Software Development Assessment

Date: Thursday Jan 12 2023
Assessment Time: 0900 - 1700 (including meal breaks)

Overview

There are **4 tasks** in this assessment. Complete all tasks.

Passing mark is **65%** (**100 marks**). Total marks is 154.

Read this entire document before attempting the assessment. There are 13 pages in this document.

Assessment Setup

Create a git repository for the assessment. This repository must initially be a **PRIVATE** repository. Click on the 'Private' radio button when you create the repository.

All subsequent work must be saved in this repository. Do not submit more than 1 repository for your final assessment submission.

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

Make your repository **PUBLIC** after **1700 Thursday Jan 12 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 is only public **AFTER 1700 Thursday Jan 30 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.

Assessment

You are developing an application that allows anyone to sell their 2nd hand items by creating a listing and posting it online.

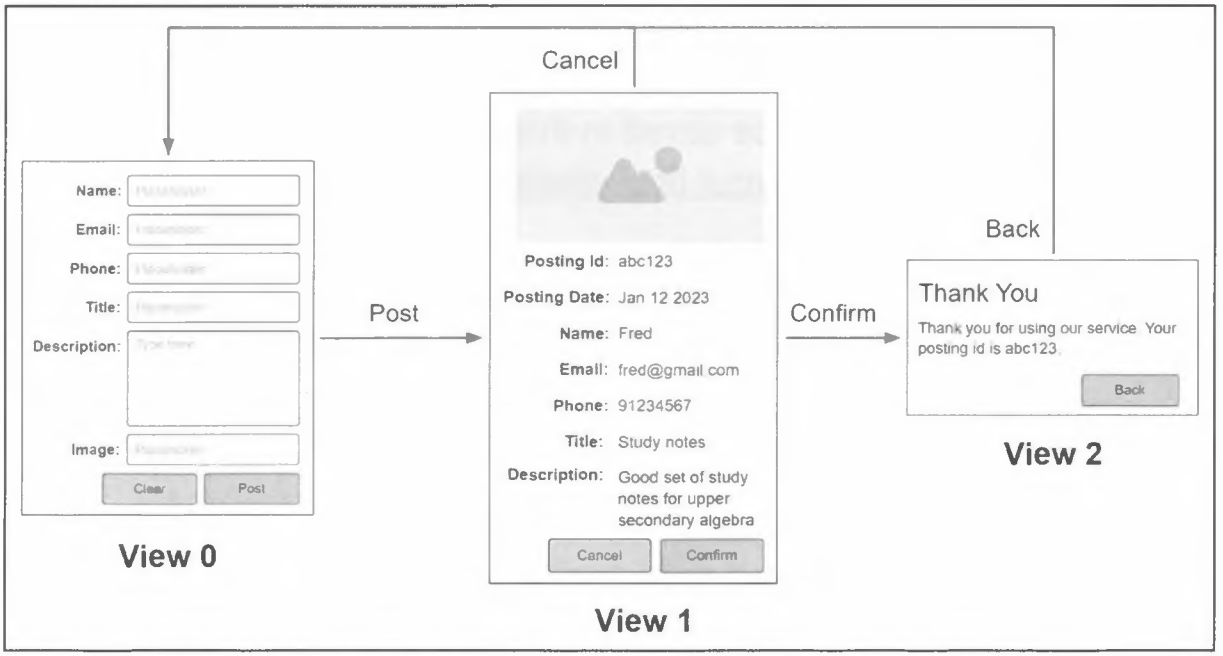
The application consists of 3 parts

1. An interactive frontend browser application written in Angular to create and post listings
2. A backend Spring Boot application to process and save the submitted listings
3. Databases to store the listing details.

The frontend application has 3 views:

- View 0 - create a listing for the customer's 2nd hand times
- View 1 - posting confirmation
- View 2 - item successfully posted

Figure 1 shows the flow between these views



* You need to create this may not done for title

Figure 1

* Need to check the date format

Detailed description of each view and what they do will be given in the subsequent tasks.

Task 1 (90 Marks)

This task describes the process of transitioning from View 0 to View 1.

Generate an application inside your repository. Call the Angular application `frontend`.

