VTTP 2023 Batch 3 Persistence and Analytics Fundamentals Assessment

Date: Fri Jul 21 2023
Assessment Time: 0900 - 1700 (including meal breaks)

There are **6 tasks** in this assessment. Complete all tasks.

Passing mark is **65%** (**75 marks**). Total marks is **114**.

Read this entire document before attempting the assessment. There are 14 pages in this document.

Important

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 also **uninstall all AI coding extensions from your IDE**. AI coding extensions are those that generate entire solutions like ChatGPT or Github Copilot. If you have installed any of the following AI extensions (and also any others that are not listed below), please uninstall them from your IDE before you start the assessment

- IntelliCode
- Tabnine
- Github Copilot and any extensions build on Copilot
- Azure Machine Learning
- ChatGPT and any extensions build on ChatGPT

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.

Do note that code completion extension is permissible; eg

- Angular Language Service
- Angular Snippets
- Java Language Support

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.

Setup

1. Assessment Project Setup

You will be given an assessment project template (ZIP file) for this assessment. Unzip this project template; a directory called `vttp2023_batch3_paf_assessment` will be created. In the directory, you will find a Spring Boot application under `bookings` directory and database related files under the `data` directory.

All the necessary dependencies have been added to the Spring Boot application. You are free to add additional libraries if you wish to do so.

Create a git repository for the assessment on Github. This repository must initially be a **PRIVATE** repository. Configure `vttp2023_batch3_paf_assessment` so that the Github remote repository is the upstream repository.

You should commit and push `vttp2023_batch3_paf_assessment` directory to the remote repository. Do not wait until the end of the assessment.

Remember to make your repository **PUBLIC** after **1700 Fri Jul 21 2023** so that the instructors can access your work.

IMPORTANT: your assessment repository is PRIVATE and should only be accessible to yourself and nobody during the duration of the assessment and should only be made public **AFTER 1700 Fri Jul 21 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.

2. Provision deployment

Provision a Railway project. In the project you should add the following services

- an instance of MySQL database
- an instance of MongoDB database
- a service for running your Spring Boot application

You may provision MySQL or MongoDB instances on other cloud providers like Mongo Atlas, Digital Ocean, etc

Assessment

In this assessment, you will be writing an application for booking accommodations much like Airbnb. The application has 4 views; the flow is shown in Figure 1 below.

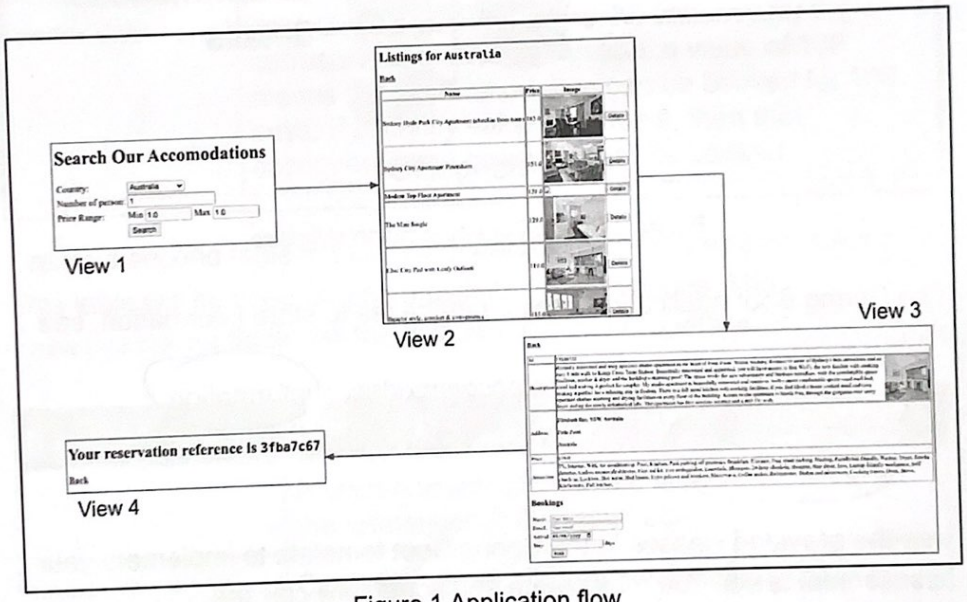

Figure 1 Application flow

- View 1 - The 'landing' page. Users uses this view to enter their search criteria
- View 2 - A summary list of accommodations that matches the user's search criteria
- View 3 - Details of an accommodation. Users can book the accommodation
- View 4 - Confirmation of the booking with a reservation reference id

More details will be provided in subsequent tasks.

