✅ Sprint 2: Validation and Delivery - Assessment Outline

Value: 25% of the Course Mark

Due: Week 7 Friday, 5pm

❓ **What is this sprint aiming to achieve?**

1. **Beyond Deployment**. When you deploy your software, that's just the beginning. *All software development is maintenance*. In this Sprint you'll need to co-ordinate the **delivery** (releasing of software to customers in a product environment) and monitoring of your service.
2. **Testing the Ecosystem**. You'll need to develop a microservice which has the sole purpose of testing at a defined level of abstraction within the ecosystem.
3. **Taking Engineering Responsibility**. This ecosystem is yours - so we're going to give it to you to determine what you want to work on as a team, and work with other teams to see it through. In industry, this is how engineering teams operate on large codebases.

1. Task 🪗

In this sprint, you will be expected to:

1. **Document** and release your service publicly to other teams
2. Define a **delivery strategy** that allow you to release your service into the production environment. This includes a mechanism to monitor your service to make sure it is running.
3. Define a **testing strategy** and generate a report that includes useful information about the service quality (e.g., test cases, results, logs of any issues, overall result).
4. Integrate the running of your testing into the **Continuous Integration pipeline** that runs whenever a service is deployed into the ecosystem.
5. Complete any other work you as a team wish to achieve, which can include completing leftover work from Sprint 1 and preparation for Sprint 3.

2. Delivery and Documentation 🚚

In Sprint 1, you created multiple microservices with one (or more) service intended to be publicly released to other teams. 🙌 Collaboratio n Page shows the services that other teams are developing, you should post the necessary info on this page for other teams to contact you for using your microservices.

Can you modify your services that could be useful for other teams? Can you provide evidence that it is working? Even if it is just an assumption and no teams are using your services, are you able to describe how your services can be interoperable with different types of applications or datasets? How would you implement this interoperability?

Please write on the Confluence page about any of your microservices for which you have considered the design of interoperability.

Afterwards, you will need to define a **delivery** strategy for releasing your service/application into the staging and production environments.

As an option, you can use the AWS **CloudWatch** platform that gives you information on the state of your application.

4. Testing

You will need to define and develop testing services (possibly by defining additional microservices) that should perform a series of automated tests on any component of your architecture.

An overview of key testing techniques will be presented during week 5 lectures. You can perform tests at a lower level of abstraction (e.g. just testing an individual service), a higher level of abstraction, or even some in-production tests.

To document your testing strategy, create a dedicated section on your Confluence pages. Provide a clear description of the tests you have implemented, including their purpose, the level of abstraction, and the specific components or services they target. Include relevant evidence such as explanations, links to code repositories, and screenshots of test results to support your work.

Reading materials that you might find useful:

Getting started with testing serverless applications | Amazon Web Services

How to test AWS Serverless Microservices - the proper way?

3. Software Quality

As you develop your testing service, you will need to integrate it into the **CI infrastructure**. You'll first need to think about where to do this - is it the CI pipeline of another service? What if you're testing multiple services? What if it's in-production tests?

Your tests will need to run and fail the pipeline if the tests fail, stopping further deployments if applicable. The output on GitHub Actions should be sufficient to debug any problems.

> Your service's **testing report** will need to be downloadable as an **artefact** from the respective GitHub pipeline. It can be as simple as a text file, though a PDF would be more desirable.

4. Marking Criteria

Criteria	Description
Delivery and Documentation (30%)	• Have microservices been designed for interoperability? • Has the team communicated with other teams in order to facilitate interoperability?? • Has the team provided useful info on the collaboration page? • Does documentation show meaningful and useful information? • Has a sensible deployment strategy been discussed? • Has an adequate strategy for monitoring the application been set up ?
Testing (40%)	• Does the service have a clearly defined level of abstraction to test at? (unit, contract integration, component, E2E) • Do the tests properly verify a service(s)? • Are the tests logically structured and well written? • Has the environment of the test service been defined? • Has the testing report been generated?
Software Quality (20%)	• Has Continuous Integration been set up? • Does the CI include all relevant aspects? • Has the running of the tests been incorporated into the CI process for the ecosystem appropriately? • Is the code well written and styled?

5. Submission

Place a link to your repository inside a Confluence page called **Codebase**.

For this sprint, we will take the state of your Jira board, Confluence space and Git Repository at the deadline as your submission. You do not need to run any submission commands.

Late submissions will not be accepted.

Applications for Special Consideration and ELS assessment accommodations will not include extensions as this is a group project with no scope for extending deadlines. The course authority will determine an appropriate adjustment in cases where a Special Consideration request is approved, or a student has an equitable learning plan. Students in either of these positions should email se3011@cse.unsw.edu.au.

6. Plagiarism

The work you and your group submit must be your own work.

The use of code synthesis tools, such as GitHub Copilot, is not permitted on this project.

The use of ChatGPT and other generative AI tools is not permitted on this project.

Relevant scholarship authorities will be informed if students holding scholarships are involved in an incident of plagiarism or other misconduct.

You are welcome to seek help from other students taking the course, however, all work you submit must be your own. The use of external contractors to complete the project or your part of the project is not permitted.

During your mentoring sessions, your mentor will be using the opportunity to interview you and ensure you are contributing to the project.