Persistence and Analytics Fundamentals Assessment

Date: Friday January 06 2023
Assessment Time: 0900 - 1700 (including meal breaks)

Overview

In this assessment you will be writing a Spring Boot application to create an order for an online e-commerce application.

There are **7 tasks** in this assessment. Complete all tasks.

Passing mark is **65% (75 marks**). Total marks is **116**.

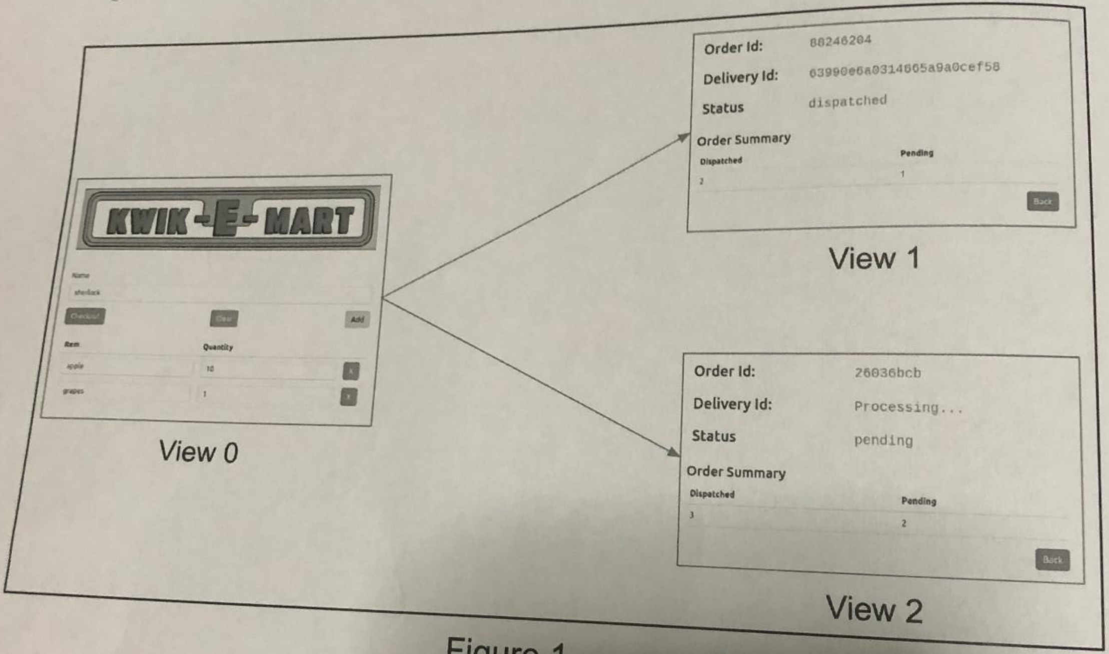
Read this entire document before attempting the assessment. There are 12 pages in this document.

Application Overview

This is your online e-commerce business. The landing page of your e-commerce store where your customer will place their orders (View 0).

When the checkout button is pressed the order will be sent to the backend Spring Boot application to be processed and dispatched to the warehouse for delivery. If the dispatch is successful, a delivery id will be returned and the customer will be shown View 1, otherwise View 2.

Figure 1 shows the flow



Name
sherlock

Item	Quantity
apple	10
grapes	1

Order Id: 88246204
Delivery Id: 63990e6a0314665a9a0cef58
Status dispatched
Order Summary