Create a component to allow a seller to post the details of the item that he/she is selling. The component should have a form with the fields described in Table 1 to capture the following details

Detail	Explanation
Name	Seller's name. This is a mandatory field. The name should be at least 3 characters in length
Email	Seller's email. This is a mandatory field. The email has maximum length of 128 characters
Phone	Seller's phone. This is not a mandatory field.
Title	The posting's title. This is a mandatory field. The title should not be shorter than 5 characters and no longer than 128 characters.
Description	The description of the item. This is a mandatory field.
Image	An image of the item. This is a mandatory field

Table 1

View 0 includes 2 buttons; Clear to clear the contents of the form and a Post button to submit the posting.

Figure 2 below is an example of View 0.

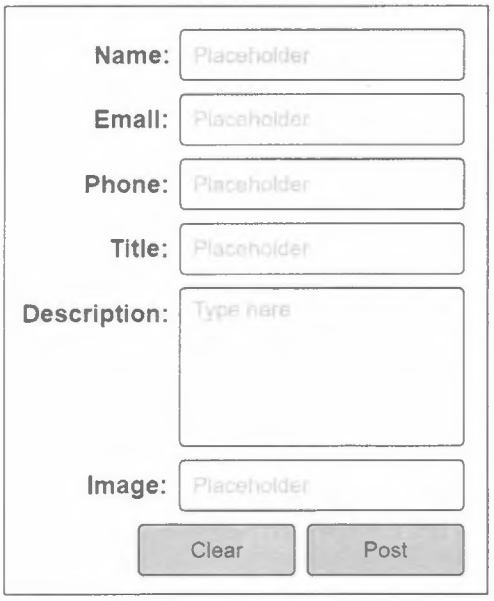


Figure 2 View 0

When the Post button is pressed, the frontend application send the following HTTP request to the backend Spring Boot application for processing

```
POST /api/posting
Content-Type: multipart/form-data
Accept: application/json
```

Write View 0 to allow a user to input their item details and perform the `POST` HTTP request to the backend. You are free to layout the form as long as it captures all the required information mentioned in Table 1.

Generate a Spring Boot application inside your repository to handle the above HTTP request from the frontend.

Hint: you will need to enable file upload
Hint: you have to decide how the Angular frontend application will be deployed with the backend Spring Boot; as a single application (same origin) or as 2 separate applications (cross origin)

You should also provision the following databases either on Railway or on any public cloud provider of your choice.

- Redis - if you are using Spring Boot 2.x then use Jedis 3.9.x. If you are using Spring Boot 3.x, then use Jedis 4.x.
- MySQL
- Spaces/S3 bucket

Configure your Spring Boot application to use these databases. Do not hardcode any sensitive information into the configuration.

Add all required dependencies to the `pom.xml` file.

Write a controller called `PostingController` to process the incoming HTTP `POST` request. The `POST` request handler should perform the following when it receives the request

- Generate a posting id for the posting. The posting id is a unique string of 8 characters. You can use the `UUID` class to generate the posting id.
- Posting date, the current date
- Save the posting together with the posting id and the current date to the Redis database. Use the posting id as the key; the posting should be saved as a JSON string as shown below. The posting will be temporarily stored in Redis until it is confirmed by the seller. Set a time to live (TTL) on this entry for 15 minutes.
- Save the image to Spaces/S3 bucket. Make the image publicly accessible. You should also set the content type and size.

When you have completed the above 3 tasks, send the following JSON document back to the frontend with an OK status code.

```
{
    postingId: <generated posting id>,
    postingDate: <posting date>,
    name: <name>,
    email: <email>,
    phone: <phone>,
    title: <title>,
    description: <description>
    image: <image URL from Spaces/S3>
}
```

When the response is received from the frontend, transition from View 0 to View 1 and display the details from the received JSON response as shown in Figure 3.

