Persistence and Analytics Fundamentals Assessment

Date: Friday Jan 26 2024
Assessment Time: 0900 - 1700 (including meal breaks)

Overview

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

Passing mark is **65%** (**73 marks**). Total marks is **112**.

Read this entire document before attempting the assessment. There are 15 pages in this document.

IMPORTANT: Before You Start the Assessment

During the assessment, you may access any website except AI related ones like ChatGPT, Bard, etc. If you access any of these sites during your assessment, your assessment will be terminated immediately.

You must **uninstall all AI coding extensions from your IDE**. AI coding extensions are those that generate entire solutions like ChatGPT, Github Copilot, Tabnine, IntelliCode, etc. If you have installed any of the following AI extensions, please uninstall them from your IDE before you start the assessment.

Random checks will be performed during the assessment. If your IDE is found to have installed any of the above extensions or other AI coding tools, your assessment will be terminated. I forgot to uninstall is not an acceptable explanation.

Code completion extension is permissible; e.g. Angular Language Service, Angular Snippets, Java Language Support, Emmet, etc.

You cannot take this document out of the classroom during the assessment period. You cannot take pictures or scan this document during the assessment period. You cannot communicate with anyone

using any means when you are in the assessment classroom. If you are found to be doing this, your assessment will be terminated immediately.

Internet Access

This is an open book assessment.

You may go online to look up information. But you are **only limited to the following sites** listed below

- Java 21 - https://docs.oracle.com/en/java/javase/21/
- Spring Boot - https://docs.spring.io/spring-boot/docs/current/api/index.html
- Spring Framework - https://docs.spring.io/spring-framework/docs/current/javadoc-api/
- Thymeleaf - https://www.thymeleaf.org/documentation.html
- Jedis - https://javadoc.io/doc/redis.clients/jedis/latest/index.html
- MySQL - https://dev.mysql.com/doc
- Redis - https://redis.io/docs
- MongoDB - https://www.mongodb.com/docs/manual/
- Docker - https://docs.docker.com/reference/
- Railway - https://docs.railway.app
- StackOverflow - https://stackoverflow.com/
- Your GitHub repository - https://github.com/<your github user>
- Your Railway dashboard - https://railway.app/dashboard

You can access any subresources prefixed by the above URLs eg https://stackoverflow.com/questions/15182496/why-does-this-code-using-random-strings-print-hello-world is permissible.

You may use Google for searching but you can only open links listed above.

If you access any other sites that are not in the above list, your assessment will be terminated immediately. If you accidentally open a URL that is not in any of the above list, close the page IMMEDIATELY. If you linger and start to read its contents, your assessment will be terminated.

You may also reference any printed materials such as your 'cheatsheet', notes, slides, books, etc.

Assessment Repository Setup

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

Make your repository **PUBLIC after Friday 1700 Jan 26 2024** so that the instructors can access your work.

IMPORTANT: this repository is PRIVATE and is only accessible to yourself and nobody during the duration of the assessment until **AFTER after Friday 1700 Jan 26 2024**. If your work is plagiarised by others, you will be considered as a willing party in the aiding and abetting of the dishonest act.

Setup

You will be given a ZIP containing your assessment template. The unzipped file contains the following directories

- `bedandbreakfastapp` - a partially completed Spring Boot application developed with Java 21. All the required dependencies for this assessment have been added to `pom.xml`. You are free to add additional dependencies. Note: some classes and methods are marked with `DO NOT MODIFY`. If you modify these classes and/or methods, you will invalidate any assessment tasks that use them.
- `data` - contains data to be loaded into the database.

Create a new Railway project. In the Railway project provision the following services:

- MySQL database
- MongoDB database
- Service for the Spring Boot application

Assessment

In this assessment you will be completing an accommodation booking application. The application allows users to search and to book accommodations for their Australian holidays.