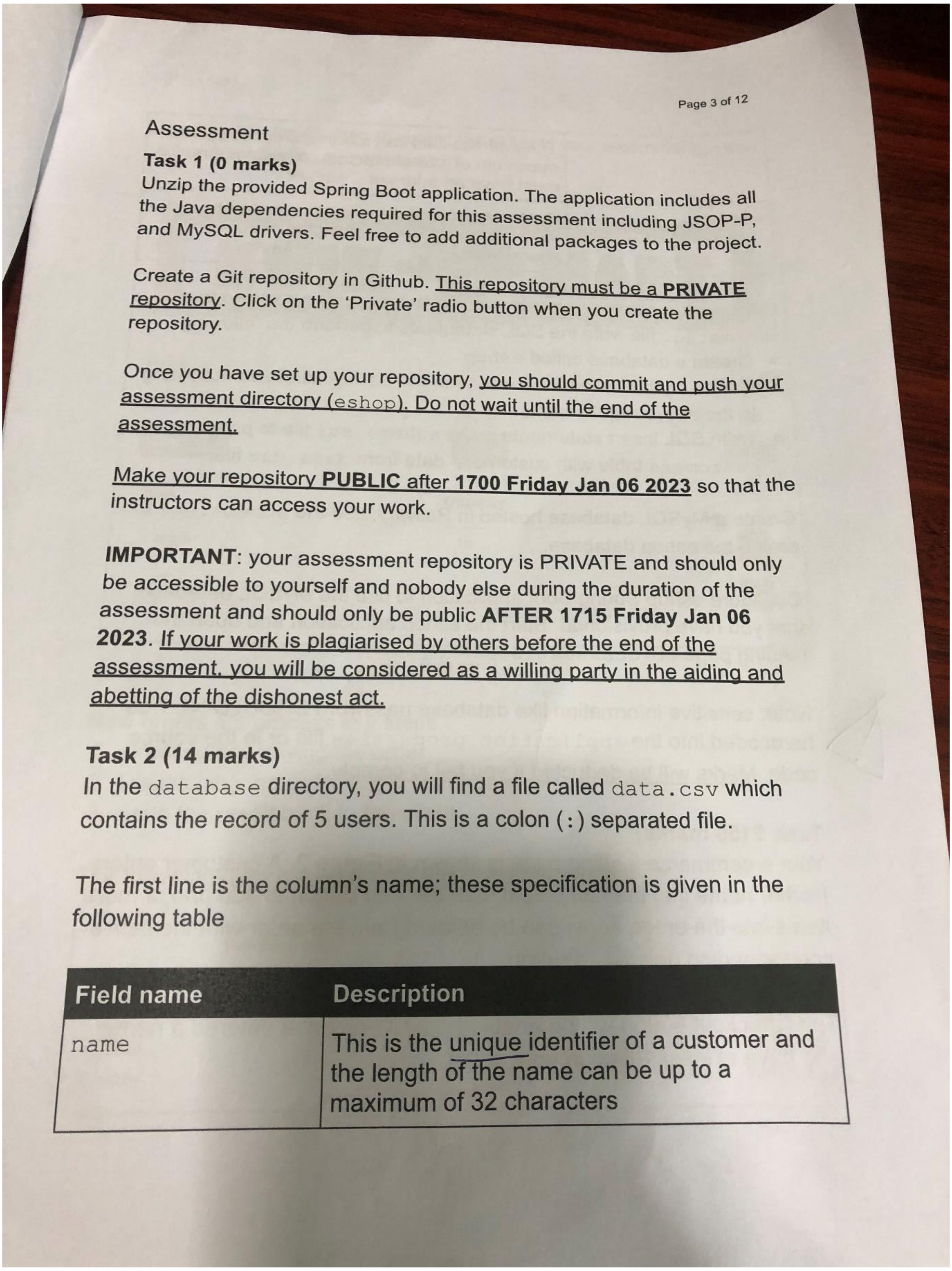
Dispatched	Pending
2	1

Order Id: 26036bcb
Delivery Id: Processing...
Status pending
Order Summary

Dispatched	Pending
3	2

Figure 1

Detailed explanation of individual views will be provided in the following tasks.



Assessment

Task 1 (0 marks)

Unzip the provided Spring Boot application. The application includes all the Java dependencies required for this assessment including JSOP-P, and MySQL drivers. Feel free to add additional packages to the project.

Create a Git repository in Github. This repository must be a **PRIVATE** repository. Click on the 'Private' radio button when you create the repository.

Once you have set up your repository, you should commit and push your assessment directory (`eshop`). Do not wait until the end of the assessment.

Make your repository **PUBLIC** after **1700 Friday Jan 06 2023** so that the instructors can access your work.