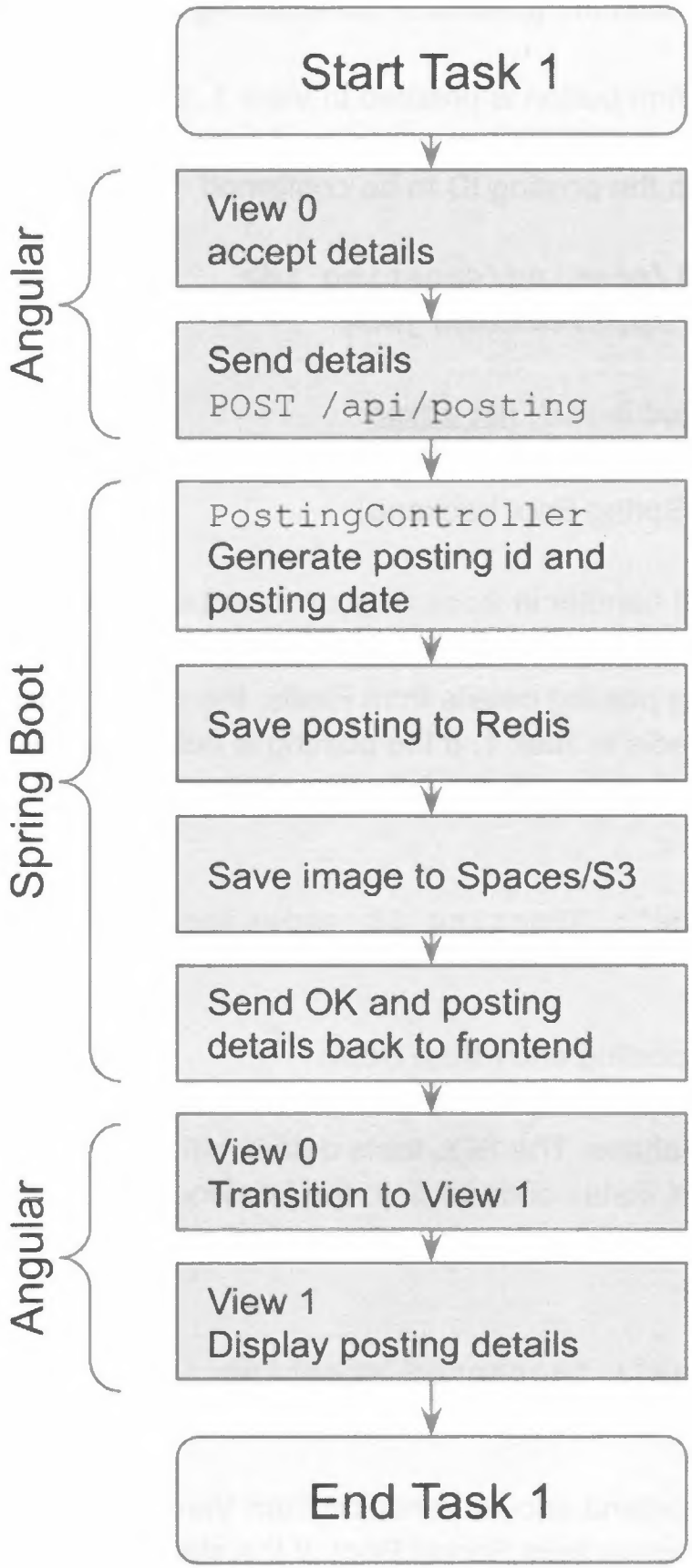
Figure 3 View 1

All the details in View 1 are not editable. You are free to layout View 1 but all specified details must be present.

If the Cancel button is pressed, navigate back to View 0. See Figure 1.

If the Confirm button is pressed, the posting will be confirmed. The confirmation process will be described in Task 2. Write View 1.

The following flowchart summarises the process in Task 1.



Task 2 (42 marks)

This task describes the process of transitioning from View 1 to View 2.

When the Confirm button is pressed in View 1, the Angular frontend sends the following HTTP request to the Spring Boot backend to confirm the posting with the posting ID to be confirmed

```
PUT /api/posting/<posting id>
Accept: application/json
```

Note: the method is `PUT`, not a typo.

Add an event handler to View 1's Confirm button to send the `PUT` request to the Spring Boot backend.

Write a request handler in `PostingController` to process the `PUT` request. The `PUT` request handler should perform the following

- Retrieve the posting details from Redis; the posting was previously saved to Redis in Task 1. If the posting is not found, return a Not Found status code with the following payload

```
{
    "message": "Posting ID <posting id> not found"
}
```

- Delete the posting entry from Redis getAndDelete()
- Save the details of the posting to a table called `postings` in the MySQL database. The SQL table details will be provided in Task 3.
- Send an OK status code back to the frontend with the following payload

```
{
    "message": "Accepted <posting id>"
}
```

The Angular frontend should transition from View 1 to View 2 when it receives a response from Spring Boot. If the status is OK, View 2 should

display the posting ID with the message "Thank you for using our service. Your posting id is <posting id>"

