THE UNIVERSITY OF SYDNEY

School of Computer Science

Enterprise-Scale Software Development (COMP5348)

Semester 2, 2025

Group Project

Group 15%
Due: 02 Nov 2025 23:59 on Canvas.

Requirements

In this group project assignment, your task as a group is to effectively and reliably build an online store application. We will refer the online store application as *Store* throughout this group project description, but you can name your own store. The store operates in a pure-online manner, and the stock are distributed in various warehouses. Your store should sell physical items instead of virtual items. Your store should also sell legal products. If you are unsure, please consult your tutor.

Enterprise applications usually need to integrate with a number of third-party systems to fulfill business requirements. You need to integrate your *Store* with 3 other external enterprise applications (*Bank*, *DeliveryCo* and *EmailService*), along with several in-house applications/components. Integration between these applications is required in order to support the following workflow through which *Store* customers can make orders.

The *Store* application should:

- Authenticate the user through username and password
    - The hashed version of the password should be stored in the database.
    - You should use this account during the demonstration: *username:* ***customer*** and *password*: ***COMP5348.***
- Allow users to place the order

When an order is placed, the *Store* application should:

- Find a warehouse that can fulfill the complete order
    - In case one warehouse cannot fulfill the complete order, then *Store* application should find the proper warehouses to fulfill the complete order.
- Request a transfer of payments/funds from the customer's account to the *Store*'s account using the services provided by the *Bank*
    - If the Bank transfer is successful, the *Store* sends a delivery request to *DeliveryCo*, advising that the items are ready for pick-up from the nominated warehouse(s).
- *DeliveryCo* notifies the *Store* when the delivery request has been received, when the goods have been picked up from the warehouse(s), when they're on the way to the customer, and when they've been delivered.
- The customer is notified via email (using the *EmailService*), when *DeliveryCo* has picked up the goods (and they're now in their depot), when they're on the delivery truck, and when *DeliveryCo* claims that the delivery is complete.

- The stock levels in the chosen warehouse should also be updated accordingly once an order is processed successfully and completely (stock checked, bank transfer OK, delivery pickup arranged).
- If the order processing fails for any reason, the customer is notified via email (using the *EmailService*) that there is a problem with their order and that the purchase order has been cancelled.
- The customer can also cancel their order before *Store* sends a delivery request to *DeliveryCo*. When an order is cancelled, the *Store* application should:
    - Update the warehouse(s) stock level accordingly.
    - Refund the payment transferred to the customer account. The customer should be notified by email of the status of refund (using the *EmailService*).

Your task as a group is to implement this application in a way to make it robust and realistic, and also to include whatever additional functionality and schema that are needed to meet all of the requirements listed above. In particular, you need to design and implement the inter-application communication so that the application provides high levels of availability and robustness.

For purposes of this group assignment, we have made the following assumptions:

- There is no need to implement the functionality of the Shopping Cart. We assume that the user can only purchase one item in various quantities at a time.
- The *DeliveryCo* simply needs to update the status of the package after a random time interval (e.g., around 5 seconds, but you can adjust this based on your demonstration plan). This means that around 5 seconds after the delivery request has been received, the goods will be picked up from the warehouse(s). The *DeliveryCo* also might lose some of the packages at each stage of the delivery process. The package loss rate can be set around 5% and adjusted based on your demonstration plan.
- The *EmailProvider* simply needs to print out that a message was sent to a certain address for this assignment (which is the current implementation). It does not need to send an actual email.
- As for the warehouse component in the system, each warehouse will have its own stock level records, and not all warehouses will hold stocks for all items. An individual order can either be dispatched from a single warehouse (if the complete order is available a single warehouse), or from multiple of warehouses if the complete order is not available in one warehouse. *DeliveryCo* should handle the process of picking multiple items from different warehouses, and you do not need to support inter-warehouse transfers in the *Store* application. You will have to implement the functionality that allows the order processing code to find out which Warehouse(s) can fulfill the order (has all the requested items in stock). For the purposes of this assignment, all warehouses that can fulfill the order are equally good – that is, you don't have to find the least number of warehouses for an order.