IMPORTANT: your assessment repository is PRIVATE and should only be accessible to yourself and nobody else during the duration of the assessment and should only be public **AFTER 1715 Friday Jan 06 2023**. If your work is plagiarised by others before the end of the assessment, you will be considered as a willing party in the aiding and abetting of the dishonest act.

Task 2 (14 marks)

In the `database` directory, you will find a file called `data.csv` which contains the record of 5 users. This is a colon (`:`) separated file.

The first line is the column's name; these specification is given in the following table

Field name	Description
`name`	This is the unique identifier of a customer and the length of the name can be up to a maximum of 32 characters

`address`	The address field can store addresses up to a maximum of 128 characters. All customers must have an address NOT NULL
`email`	Customers' email. Emails are mandatory and the maximum allowable length is 128 characters NOT NULL

Create a file called `schema.sql` in the `database` directory. In the `schema.sql` file write the SQL statements to perform the following

- Create a database called `eshop`
- Create a table called `customers` in the `eshop` database according to the above requirements.
- Write SQL insert statements in the `schema.sql` file to populate the `customers` table with customers' data from `data.csv` file.

Create a MySQL database hosted in Railway; use the `schema.sql` to create the eshop database.

Configure eshop Spring Boot application to use the MySQL database that you have provisioned. Start the eshop application and open the landing page at `localhost:8080`.

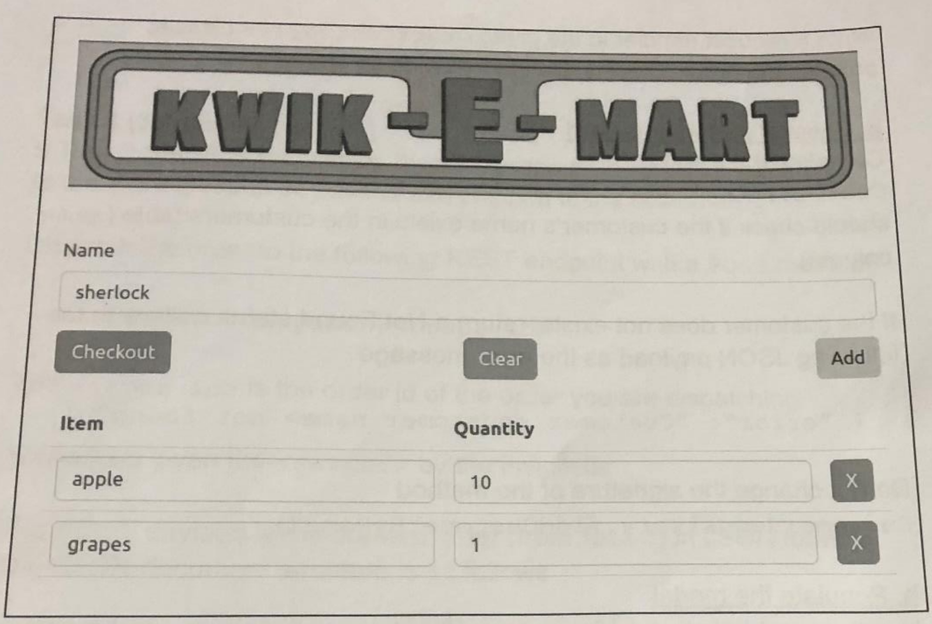
⚠ Note: sensitive information like database password SHOULD NOT be hardcoded into the `application.properties` file or in the source code. Marks will be deducted if you fail to comply.

Task 3 (50 marks)

Your e-commerce landing page is shown in Figure 2. A customer enters his/her name into the name field; use the Add button to add one or more items into the order. Items can be deleted from the order with the item's corresponding delete (X) button.

The Checkout button will be enabled only if you have entered a name and have at least 1 item in your order.

Click the Checkout button to submit the order to your e-commerce backend application for processing.



Name

sherlock

Item	Quantity
apple	10
grapes	1

Figure 2

New orders are process in 3 steps

1. Creating and saving the order
2. Dispatching the order to the warehouse
3. Send the result back to the client

The following diagram illustrates the checkout process.

