the below functionalities.

7. The admin can also add new currency types in addition to the existing currencies and its conversion rates. The most up-to-date currencies should be used in currency conversion and in the most popular currencies table.

8. When the user chooses to display the 4 x 4 common currencies table, if the new rate has decreased as explained in (6), this must be indicated by a down arrow (or the letter D) as shown in *table 1* (i.e. the rate of conversion from AUD to SGD has been decreased compared to previous rate, 0.99 to 0.97). You may choose between arrow signs and letters (e.g., (D) for decrease and (I) for increase).

9. Given the history of the conversion rates (based on admin input), the user can print summary of the conversion rates of 2 currencies they choose within a specific duration (start and end dates). This includes all conversion rates, average, median, maximum, minimum and standard deviation of the conversion rate of the 2 currencies during the specified start and end date.

10. The application must implement a simple user interface to allow users to select from the different available functions (stated above) with clear instructions and messages to guide the user on what should be done, corrected in case of incorrect selection/input and what options available on current or other functions. The user interface **can be** a GUI (java swing, javaFX etc.) or a command-line based Currency Exchange and instruction/messages. You are encouraged to have GUI, but not forced to do so.

11. All the above functionality must produce correct output as a result of using the Agile development tools.

12. Wherever certain function behaviour or requirement is not specified above, you as a team can decide on how to handle/implement such behaviour as far as that does not lead to any errors or produce inconsistent or wrong output.

The application design can use **any UI (terminal or GUI)** for its functionality. You can decide on the application design/architecture; both text files (txt, JSON, XML, etc.) and persistent databases (SQLite, MongoDB, MySQL, etc.) are acceptable options for storing your application data. The software must always produce correct output and maintain the correct and consistent state of all included entities.

Each group must carry on the development of the Currency Application using all the tools they setup in part 1. Teams must demonstrate the proper use of these tools and practices to ensure efficient and effective development as well as delivery of correct application behaviour. This includes:

- **GitHub collaboration** (branching, merging and conflict resolution). Make releases and version your software properly on GitHub.
- **Build automation** triggers successfully with appropriate reporting
- **Automated tests** trigger successfully with appropriate test/code coverage and reporting
    - You must make sure your unit tests have good code coverage. You must write unit test cases that result in more than **75% code coverage**.
    - Tests must be such that it covers all edge cases and normal cases as well. Tests must be written for the functionality of the application (listed above).

Each group must also apply relevant continuous integration practices, covered in this course, in their application development. All groups must show evidence (in their report and demo) of the proper use and application of the above-mentioned agile tools and development/CI practices.

Technical Report

Each group must prepare a technical report that records evidence of the above development activities (besides the actual GitHub repo. logs and reports which will be checked as part of the marking). This specifically includes:

1. **Explain how the group collaborated to complete the development of the Application**. This should include individual and group contributions, and group communication (recorded minutes for all meetings). Each *team member* must explain how they contributed towards implementing the features of the application.
2. **A README file or a page in the GitHub wiki** explaining how to run the program, and how to test it. You may also include other instructions as to how to contribute/collaborate on the codebase.
3. **GitHub collaboration**. You must explain how GitHub was used for building the application. This might include Project boards, issues, pull requests etc. and how they were used in favour of completing the implementation of the application.
4. **Explain how Gradle was used**. This may include any extra tasks/dependencies used for the application including brief comments on how those extra tasks/dependencies helped you build the application.
5. **Explain how Jenkins was used** and the work carried out by the team to automate the CI pipeline including GitHub integration, automated build and testing, generated outputs and CI practices.

6. **Code coverage report for the final commit**, including JUnit results for the commits. All Junit test cases that are written must be documented and explained (i.e. why the unit test was chosen and what it tests (regular input, edge cases etc). Document any tests that may have failed. You will also need to explain what the test coverage report is displaying.

Project Demonstration

- Each group must demonstrate their project work. You will have to demo your group work and individual contributions to your tutor during week 8 tutorials. The schedule and any information will be shared by your tutor before week 8 tutorials.
- Each group will have 15-20 minutes to demonstrate the functionality of their Currency Exchange application and work done through your GitHub repo, automated builds, automated tests and Jenkins.
- All team members must be present in the project demo. Also, every member must demonstrate their contributions to the project besides the group work. Your tutor will ask you questions about the entire collaboration and the tools used. Every team member will be asked questions about the development work done by the team and by themselves.

More details about the demonstrations will be announced closer to the submission deadline.

Group Member Contribution

- This is a group project assignment, and the assessment will be based on individual contributions to the group work as per the above requirements. The whole project work cannot be handled by one group member or individually. It must be carried out as a group and every member must contribute to the technical development and use of the Agile tools and setup described above.
- If members of your group do not contribute sufficiently you should alert your tutor as soon as possible. The course instructor has the discretion to scale the group’s mark for each member as follows:

Level of Contribution	**Proportion of final grade received**
No contribution	0%
Poor contribution (based on available evidence from GitHub organisation, logs from used tools and other documentation)	1% - 49%
Partial contribution (based on available evidence from GitHub organisation, logs from used tools and other documentation)	50%-99%
Full contribution	100%

Deliverable and Submission Guideline

- Each team must submit their technical report as PDF (one submission per group) through the provided link on this Canvas page. All group members must sign the assignment coversheet Links to an external site.. Download the assignment coversheet and attach it to the first page of the technical report. (reports that do not include the signed assignment cover sheet will not be marked).

- Each team must submit their latest source code version of the Currency Exchange Application as a zip file with all your project files. A separate submission link will be supplied).

Marking Guide

The above requirements should help you to prepare and structure your group project report and demo. The marking criteria will be based on the above requirements. The below marking criteria should help you to prepare and structure your group project report and demo besides the above assignment requirements.