The application consist of 4 views; the transition between the views are show Figure 1

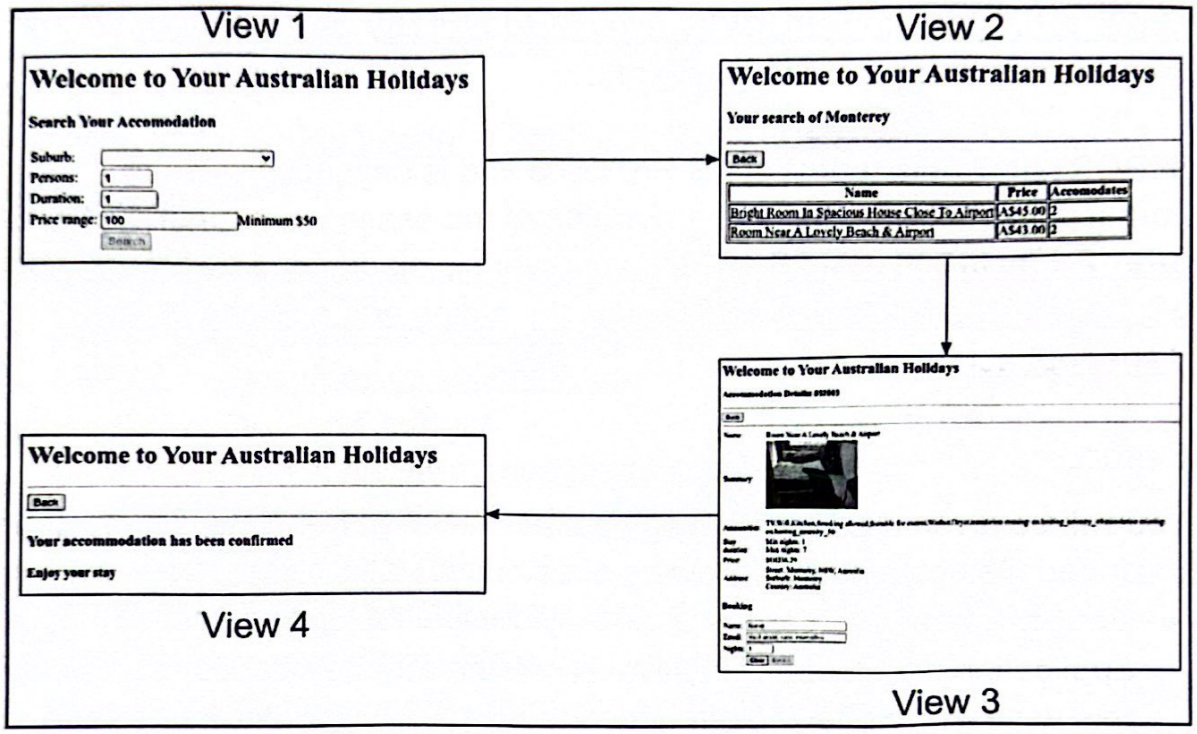

Figure 1 Application flow

- View 1 - this is the landing page. You can search Australian accommodations based on a set of criterias like location/suburb, persons, etc.
- View 2 - once you have specified your search criteria and press Search, you will be transitioned to View 2. In this view, the application will display the list of accommodations that matches your criteria if there are any.
- View 3 - to view the details of an accommodation, click on an accommodation name in View 2. The application will then transition to View 3 with the accommodation details. You can also book the accommodation with the booking form in the view.

- View 4 - the application will transition to View 4 if your booking is successful.

All views have a 'Back' button to return to View 1.

There will be more details on the views in the individual tasks.

Task 1 (12 Marks)
In this task, you will be creating a relational database to store the users, bookings and reviews in MySQL.

Create a database called `bedandbreakfast`. In the `bedandbreakfast` database create the following tables:

Table name: `users`

Column name	Type	Description
`email`	`varchar(128)`	Email. This is the record identifier
`name`	`varchar(128)`	User friendly name

Table name: `bookings`

Column name	Type	Description
`booking_id`	`char(8)`	Random booking identifier of 8 characters. This is the record identifier
`listing_id`	`varchar(20)`	The accommodation's id of this booking
`duration`	`int`	The number of days a user will be staying
`email`	`varchar(128)`	The email of the user making the booking. The email must exist in the `users` table.

Table name: `reviews`

Column name	Type	Description
`id`	`int`	A running number. This is the record identifier
`date`	`timestamp`	Review date
`listing_id`	`varchar(20)`	Accommodation id
`reviewer_name`	`varchar(64)`	Reviewer's name
`comments`	`text`	Reviews of the accommodation

Write the SQL to create the `bedandbreakfast` database and all its tables in a file called `task1.sql`. Save this file in the `data` directory.