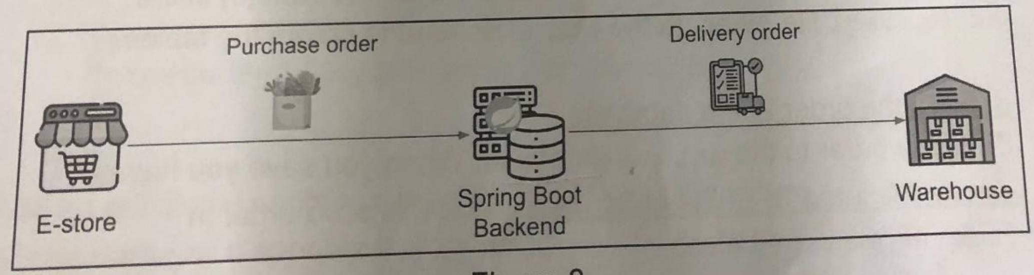


Figure 3

e-store $\xrightarrow{\text{order}}$ app backend

Step 1 will be performed in this task, Step 2 and Step 3 will be described in Task 4 and Task 5 respectively.

Write a request handler in the given `OrderController` class to process the order (Step 1). Do the following for Step 1

a. Check if customer is valid

Complete the method `CustomerRepository.findCustomerByName()`. This method should check if the customer's name exists in the customers table (`name` column).

If the customer does not exists, return a Not Found status code with the following JSON payload as the error message

```
{ "error": "Customer <customer_name> not found" }
```

Do not change the signature of the method `CustomerRepository.findCustomerByName()`.

b. Populate the model

Use the provided models (in `models` directory) to hold the order details sent from the frontend e-store. → name

Item
quantity

Generate a unique 8 character long order id for the order and set the order date. Hint: use the UUID class to generate the order id.

c. Create a Order table

Create one or more database tables to store the order in the `estore` database. Write your SQL statements to create the table(s) in the `schema.sql` file. Execute the SQL statement to create the tables.

d. Save the order to the database

Save the order to the `estore` database. When you save you have to ensure the integrity of the order data. Write this save order in `OrderRepository` class.

If the save is fails, return a Internal Server Error status with the following JSON payload as the error message

```
{ "error”: <the error message> }
```

Task 4 (30 marks) → creating & saving order

If Task 3 Step 1 is successful, then the order will need to be dispatched to the warehouse to be packed and shipped to the customer.

Dispatch the order to the following REST endpoint with a `POST` method

```
<server>/dispatch/<order id>
```

The `<order id>` is the order id of the order you are dispatching.

You will be given the `<server>` by the invigilator.

The POST payload is the created order (from Task 3) in JSON format. The JSON document structure is as follows

```
{
    "orderId”: <order id>,
    "name”: <name>,
    "address”: <address>,
    "email”: <email>,
    "lineItems”: [
        { "item”: <item>, "quantity”: <quantity> },
        { "item”: <item>, "quantity”: <quantity> },
        …
    ],
    "createdBy”: <your name as in NRIC>
}
```

Add an additional attribute called `createdBy` to the order; `createdBy` is your name as it appears in your NRIC.

Use the `WarehouseService.dispatch()` to write the order dispatch function. Do not change the `dispatch()` method's signature.

If the dispatch request is successful, the endpoint will return an OK status with the delivery id in the following JSON payload

```
{
    "orderId": <order id>,
    "deliveryId": <delivery id>
}
```

where `<delivery id>` is for the corresponding `<order id>`.

The dispatch can fail if the order details are incorrect or incomplete. The endpoint is also unstable and has a 33% chance of failing.

↓ throws Internal Server Error

Write the SQL statements to create a table called `order_status` *in the* `schema.sql` *file.The table contains the following mandatory columns*

Field name	Description
`order_id`	The order id of the dispatched order
`delivery_id`	The delivery id returned by the dispatch endpoint. The delivery id is not longer than 128 characters
`status`	Delivery status. This column can contain either `pending` or `dispatched` value
`status_update`	A timestamp when the orders status is inserted into the `order_status` table

Create the table in your `eshop` database.

The `order_status` table is for recording the dispatched status. An order status record is inserted into this table whenever an order is dispatched to the warehouse.