1. **Quality of Version Control of the Application Source Code (GitHub) (15%)**

Clear and sensible explanation of the work carried out by the team (what, how and why). This should be demonstrated through the project report and the group demo. including:

- Collaboration model of GitHub repository (from GitHub organisation). Every member must have contributed to the Currency Exchange application development through this repository.
- GitHub setup and git commands
- Examples of representative of GitHub, e.g. merging, branching, and conflict resolutions
- Examples of the steps to deal with these issues in the previous point
- How the group collaborated using GitHub to complete the project

All the above must be supported by appropriate evidence where applicable (e.g., screenshots, outputs, logs). The GitHub organization will be audited during the demo and while marking the report to ensure consistency.

2. **Quality of Build Automation (Gradle) (10%)**

Clear and sensible explanation of the work carried out by the team (what, how and why) to automate the Currency Exchange Application build. This should be demonstrated through the project report and group demo including:

- Relevant Gradle commands used
- A brief explanation of the build.gradle file
- Explanation of the results/outputs obtained from relevant Gradle commands

All the above must be supported by appropriate evidence where applicable (e.g., screenshots, outputs, logs).

3. **Quality of Testing (JUnit) (20%)**

Clear and sensible explanation of the work carried out by the team (what, how and why) to automate unit tests. This should be demonstrated through the project report and group demo including:

- How unit tests are run
- How code (test) coverage is incorporated
- JUnit test files describing test cases to test the program
- How and why test cases were designed
- Representative results/output from JUnit tests
- Results/output from code/test coverage

All the above must be supported by appropriate evidence where applicable (e.g., screenshots, outputs, logs)

4. **Quality of Continuous Integration (Jenkins) (20%)**

Clear and sensible explanation of the work carried out by the team (what, how and why) to automate the CI pipeline. This should be demonstrated through the project report and group demo including:

- How Jenkins was integrated with GitHub
- Build and test coverage on Jenkins
- Jenkins setup including webhook, automatic build/test, Jacoco test report displayed on Jenkins, and Jenkins archived outputs (jar and test results)
- Explanation of representative Jenkins outputs
- Adopted CI practices

All the above must be supported by appropriate evidence (e.g. screenshots, outputs, logs).

5. **Quality of Application Development (25%)**

Clear and sensible explanation of the work carried out by the team (what, how and why) to implement the requirements of the Currency Exchange Application. This should be demonstrated through the project report and group demo including:

- Description of how the group collaborated to complete the application using the above tools set up, individual and group contributions and communication (you can reference other sections/parts if already presented previously in your report)
- Overview of the Currency Exchange Application design (e.g., class diagrams, sequence diagrams, or any format)
- How the Currency Exchange application produces correct output/behaviour with various inputs and adheres to the functional requirements during the demo (recorded and the live Q&A)

All the above must be supported by appropriate evidence where applicable (e.g., screenshots, outputs, logs).

6. **Quality of Demonstration (10%)**

- The demo correctly covers the key important scenarios listed above
- The Currency Exchange application adheres to the functional requirements and produces correct output during the live Q&A
- The team is able to sensibly answer questions, explain and justify the work they’ve done in the project, and the results produced during the live Q&A

7. **Quality of individual contribution**

- Each team member made a fair and enough contribution to the setup of Agile tools and the development of the Currency Exchange application. All members collaborated and worked as a team. This should be evident by the GitHub repository and provided evidence in each of the development activities above.
- Each team member should clearly indicate their contribution in the report (table of roles and contributions and reference to the sections in the report). They should also each include a short statement to describe how they have contributed to the outcomes achieved by the assignment.

The mark for each member will be scaled based on their contributions to the project as per the contribution percentages described in the assignment requirements (0%-100%).

Academic Integrity

While the University is aware that the vast majority of students and staff act ethically and honestly, it is opposed to and will not tolerate academic integrity breaches and will treat all allegations seriously.

Further information on academic integrity and the resources available to all students can be found on the academic integrity pages on the current students' website: https://sydney.edu.au/students/academic-integrity.html.

Further information on research integrity and ethics for postgraduate research students and students undertaking research-focussed coursework such as Honours and capstone research projects can also be found on the current students' website: https://sydney.edu.au/students/research-integrity-ethics.html.

Compliance Statement

In submitting this work, I acknowledge I have understood the following:

- I have read and understood the University of Sydney's Academic Integrity Policy 2022.
- The work is substantially my own and where any parts of this work are not my own I have indicated this by acknowledging the source of those parts of the work and enclosing any quoted text in quotation marks.
- I have acknowledged any assistance provided in preparing the work including the use of copy-editing, proofreading, and automated writing and drawing tools (including artificial intelligence (AI), reference generators, translation software, grammar checkers, but not spell checkers).
- The work has not previously been submitted in part or in full for assessment in another unit unless I have been permitted by my unit of study coordinator to do so.
- The work will be submitted to similarity detection software (Turnitin) and a copy of the work will be retained in Turnitin's paper repository for future similarity checking. Note: work submitted by postgraduate research students for research purposes is not added to Turnitin's paper repository.
- Engaging in plagiarism or academic dishonesty in coursework will, if detected, lead to the University commencing proceedings under the Academic Integrity Policy 2022, and the Academic Integrity Procedures 2022.
- Engaging in plagiarism or academic dishonesty in research-focused work will lead to the University commencing proceedings under the Research Code of Conduct 2013 and the Academic Integrity Procedures 2022.
- Engaging another person to complete part or all of the submitted work will, if detected, lead to the University commencing proceedings against me for potential student misconduct under the University of Sydney (Student Discipline) Rule 2016.