The `users.csv` file in the `data` directory are records for `users` table. Write a batch SQL `insert` statement in `task1.sql` to insert all the records in `users.csv` into the `users` table.

Execute the `task1.sql` to create the `bedandbreakfast` database, its tables and records.

Task 2 (25 marks)

In the `data` directory, you will find a ZIP file called `data.zip`. Extract the file `listings.json` from the ZIP file. `listings.json` contains accommodations from around the world.

Task 2.1

Import the contents of `listings.json` into your Mongo database; the collection should be called `listings_and_reviews`. The Mongo database is called `bedandbreakfast`.

Write the command you use to import `listings.json` in a file called `task2.txt` in the `data` directory.

Task 2.2

Since the application only deals with accommodations from Australia, use either `mongosh` or Studio3T to filter all the Australian accommodations from `listings_and_reviews` collection into a new collection called `listings` in the `bedandbreakfast` Mongo database. The filter should be case insensitive viz. it should filter all documents regardless of what case Australia is written in.

Write the native MongoDB query you use to create the `listings` collection in the file `task2.txt` in the `data` directory. You must use a single Mongo query to accomplish this task.

You may drop `listings_and_reviews` collection.

Task 2.3

Write a single native MongoDB query to extract `reviews` from the Australian `listings` collections (from Task 2.2) and save it into its own collection called `reviews`.

The `reviews` document should only contain the following attributes:

- `_id`
- `date`
- `listing_id`
- `reviewer_name`
- `comments`

During the extraction (of `reviews`) perform the following cleansing operations on each `review` document:

- Delete all `\n` character from the `comments` attribute
- Delete all `\r` characters from the `comments` attribute
- Delete all comma (`,`) from the `reviewer_name` attribute

Hint: `$replaceAll`

Write the native MongoDB query you use to extract the `reviews` into its own collection in the file `task2.txt` in the `data` directory.

Task 2.4

Write a single native Mongo statement to delete the `reviews` attribute from the `listings` collection.

Write the native MongoDB query to delete the `reviews` attribute from `listings` in the file `task2.txt` in the `data` directory.

Task 2.5

Export `reviews` collection a CSV file. You should export only the following attributes from the `reviews` collection in the order listed below:

1. `date`
2. `listing_id`
3. `reviewer_name`
4. `comments`

Write the command you use to export `reviews` to a CSV file in the file `task2.txt` in `data` directory.

Task 2.6

The `csv2sql.jar` file is a Java program, written in Java 21, for converting the CSV to SQL inserts statement. The insert statements will insert into the reviews table that you have created in Task 1.

Execute `csv2sql.jar` to convert the exported CSV file into SQL with the following command:

```
java -jar csv2sql.jar reviews.csv reviews.sql
```

where `reviews.csv` file is your exported CSV from `reviews` collection. The `reviews.sql` is the file that will be created by the program.

Execute `reviews.sql` to insert the reviews record into the `reviews` table in MySQL. Do not delete the generated `reviews.sql` or this task will not be graded.

Note: You should not encounter any errors when executing `reviews.sql`. If you encounter any errors, then you have either incorrectly created the `reviews` collection or the CSV file.

Task 3 (12 marks)

Start the Spring Boot application in `bedandbreakfastapp` directory and use your browser to view the application on http://localhost:8080

View 1, shown in Figure 2 below, allows a user to search for Australian accommodations from the Mongo database that you have setup in Task 2.