If the dispatch is successful, write the order id, the delivery id and the value `dispatched` in the `status` column and the timestamp.

If the dispatch is not successful, write the order id and the value `pending` in the `status` column and the timestamp.

Task 5 (5 marks)
Send the status back to the e-store frontend. If the dispatch is successful, send a OK status with the following JSON payload

```
{
     "orderId": <order id>,
     "deliveryId": <delivery id>,
     "status": "dispatched"
}
```

If the dispatched failed, send a OK status with the following payload

```
{
     "orderId": <order id>,
     "status": "Pending"
}
```

The following flowchart summarises the checkout stages (Task 3 to Task 5) in the Figure 4

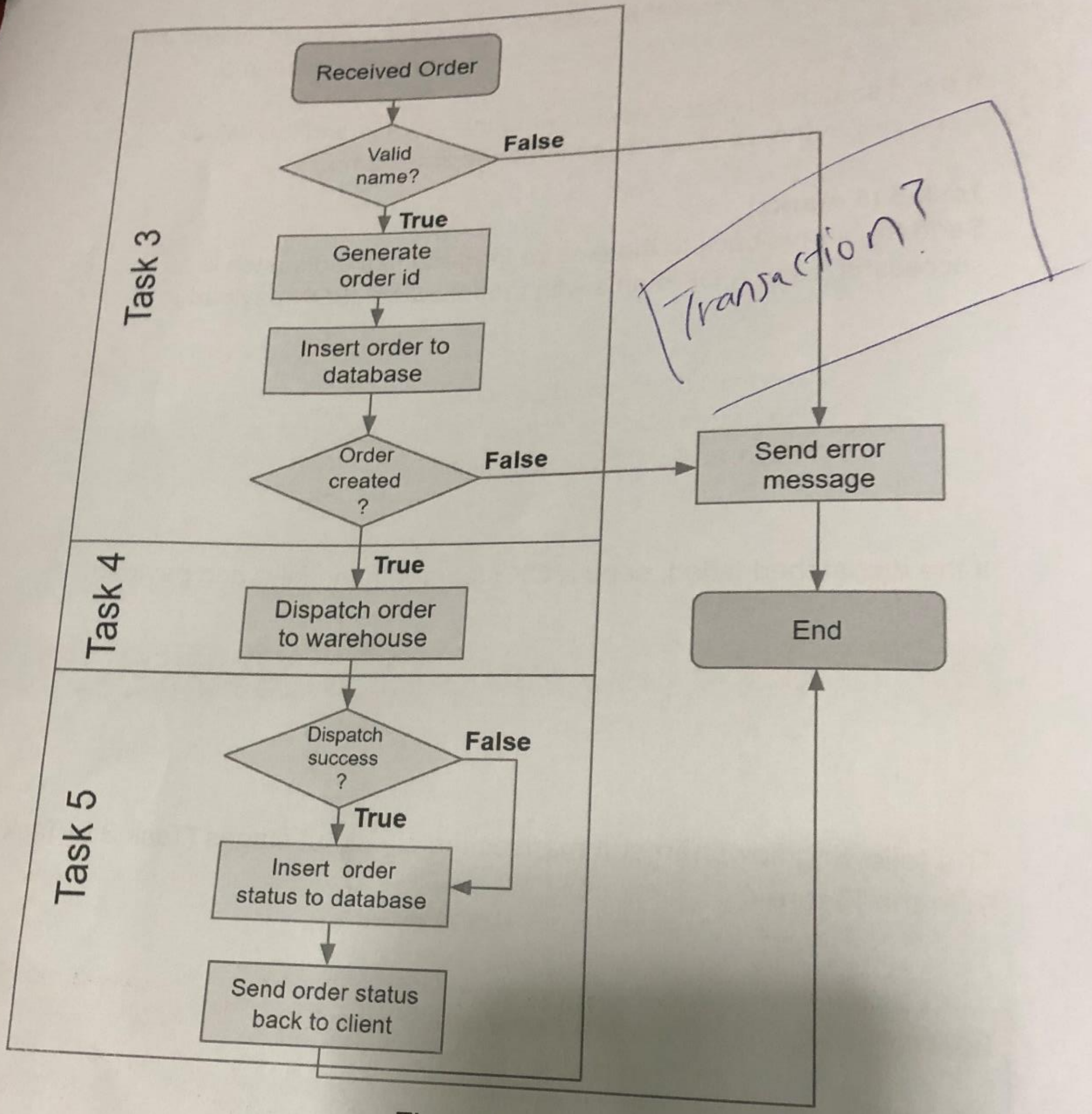


Figure 4

Task 6 (12 marks)