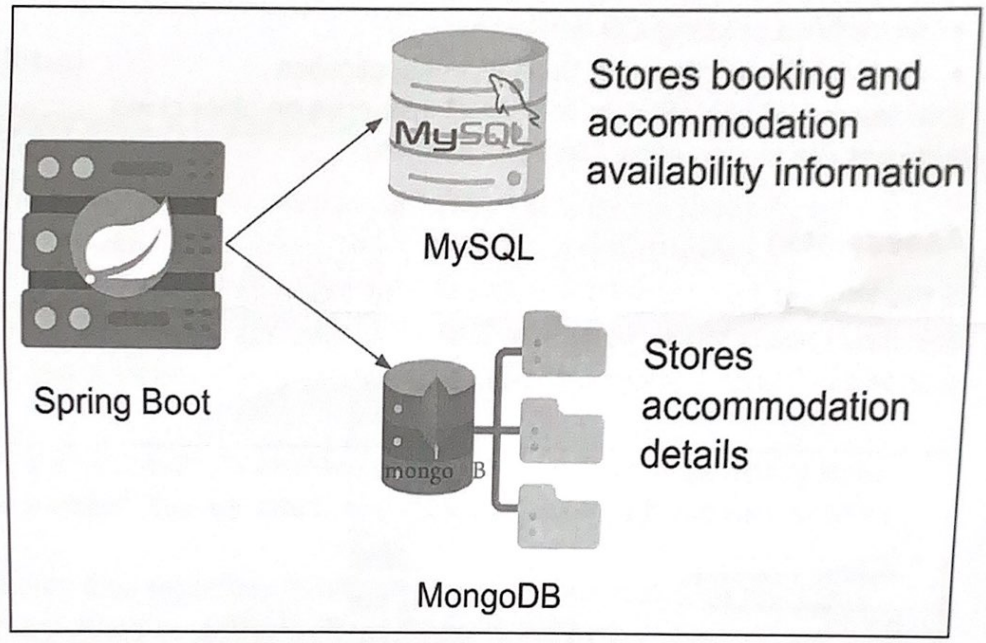

Figure 2 Spring Boot application structure

The Spring Boot application uses 2 databases to store information; see Figure 2

- MongoDB database stores accommodation information
- MySQL databases stores customer bookings and accommodation availability

Use the provided classes in the Spring Boot template to implement your assessment tasks. You may create additional new classes.

Task 1 (16 marks)

In the `data` directory, you will find 2 files

MONGODB

1 **`listing.json`**

Each document in this file represents the details of a single accommodation. Import this file into a Mongo database; call the collection `listings`. You may call the Mongo database any name.

Write the command you used to import `listing.json` into your Mongo database in a file called `mongo_import.txt` in the `data` directory. Do not modify the contents of the `listing.json`.

MYSQL.

`acc_occupancy.sql`

1 Create a MySQL database called `mybnb`. Create a table called `acc_occupancy` with the following column properties.

Name	Column Type
`acc_id`	Accommodation id, up to 10 characters. This column corresponds to the id attribute in `listings.json`.
`vacancy`	Integer. This column stores the vacancy for the corresponding accommodation; a value of 100 means the accommodation can be booked for 100 days. If vacancy value reaches 0, then that accommodation can no longer be booked

2 Create a second table in the `mybnb` database called `reservations`. This table will be used to record customer reservations. The `reservations` table has the following column properties

Name	Column Type
`resv_id`	A unique reservation id, 8 characters. This id is the reference for the reservation
`name`	Customer name, up to a maximum of 128 characters.

email	Customer's email, up to a maximum of 128 characters.
acc_id	Accommodation id. This should be a valid value in the acc_id column from the acc_occupancy table.
arrival_date	Date of arrival, date
duration	Number of days reserved

Write the following in a file called mybnb.sql in data directory

- Create the mybnb database
- Schema for creating acc_occupancy and reservations table described above

Execute mybnb.sql file to create the mybnb database. Once the database and tables have been created, execute the acc_occupancy.sql file to populate the acc_occupancy table. Do not modify the contents of the acc_occupancy.sql file.