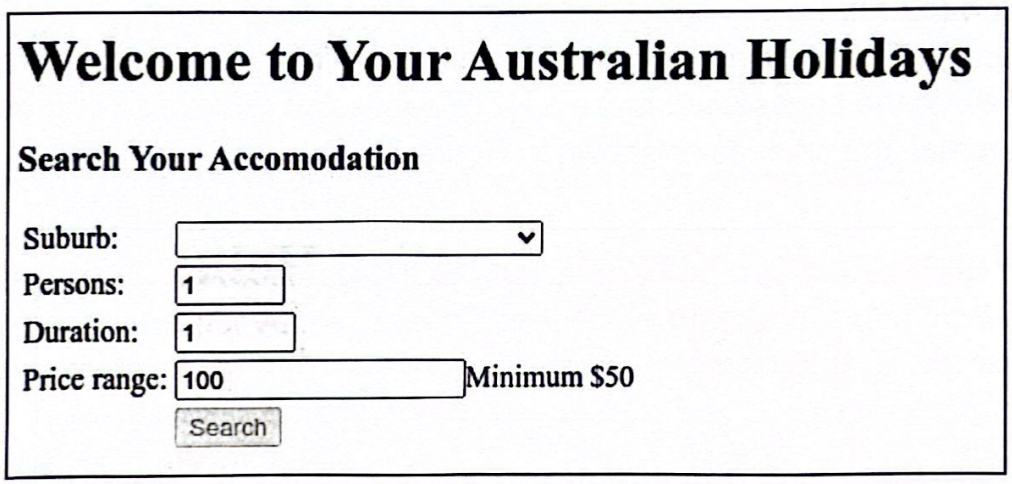

Figure 2 View 1

View 1 makes a request to the Spring Boot application to retrieve a list of suburbs from the attribute `address.suburb` in the `listings` collection from Mongo.

Examine the Spring Boot application and identify the handler of this request; complete View 1 by writing a single Mongo query to return a list of suburb names from the `listing` collection.

The Mongo query should return the result in the following document structure:

```
[
  { "_id" : "Abbotsford/Wareemba" },
  { "_id" : "Alexandria" },
  { "_id" : "Annandale/Leichhardt" },
  ...
]
```

Do not return any suburb names that are either an empty string or are null.

Write the native Mongo query that you used for the query inside a comment block above the Java method. Marks will be awarded for the Mongo query.

Task 4 (12 marks)

When the Search button is pressed after specifying the search criterias in View 1, the browser will make a HTTP request to retrieve all listings that match the criterias as shown in Figure 3.

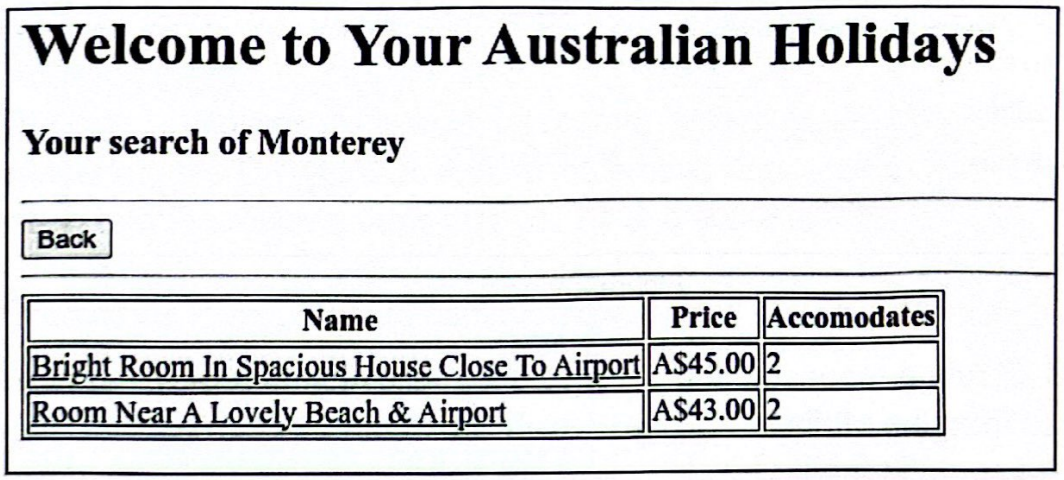

Figure 3 View 2

Examine the Spring Boot application and identify the handler of this request; complete View 2 by querying from `listings` collection all accommodations that match the search criteria.

Write a single Mongo query to search the listings collection according to the following requirements:

- Search for listings that match the provided suburb; the match should be case insensitive
- Search price, `price_range`, should be less or equal to the listing `price`
- Number of person should be greater than or equals to the the listing's `accommodates` attribute
- Duration (duration), the number of nights the user is staying should be less than or equals to the attribute `min_nights`
- The query should return only the following attributes: `_id`, `name`, `accommodates` and `price`
- The result should be sorted in descending order with the most expensive listing appearing first

Note: When you read the `price` attribute use

```
doc.get("price", Number.class).floatValue()
```

to return the price, where `doc` is `org.bson.Document` object.

