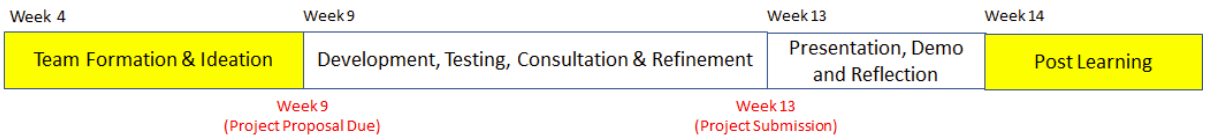
Executive Summary

Requirement Title	**Design and Implementation of an Enterprise Solution Using a Microservices Architecture**
Team Size	5-6 persons
Grade Percentage	35% of all assessments
Due Dates/Time	
Proposal	**Before the Week 9 class:** Your team must have prepared a proposal to describe your proposed scenario (e.g. Taxi Application, Food Delivery, ERP Gantry, etc.) and draft solution design. The details of the proposal are under the Deliverables-Proposal section. **During the Week 9 class:** Each team gets **20 minutes** to present and discuss your scenario and draft solution design with the instructor(s) face-to-face in the classroom.
Presentation, Report, and Executables	**30 minutes before the start of your respective class in Week 13:** E.g., 0815 class □ 0745 hrs 1530 class □ 1500 hrs **Submit** the following items on eLearn: ● **Presentation Slides** ● **Project Report** ● **All of your code with required documentation** ● **A video recording of a demo of your solution** See the corresponding Deliverables subsections for details. **During the Week 13 class:** **Present & demonstrate** your scenario and solution (**within 15 minutes**) followed by **Q&A**
Submissions	We will take and mark **the latest submission on eLearn**, and apply **heavy penalties for late submissions**. See Deliverables-Submissions section for details.
Intra-Team Peer Evaluation	**On a CASE-BY-CASE basis** ● If there is any issue(s) within the team, you **MUST** let us know the problem(s) **AS AND WHEN** the problem(s) surfaces. If you have issues that persist throughout the project but inform us about it only near the presentation of the project, we are afraid that there will **not be much we can do to resolve the issues at that late stage**. ● There is no official intra-team peer-evaluation for every team. We will investigate on a **CASE-BY-CASE** basis and make appropriate decisions on penalties to the member(s) who did not do their part for the project.
Inter-Team Peer Evaluation	Each team will be given an evaluation form in class in Week 13 to evaluate the presentations of other teams.

Overview of project stages:



Learning Outcomes

With the emergence of new technologies and evolution of existing ones, organizations are changing the way they build enterprise solutions. Rather than building monolithic applications, the current emphasis is on building solutions by leveraging the microservices architecture. This approach to building solutions using microservices follows the Service Oriented Architecture (SOA) paradigm, where applications are structured as assemblies of loosely coupled (micro)services organized in various layers.

In this project, your team must follow **a methodical approach** to come up with **an enterprise solution** based on the **microservices architecture** for **a business scenario** (e.g., Taxi Booking, Food Delivery, ERP Gantry, etc.) using **appropriate technologies and tools** (e.g. Python, Flask, REST, RabbitMQ, WampServer, Docker, Visual Studio Code, etc.).

Minimum Technical Requirements

- Your business scenario and solution must implement and showcase your **3 most interesting* user scenarios**
    - Sample user scenarios and data entities involved for Taxi booking

      (1) Passenger sends a booking request…
        - Passenger, Booking, …

      (2) Driver is notified and accepts a booking…
        - Driver, Booking, Notification, …