Task 2 (16 marks)

Develop the 'landing page', View 1, of the application. The landing page allows users to filter their search. The landing page includes the following fields

- Country - the country of the accommodation "address.country"
- Number of person - the number of person staying "accomodates"
- Price range - accommodation's price range

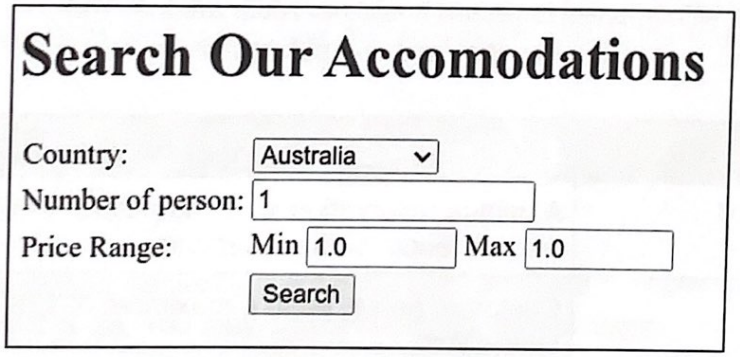

Figure 3 View 1

An example of View 1 is shown in Figure 3.

The country must be queried from the `listing` collection and not hard coded into your application.

When a search is performed, the fields should be validated according to the following constraints

- Country - must not be null or an empty string
- Number of person - a number be between 1 and 10 inclusive
- Price range - a number between 1 and 10000 inclusive

If the search criteria violates any of the above search constraints, return to View 1 with an appropriate error message.

You may layout View 1 according to your preference but all the required fields must be present.

Write the native Mongo query in a comment above the Java method which performs the Mongo operation. Marks will be awarded for the native Mongo query.

Implement this task in `ListingsController`, `ListingsService` and `ListingsRepository`. You may create additional classes. Marks will be awarded for proper 'layering' of your implementation.

Task 3 (26 marks)

When the 'Search' button is pressed, it makes the following HTTP request to the controller

```
GET /search?...
```

Write a request handler in `ListingsController` to process the above HTTP request.

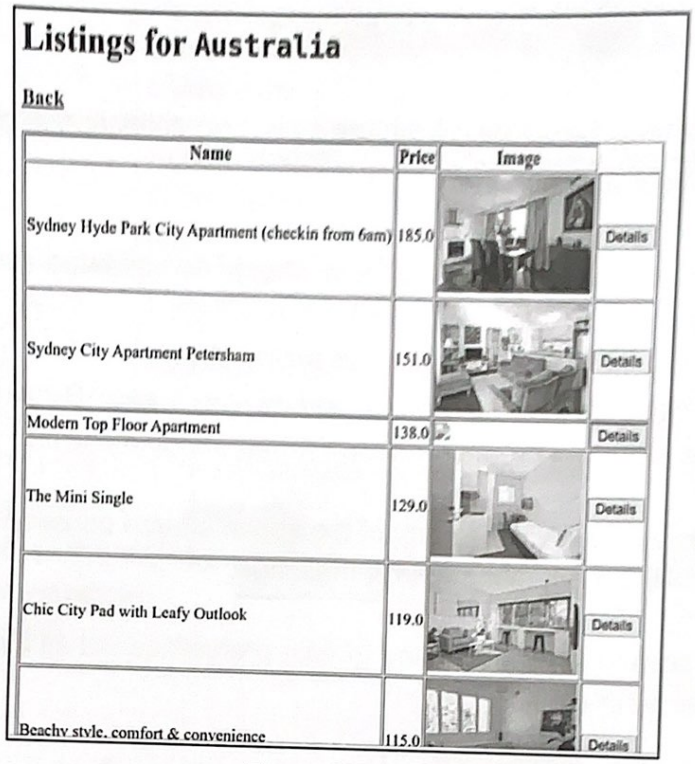

Figure 4 View 2

Find all listing documents that match the search criteria from the request according to the following requirements

- Country - match the `country` in the `address` attribute. The match should ignore case
- Number of person - match the `accommodates` attribute
- Price range - match the `price` attribute; the `price` attribute should be between the min and max price range.