Write the native Mongo query that you used for the query inside a comment block above the Java method. Marks will be awarded for the Mongo query.

Task 5 (10 marks)

When an accommodation is selected in View 2, a request will be made from the browser to Spring Boot to retrieve the details of the accommodation. The price of the accommodation, in Australian currency (AUD) should be converted to Singapore currency (SGD) before displaying in View 3. See Figure 4 below.

Examine the Spring Boot application and identify the handler of this request; complete the handler by writing the REST API call to convert the Australian currency to Singapore currency with the following REST API

```
https://api.frankfurter.app/latest
```

You can find the documentation here https://www.frankfurter.app/docs. If the REST endpoint returns any error, return -1000 as the converted price.

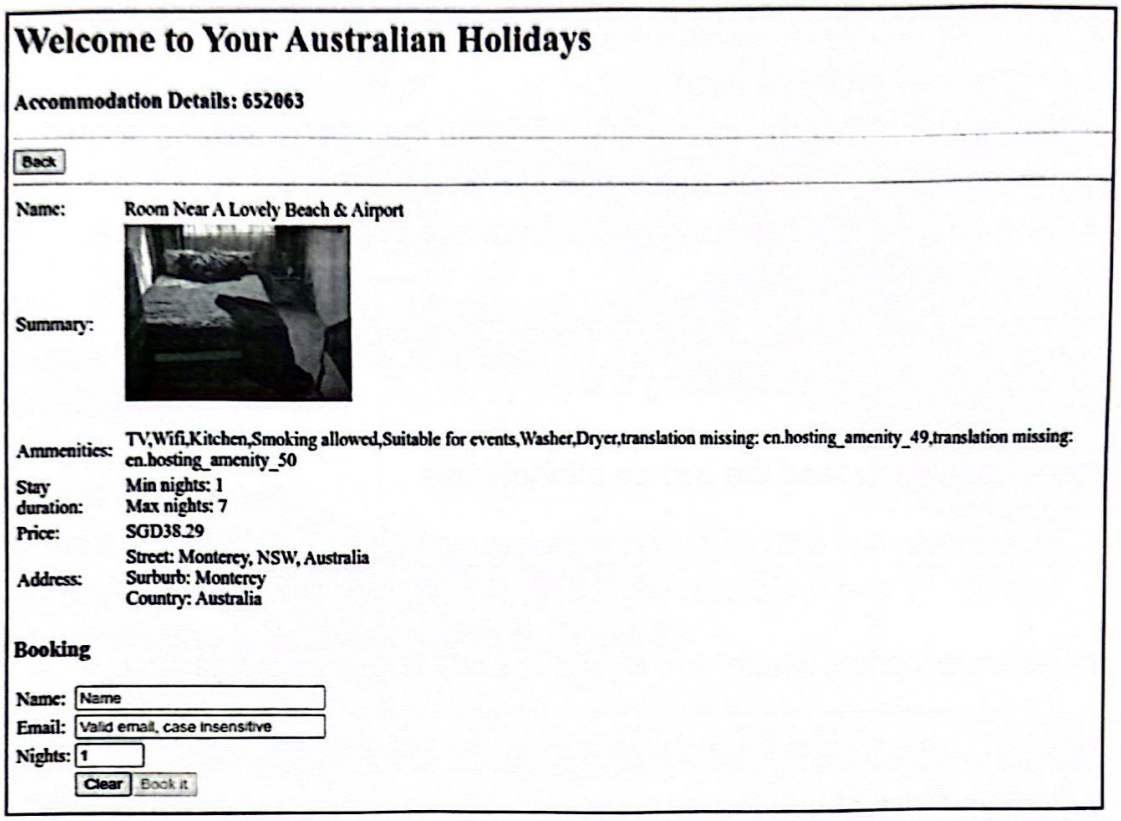

Figure 4 View 3

Task 6 (26 marks)

If a user wishes to book the accommodation in View 3, the user would enter his/her details into the Booking form. When the 'Book it' button is pressed, the browser will make the following HTTP request to Spring Boot

```
POST /api/accommodation
Content-Type: application/json
Accept: application/json
```

...