      (3) Passenger pays for a ride…
        - Passenger, Payment, Booking, …

      ** E.g. user scenarios that are NOT interesting*
        - *User registers for an account*
        - *User logins*
        - *User updates his profile*
- Your scenario and solution must have **a minimum of 3 (simple) microservices for 3 different data entities**.
    - You may use any programming language and/or framework to implement the microservices.
    - Your presentation and report should clearly explain the design and implementation of the user scenarios, and the rationale of the technologies used.
- **At least one of the microservices should be reused** across different user scenarios.
    - E.g., the Passenger microservice is used in both (1) and (3) user scenarios above.
    - Your presentation and report should clearly explain the reuse cases and their benefits related to the scenarios.
- **At least one use of an external service** (see sample service providers near the end).
- **At least two user scenarios where a (micro)service orchestrates or initiates a choreography of other (micro)services**.
- Each microservice can have more than one operation (CRUD).
- Each microservice must have exclusive access to its own data store (if it has a data store). A data store can be a file or a DB table or cloud storage.
    - **At least one microservice has to use a DB** as its data store.
- Your solution must **use HTTP** communication between some microservices.
- Your solution must **use AMQP**-based communication between some microservices.

- Your application must have **at least one web based Graphical User Interface** which can be developed in any programming language or framework (e.g. HTML/Javascript/Vue/jQuery, etc.)
- **Use JSON data** in some of your microservices and/or Web UIs.
- **Use Docker** in your solution in a way suitable for your scenario.
- **Use Docker Compose** to deploy your microservices in a local environment (a deployment in the cloud is NOT required) .

What is an interesting scenario? Is my project scenario good enough? Is it interesting enough?

- Make sure your project scope ticks all the "Minimum Technical Requirements". That covers the basic stuff.
- To make your scenario more interesting, some possibilities (not exhaustive) are:
    - E.g. Place Order for Books - decrease book's stock in warehouse, create Order, request for shipping. Say, the destination is a country not supported (no air services because of COVID), we have to cancel the whole order. You need to "rollback" the order creation, put the books back into the warehouse. Such twists in business logic and handling of business errors makes a scenario interesting.
    - In a delivery service, it may be unlikely to get a reply from a driver immediately, thus, asynchronous request-reply is the most appropriate pattern. If your project has a valid business scenario that requires async RR (or Choreography), it will be more interesting.
    - The above examples are not covered in labs, that is what makes them interesting.

Beyond-The-Labs (BTL) Requirement

- You are recommended to implement and demonstrate something that is **beyond the labs** (something not covered in the labs or something considered as advanced topics in the labs that you need to research and implement on your own effort)
    - See below for examples beyond the labs and examples NOT counted as BTL.
    - Consult with your instructor(s) on what is considered BTL if your idea is not covered in these examples. Do not simply assume it will count towards this component.
- Your presentation and report should clearly justify that the uses of the BTL are indeed beneficial for your scenarios.
- The grade for this requirement is **up to 3 marks**, as part of the total 35 marks for the project.

General Guidelines

- Your business scenario can be imaginative or futuristic, but it would be better to have justifiable linkage to a real-world problem and your proposed solution may address some real-world needs or pain points. It must be legal and morally suitable for use in academic settings; e.g., scenarios suggesting illegal gambling will not be allowed.
- State your simplification and/or assumptions, if any, for your scenario and solution.

- Use simple code for the user interface. Functionality is more important than the look-and-feel. For example, a fancy UI is nice, but if it does not help you in any way other than just looking pretty, it is not going to earn you a lot of marks (or any at all).
- You may reuse any code in the labs with and without modifications.
- You may use any technology and tool to implement your solution, not limited to the technologies and tools covered in the course. Whether those technologies and tools are considered as going beyond the labs depends on how you use them and how you justify their uses for your scenarios in your presentation and report and is at the discretion of the instructor(s).
- To cater for some unfortunate situations (e.g., your demo works before the class but it fails during the actual demo in class), please prepare a video recording of your demonstration before the class as a backup. Also, try to have the setup on more than one machine for backup.

Deliverables

Proposal

You must prepare a proposal **before** the discussion session with your instructor(s). Discussing with the instructor(s) and your classmates early is an opportunity for you to get feedback on your scenario choice and solution design and allows ample time for implementation. Your scenario and solution **can still be changed or refined after the discussion**.