Your Mongo query should only return the minimum amount of attributes to render and support the functionalities of View 2. The search result should be ordered by the accommodation price (`price` attribute) with the most expensive accommodation first (descending order by price).

aggregate

Write the native Mongo query in a comment above the Java method which performs the Mongo operation. Marks will be awarded for the native Mongo query.

Display the result of the search in View 2 (see Figure 4). The address, price and image from View 2 corresponds to `address.street`, `price` and `images.picture_url` attribute respectively from the `listings` collection.

Each accommodation should also have a corresponding 'Details' button; when this button is pressed, a detailed description of the accommodation is displayed in View 3 (next task).

th: if.

Display an appropriate message if the search returns no accommodation listing.

You may layout View 2 according to your preference but all the required fields must be present.

Implement this task in `ListingsController`, `ListingsService` and `ListingsRepository`. You may create additional classes. Marks will be awarded for proper 'layering' of your implementation.

Task 4 (16 marks)
When the 'Details' button is pressed in View 2, retrieve the details of the accommodation and display these in View 3 as shown in Figure 5.

View 3 should display the following

- Accommodation id - `_id` attribute STRING.
- Description - `description` attribute STRING
- Address - from `address.street`, `address.suburb` and `address.country` attributes STRINGS WITHIN OBJ.
- Image - from `images.picture_url` attribute STRING WITHIN OBJ.
- Price - from `price` attribute INT
- Amenities - from `amenities` attribute ARRAY

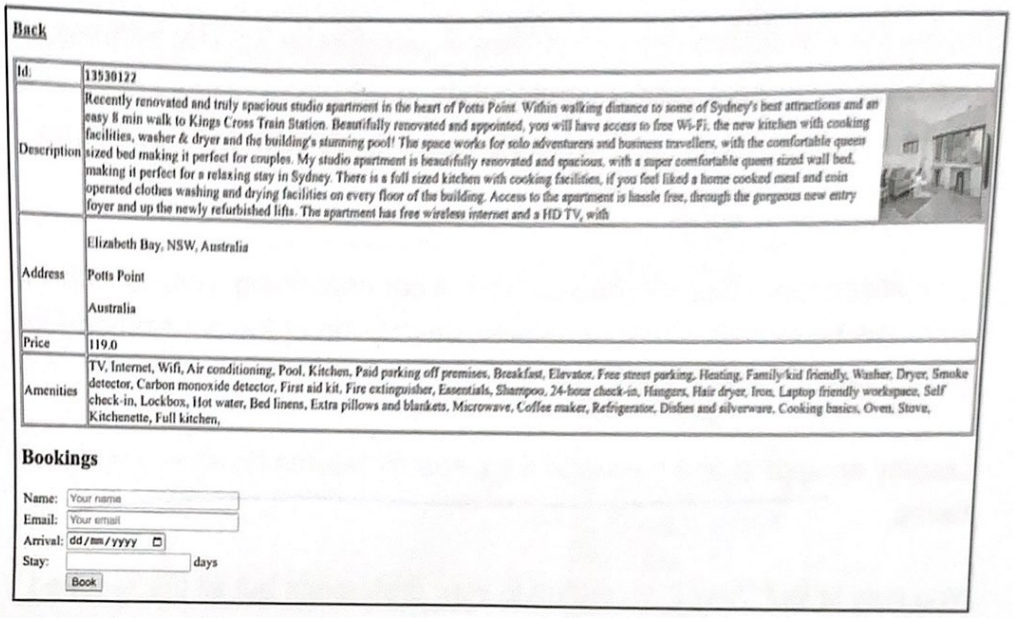

Figure 5 View 3

View 3 should also include a booking form for users to reserve the displayed accommodation. The booking form should include the following fields

- User's name
- User's email
- Arrival date
- Duration of stay in days

A 'Back' link (or button) returns the user to View 2. View 2 should redisplay the search results from the user's original search criteria.

hold search in session. OR Generate it with the query p

You may layout View 3 according to your preference but all the required fields must be present.

Write the native Mongo query in a comment above the Java method which performs the Mongo operation. Marks will be awarded for the native Mongo query.