As a group, you will need to:

- Implement the system in a way that system's availability and fault-tolerance.
    - Design and implement at least 2 failure scenarios to show how your design facilitates availability and fault-tolerance.
- Consider proper transactional support, asynchronous messaging, message-oriented middleware and various types of Web Services in your design and implementation

- Log most critical information (including time stamps) that occurs during component/application communication and related events.

In your design and implementation, you will also need to take the following questions into account:

1. How are the availability and reliability quality attributes improved in your solution? You need to show what will happen under various partial system failures (failure of one or more application components).
2. Apart from availability and reliability, how well are the other quality attributes mentioned in the first lecture incorporated by your design? Consider and discuss trade-offs between different related quality attributes.
3. How does the use of an ORM database interface model affect the implementation of the tiered architecture within your Store application? Does your design maintain a clear separation between layers, or are there any areas where the architecture becomes blurred? Plotting the ERD Diagram should help.

As a team you will need to prepare a report that concisely illustrates and justifies how your design and implementation address all the above requirements and questions.

Group Formation

This assignment is group-based work. Each group must contain 4 students. Students must register their groups on Canvas by signing up to one of the groups listed under the page People > Group Projects.

Your group should use a source code management tool. The University of Sydney Enterprise GitHub should be used (github.sydney.edu.au). For each group, please create a new **private** GitHub repository under the Enterprise GitHub (https://github.sydney.edu.au/COMP4348-5348-2025), with the repository name being 'Tutorial-00-Group-00'. All code collaboration should happen in this repository. Some GitHub resources are available here https://docs.github.com/en/get-started/start-your-journey/hello-world. This will help you manage revision control and collaborative development. It also makes it easier to track and incorporate any bug fixes that might be made to the proof-of-concept code base. All team members must contribute to the development of above requirements and also be aware of the development done by other team members. Group member contributions to the project source code will be assessed based on the code repository used.

Group Contribution and Demonstration

It is crucial that every member of the group understand the design decisions, and the detailed working of the whole system. This is the natural outcome of effective teamwork, collaboration and integration of the whole system design and implementation, and this should take place in your group project work. It is not acceptable to divide the work so that each team member just writes code in one or two components, or (even worse) one person writes all the report and none of the code. All team members should contribute to the project work.

Each team will be required to demonstrate their work and answer questions about their design and implementation. More details will be shared about demo sessions/schedule and instructions on how to prepare your demo. All team members must attend the allocated demo session.

If members of your group do not contribute sufficiently you should alert your tutor as soon as possible. If a team member continues not to contribute (partially contribute) after the case is discussed among the team members and the tutor/TA, then the contribution percentage should be specified and signed by all members in the Academic dishonesty sheet. The course coordinator and the tutors have the discretion investigate the contributions based on the provided evidence and to scale the group's mark for each member as follows:

Level of Contribution	**Proportion of final grade received**
No contribution	0%
poor/partial contribution	1% - 49%
Partial but not enough contribution	50%-99%
Major contribution	100%

Project Deliverables/Submission

Each group must make one submission of each of the following through the provided links on Canvas:

1. A zipped file containing your source code for your application and a ReadMe file that concisely explain how to run your application along with any source code required to create entities and populate data.
2. A report that concisely discusses the architectural decisions that you made when designing your system and implementation for all the requirements, including your answers to the questions, with a focus on message exchanges between your components and showing how you will handle the unavailability of various systems, and how data integrity is maintained across partial system failures. You should justify any trade-offs that you. Submit through the other assignment page (Group Project - Report).
3. A copy of the Academic Honesty cover sheet (group work version) signed by all members of the group (to be included in your zip) indicating the group work in the source code and the report. Projects will not be assessed until a copy of the academic honesty is provided and signed all members.
4. AI Tools Usage Form.

   Download the "AI Tools Usage Form" and complete it, making sure that you follow the instructions in this form. You will need to name every automated writing tool and generative AI tools you have used and explain how you have used them (which tool, input, output, provide explanation if necessary). Once you have completed this task, save this form in plain text or docx format.

Important note: Make sure your team submit the right assignment work (download and check the submitted files). Any wrong submitted files will be treated as the actual submission and any re-submission will receive late penalty as per the unit outline polices.

Note: any assignment updates will be made in Canvas and announced via Ed.