A template of the proposal is included with this requirement document for your reference. You are recommended to prepare your proposal following the "**Proposal Template**" file for the discussion.

During the 20-minute discussion session in class, your team is encouraged to

- Present (informally) the proposal documents to your instructor(s);
- Explain your chosen scenario, simplification and/or assumptions, microservice interaction diagrams, design choices, possible uses of different technologies, and potential challenges in implementing the scenario and design;
- Present a conversational log with a generative AI of your choice (e.g. ChatGPT, Bing Chat, etc.), if any, to identify the services for your chosen scenario. Share your views on the identified services and how your team adopts and enhances it to arrive at your presented design above;
- Answer any question about your scenario and solution design and/or take notes of it for improvement in your later design refinement and implementation stages.

All the team members should be present for the discussion session. You are not required to submit the proposal. **The proposal is not graded, but very poor preparation of the proposal documents will get a penalty towards your final project marks**.

Presentation & Demonstration (25 marks)

- **Presentation slides**
  - You must submit your Presentation slides (in PPT or PDF format) **before the submission deadline**.
  - You must clearly show your business processes and the design of your microservices.
    - You may draw/use Microservice Interaction diagrams, SOA Layer diagrams (as shown in slides), or your own diagram styles if your team deems so.
  - If your team has implemented something **beyond the labs that are beneficial for your scenario**, you should have at least 1 slide to explain and justify that.
- **In-class presentation and demonstration (15 minutes)**
  - You must use the submitted presentation slides to present your scenario and solution;
  - You must reserve ample time within the time limit for the demonstration of your solution;
  - You will not be allowed to present beyond the time limit regardless of whether you have finished your demonstration, which may lead to a failure mark for your demo portion.
- **A Q&A session** will follow each presentation and demonstration. Every member of your team should be prepared to answer questions from all persons watching your presentation and demonstration, including your classmates and your instructor(s). Some immediate feedback will be provided by your instructor(s) at the end of the Q&A session.

Report (10 Marks)

- You must submit a report (in Word or PDF format, minimum font size 10) **before the submission deadline**.
  - You **must write the report following this** "Report Template".
  - Your report should include an appendix containing
    - the **Technical Overview diagram(s) or SOA Layer diagram(s)**
    - the **API docs** for all of your (micro)services
    - a table listing the **technical contribution of every team member**
  - The page limit is **max 6 pages**, **excluding** the cover page(s) and appendix (no page limit for the items in the appendix). Any page in the report that is beyond the page limit will not be marked.
  - Penalty applies if the required items in the appendix are missing, even though they are not explicitly graded. Whenever needed, they will be used to verify against your solution.
  - Your report must be presented in a coherent way and the flow must be consistent. Note that your report and presentation will be graded separately. Your report should be self-contained, including the beyond-the-labs, and clearly understandable by a person who may have not attended your presentation, as the instructors grading your report and presentation can be different.

Executables

- You must submit all of your code (e.g., .py, .html, .js files), batch scripts (e.g., .bat, .sh), Dockerfile(s), configurations (e.g., particular database accounts and passwords, docker settings for port forwarding/environment variables, .yml), and data files (e.g., pictures, .json, .csv, exported database .sql files), separated from the slides and report.
  - If you submit all the code and data into eLearn, please zip them together in one file.
  - Do NOT include files that can be generated from your code/scripts/configurations/data (e.g., the Docker images). If your zip file is too big (e.g., exceeds 100MB), consult your instructor(s) before submission.
  - You may also submit the code and data via a link to your own repository (e.g., hosted by github, sharepoint, etc.), instead of uploading all the files to eLearn.
- You must also write a readme file, **name it README.txt or README.md**, giving instructions on how to set up and run your solution using your submitted file (e.g., say so if your solution needs to run WampServer, or use a special account, or install additional python modules or other software, etc.)
- Penalty applies if there is no submission of code or README, even though they are not explicitly graded. Whenever needed, they will be used to verify against your solution as demonstrated in your presentation and described in the report.