☆ Display an appropriate message if the search returns no result.

Implement this task in `ListingsController`, `ListingsService` and `ListingsRepository`. You may create additional classes. Marks will be awarded for proper 'layering' of your implementation.

Task 5 (30 marks)

When the 'Book' button is pressed, a booking will be made for the accommodation with the details in the booking form.

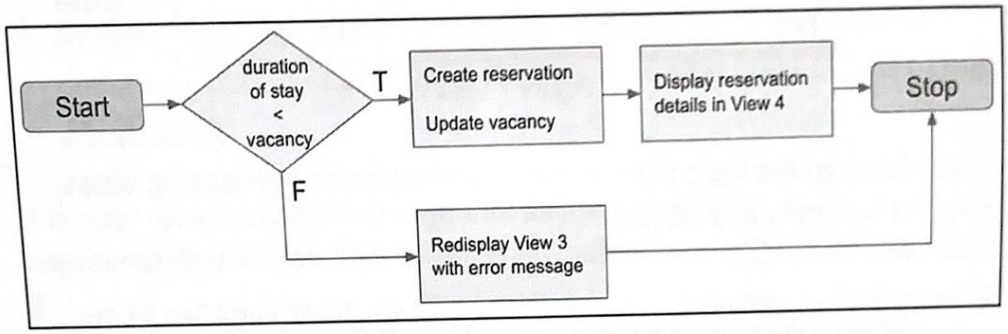

Figure 6 Booking process

Bookings are made according to the following described algorithm which is summarised pictorially in a flowchart in Figure 6

1. Check if vacancy is available for a particular accommodation; the duration of stay is compared against the `vacancy` field in `acc_occupancy` table. An accommodation is available only if the `vacancy` field is greater than or equal to the duration of the stay.
2. If the accommodation is available, generate a unique 8 character reservation id (`resv_id`) for the booking and add the booking details to the `reservations` table
3. The user's duration of stay is deducted from the `vacancy` field in `acc_occupancy` table. For example, if the duration of stay is 2 days, and the vacancy is 20 days, then you should deduct 2 from the 20 days.

- If an accommodation is not available, redisplay View 3 with an appropriate error message.

- If any of the booking details are missing or erroneous, redisplay View 3 with an appropriate error message.

You should ensure that the `vacancy` field consistently reflects the correct number of vacancy days during the entire booking process.

If the booking is successful, display the reservation id in View 4; an example is shown in Figure 7

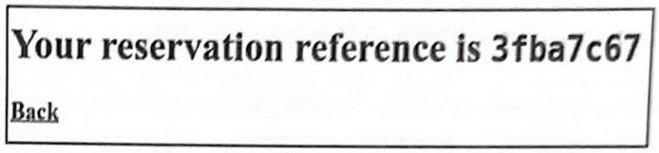

Figure 7 View 4

Note: To keep the logic simple, we will not consider overlapping dates, only the vacancy availability.

Implement this task in `ListingsController`, `ListingsService` and `ListingsRepository`. You may create additional classes. Marks will be awarded for proper 'layering' of your implementation.

Task 6 (10 marks)
Deploy your assessment to Railway.

Do not undeploy your application from Railway until after **0000 (12AM) Thur Jul 26 2023**.

Submission

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

Only commits on or before **Fri 1700 Jul 21 2023** will be accepted. Any commits after **Fri 1700 Jul 21 2023** will not be accepted. No other form of submission will be accepted (eg. ZIP file).

Your Railway deployment (**Task 6**) must be from a commit of your repo on or before **Fri 1700 Jul 21 2023**.

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

1. Your name (as it is shown in your NRIC)
2. Your email (as in Canvas)
3. Git repository URL. Remember to make it public after **Fri 1700 Jul 21 2023**
4. Railway deployment URL. The Railway deployment URL. Do not undeploy your application from Railway until after **0000 (12AM) Thur Jul 29 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 4 items mentioned above is missing from your details posted in `#paf-submission` channel, and/or
2. Your submission information did not comply with the submission requirements eg. not providing your full name as per your NRIC, and/or
3. Your repository is not publicly accessible **after Fri 1700 Jul 21 2023**

You should post the submission information to the Slack channel `#paf-submission` no later than **Fri 1715 Jul 21 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 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.