The request payload consist of the following details

- `id` - the accommodation's id that the user is booking
- `name` - the user's name

- `email` - the user's email
- `nights` - the number of nights

Write the request handler to add the booking details from the HTTP request into the `breadandbreakfast` MySQL database. The booking is processed in the following manner:

1. Check if the user exists in the `users` table. If the user does not exist, add the user into the `users` table. If the user exists, then proceed to the next step
2. Add the booking details into the `bookings` table.

Figure 5 below illustrates the booking process.

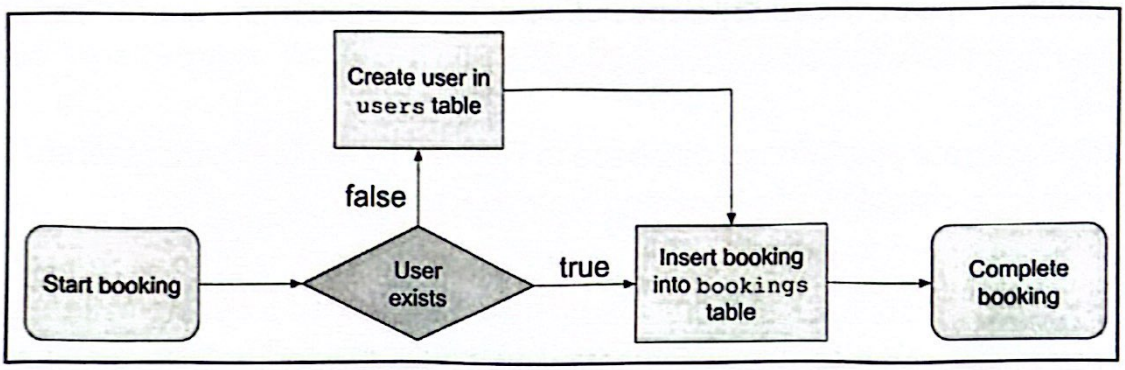

Figure 5 Booking process

If the booking is successful, return a 200 status code with an empty JSON document as payload "`{}`".

If the booking is unsuccessful, return a 500 status code in the following JSON document as payload

```
{ "message": "<error reason>" }
```

Implement the following methods:

- `BookingsRepository.newUser()` - to create a new user
- `BookingsRepository.newBookings()` - to create a new booking

Use the methods in `BookingsRepository` class to implement `ListingsService.createBooking()`.

Implement the HTTP request handler for the booking in the `BnBController` class. Use the `ListingsService.createBooking()` method to create the booking.

Task 7 (15 marks)
Deploy your application to Railway project that you have provisioned.

Write a Dockerfile to provision the Spring Boot application using Java 21. Write a multi stage Docker file so that the final image should only contain the application JAR file and not your application source.

The databases configurations eg. password, etc. should not be exposed either in `application.properties` or hard coded in the source code. Marks will be deducted if they are exposed.

Hint: migrate your Mongo database to Railway by exporting them from your local machine and importing them to Railway.

Submission

You must submit your assessment by pushing it to your repository at GitHub.

Only commits on or before **Friday 1700 Jan 26 2024** will be accepted. Any commits after **Friday 1700 Jan 26 2024** will not be accepted. No other form of submission will be accepted (eg. ZIP file).

Remember to **make your repository public after Friday 1700 Jan 26 2024** so the instructors can review your submission.

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

1. Your official name (as it appears in your NRIC)
2. Your email
3. Railway deployment URL, the URL to access your application

4. Git repository URL. Remember to make your repository **PUBLIC after Friday 1700 Jan 26 2024**

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 4 items mentioned above is missing from `#03-paf-submission` channel, and/or
2. Your information did not comply with the submission requirements eg. not providing your official name, and/or
3. Your repository is not publicly accessible after **Friday 1700 Jan 26 2024**

You should post the submission information to the Slack channel `#03-paf-submission` no later than **Friday 1715 Jan 26 2024**.

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 or use AI tools such as ChatGPT to generate output and submit it as part of your assessment. This will result in an automatic disqualification from the assessment.

The NUS ISS takes a strict view of cheating in any form, deceptive fabrication, plagiarism and violation of intellectual property and copyright laws. Any student who is found to have engaged in such misconduct will be subject to disciplinary action by NUS ISS.

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.