Video

- You must submit a video recording (maximum 3 minutes) of the application demo.
- Put the URL of your **YouTube video** in a file **named video.txt** and include it in the submission on eLearn. It is your duty to ensure that the appropriate access has been given.

- Penalty applies if the video is missing, even though it is not explicitly graded. Whenever needed, they will be used to verify against your solution.

Submissions

- All submissions must be done electronically through the right eLearn Assignments drop-boxes; any other medium for submission (e.g., email, printout) will not be accepted.
- All submission deadlines must be strictly adhered to. You are strongly encouraged to submit early, taking potential network congestion into account.
- **Heavy penalties for late submissions** are as follows:

within 1 hour (one second late is late)	10% deductions off the marks you deserved
each subsequent hour	Penalty will double (i.e. 20%, 40%, 80%, and finally 100%)

Examples that are beyond the labs

Note: technologies beyond the labs are the things that fall under the broader IS213 learning objectives but that you need to research and implement on your own efforts; for fairness, the teaching team will not provide technical support for such things, except for high-level feasibility advice and brief resource pointers.

- Implement/invoke other types of APIs beyond REST, e.g., GraphQL, gRPC.
    - Note that your project should still have REST APIs to demonstrate your team's understanding and ability to apply what you have learnt; i.e. not a total replacement.
- Enable communication among microservices running on different physical machines.
    - If you are using WLAN-SMU, peer to peer network communications may not work. You may need to install SMU VPN and/or open up or disable the firewall on your laptop (e.g. Windows Defender) to enable cross-machine communication within the SMU network.
    - If you are using your own hotspot for networking, there should be no restrictions.
- Use multiple programming languages and/or frameworks (e.g., Spring Boot for Java and Kotlin, OutSystems for low-code development) to implement the microservices and their communication.
- Use a different message broker instead of RabbitMQ in a reasonable way suitable for the scenario.
- Use any API gateway, e.g., KONG or others, in the solution in a reasonable way for the scenario.

Examples that are <u>NOT</u> counted as beyond the labs

- Deploy and run the solution on cloud platform(s).
    - e.g., Deploy the API gateway in a cloud platform.
- Technologies covered in other courses (e.g., ESM) are NOT counted as beyond the lab.
- Implement complex functionality (e.g., a speech recognition engine) by yourself but the functionality can be available from invoking external APIs (e.g., Microsoft Azure Speech service).
- Use a different DBMS (including cloud storages) to implement the databases that can be implemented in MySQL.
- Use a different IDE other than VS Code.
- Use some version control systems (e.g., git) (this is a basic skill you should learn).

Penalty Situations

- Fail to show your scenario or draft solution design during the discussion session.
- Late or no submission(s) on eLearn.
- Presentation of a version of your slides different from the latest version you have submitted on eLearn before your presentation.
- Change your slides or code after submitting on eLearn. **No one is allowed to be "busy" working on his/her own work while others are presenting** as you are meant to be writing your own evaluations of other teams' presentations.
- Run over time during the presentation due to your own problems (e.g., lack of preparation or rehearsal, lack of testing of your demo code, etc.).
- Fail to demonstrate your solution during the presentation, which will lead to a failure mark for your demonstration portion right away.
- The report exceeds the page limits.
- Poor inter-team peer evaluation (such as consistently low points and feedback from other teams evaluating you).
- Poor intra-team peer evaluation (if any).
- Any other inappropriate situation at the discretion of your instructor(s).

Some considerations when using external services

- DO NOT misuse the services kindly provided by the service providers.
- For fairness to other teams, you should **NOT** rely on **PAID** external services to implement your user scenarios (using free development sandbox versions is fine).
- As a rule of thumb, use your official SMU email to create only **ONE** test/demo account, and avoid unnecessary requests to those servers.
- Test only a few times with the actual external server and setup (as per what you have registered), and after that, reuse and simulate the replies or outputs from the servers during your further development and testing so as to reduce the hits on those live servers. Only switch back to use the actual servers when you are close to the demonstration.