The e-store front makes the following HTTP request to get the total number of dispatched and pending orders for a customer.

```
GET /api/order/<name>/status
Accept: application/json
```

Write a request handler to process the request and return the required data in the following JSON document structure

```
{
      “name”: <name>,
      “dispatched”: <number of dispatched orders>,
      “pending”: <number of pending orders>
}
```

Use a **join statement** to get this result from your `estore` database.

Task 7 (5 marks)
Deploy your application to Railway. The deployment should be based on any commits on or before **1700 Friday January 06 2023**.

Do not undeploy your application before **2359 Friday January 16 2023.**

Submission

You must submit your assessment by pushing it to your repository to either GitHub, GitLab or BitBucket.

Only commits on or before **1700 Friday January 06 2023** will be accepted. Any commits **after Friday January 06 2023** will not be accepted. No other form of submission will be accepted (eg. ZIP file).

Remember to **make your repository public after Friday January 06 2023** so the instructors can review your submission.

After committing your work, post the following information to Slack channel `#03-paf-submission`

1. Your name (as shown in your NRIC)
2. Your email
3. Batch - 2a or 2b
4. Git repository URL

5. Railway deployment URL. Please do not undeploy your application from Railway or tear down your MySQL database until after **2359 Friday January 16 2023**

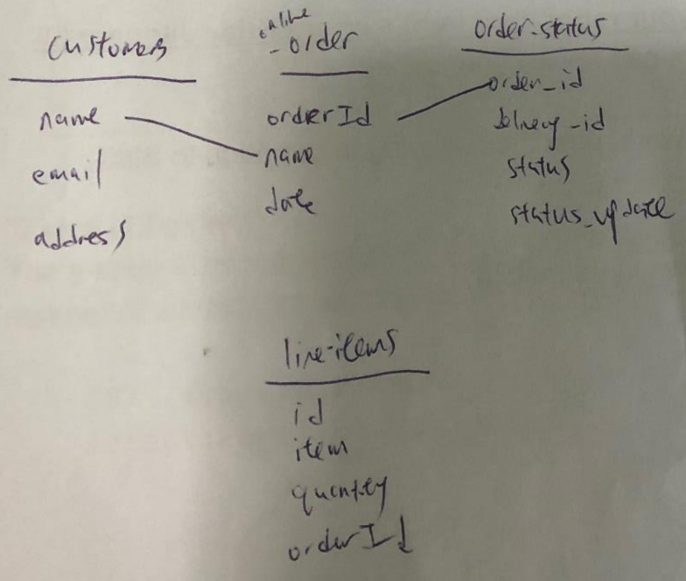
It is your responsibility to ensure that all the above submission requirements are met. Your assessment submission will not be accepted if

1. any of the 5 items mentioned above is missing, and/or
2. your information did not comply with the submission requirements eg. not providing your full name as per your NRIC, incorrect email, forgetting to post the Railway deployment URL, etc, and/or
3. the repository is not public after **1700 Friday January 06 2023**

Academic Integrity

This is an open book assessment. You may search the Internet for resources or use reference books during the assessment. The assessment must be your own work. You cannot ask a third party to write any part of this assessment. This will result in an automatic failure.

You are to ensure the integrity and working condition of your PC/notebooks (eg. wireless/internet connection, battery, screen, accidents like water spillage) during the assessment. NUS ISS will not accept any of these as a reason for deferring or retaking your assessment.


customers
name
email
address
online
_order
orderId
name
date
order_status
order_id
delivery-id
status
status_update
line-items
id
item
quantity
orderId