  E.g., You have registered for an email service via a web service. After testing it via code to send emails via this web service once or twice and knowing its behaviour, you replace the actual code with some simulation code to mimic the sending and receiving of emails to get things still working (although not like production). Only when it is nearer to the presentation, you change the code back to the actual code and test it a few more times with the live servers before the demo.
- It can happen that the external services become unavailable during the demo. To cater for such unfortunate situations, you may prepare a video recording of your demonstration with those live services before the presentation, **but penalties will apply** at the discretion of instructor(s).
- You may turn off all of your (micro)services, including the external ones you use, after your presentation.

Sample External Service Providers

National Digital Identity	https://api.singpass.gov.sg/developers Intro Slides: Empowering Business with SingPass Intro Video: NDI programme and its business applications by Shou Xin Poon (~22 minutes) and NDI APIs and usage examples by Roland Tan (~6 minutes)
Google	https://developers.google.com/
Amazon	http://aws.amazon.com/
Xignite	https://www.xignite.com/financial-data-apis
Facebook	https://developers.facebook.com/docs/
PayPal	https://developer.paypal.com/docs/checkout/
Stripe	https://stripe.com/docs
YouTube	https://developers.google.com/youtube/
Twitter	https://developer.twitter.com/en/docs

Marking scheme

Here are some (but not restricted to) of the criteria that may determine your grade for the project:

(1) In Class-Presentation with Demo (25 marks)

- Is the business scenario clearly explained and justified?
- Do the user scenarios go beyond single/simple UI-to-microservice interactions? For example, can they handle business exception situations, such as what happens if no driver responds to a taxi booking request within a time limit, or a customer cancels a taxi booking that a driver has accepted?
- Is there a smooth flow of the ideas throughout the presentation?
- Are the slides well designed and clearly readable?
- Is the presentation well-paced?
- Are there sufficient details provided in the presentation to help the audience understand all the diagrams?
- Does the demo clearly demonstrate the scenarios discussed?
- Does the demo work without any hiccups?
- Are the user interactions and inputs/outputs clearly visible?
- Was the team able to answer questions asked by the audience?
- Did the presentation show the team's understanding of various concepts relevant for the course?

Technical Depth

- Have the minimum technical requirements as per this document been fulfilled?
- Are there any technical aspects that went beyond the labs?
- How relevant are the technical implementations (including those beyond the labs) for the scenario? How justified are your reasons in adopting the technical implementations and architectural/design choices in relation to their scenario?
- How complex are the technical implementations (including those beyond the labs)?

Rubrics

Marks	Details
<=12	Presentation with Demo sloppy. Bare minimum technical depth.
>12-16	Presentation with Demo is of reasonable standard. Reasonable technical depth
>16-20	Presentation with Demo is good. Good technical depth.
>20-22	Presentation with Demo is very good. Very good technical depth.
>22	Presentation with Demo is exceptional. Exceptional technical depth.

(2) Report (10 marks)

- Is the report well-structured (including professionalism)?
- Does the report explain the business scenario clearly?
- Does the report clearly describe and justify the design and technical implementation?
- Are there discrepancies between the different sections of the report?
- Are the diagrams clear and consistent with the scenario and technical implementation?
- Are the technical aspects (including those beyond the labs) clearly explained in the report and justified well for the scenario? Why are the technical aspects (including BTLs) and design good for the respective scenarios?
- Are the terminologies used in the report correct and consistent?

Rubrics

Marks	Details
<=5	Report is sloppy.
>5-7	Report is of reasonable standard.
>7-8	Report is good.
>8-9	Report is very good.
>9	Report is exceptional.

(3) Penalty Situations at the discretion of the instructor(s).