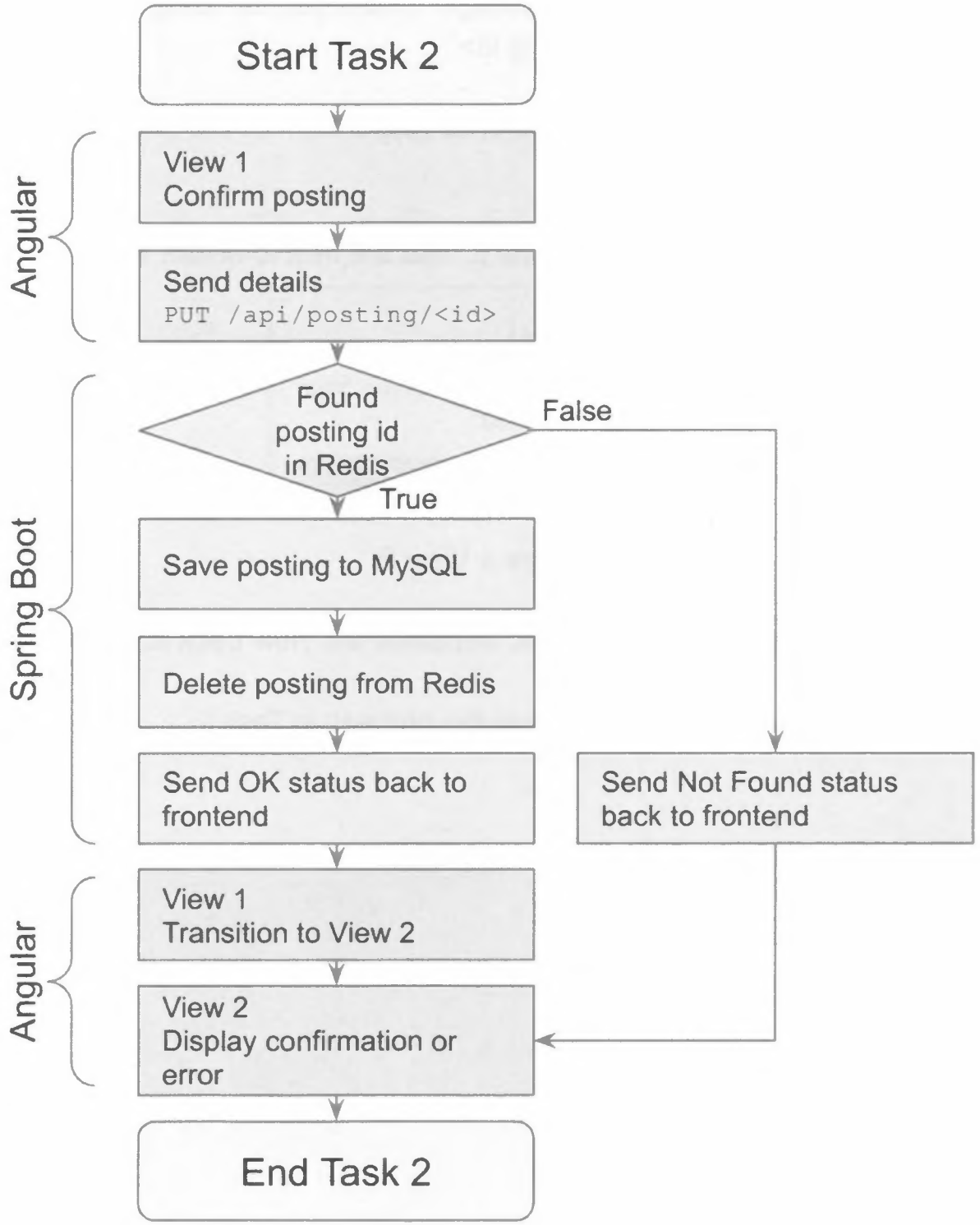
If the status is an error, display the error response with the JavaScript `alert()` function.

Figure 4 shows an example of View 2. You are free to layout View 2.

Figure 4 View 2

Write View 2. Add a Back button to transition the view back to View 0.

The following flowchart summarises the process in Task 2.


Start Task 2
Angular
View 1
Confirm posting
Send details
PUT /api/posting/<id>
Spring Boot
Found posting id in Redis
False
True
Save posting to MySQL
Delete posting from Redis
Send OK status back to frontend
Send Not Found status back to frontend
Angular
View 1
Transition to View 2
View 2
Display confirmation or error
End Task 2

Task 3 (12 marks)

This task describes the MySQL database and `posting` table for saving the postings when a posting is confirmed.

Create a file called `schema.sql` in your repository. Write the following SQL statements in schema.sql

- Create a database called `second_hand`
- Create a table called `postings` with the following columns (Table 2)

Column Name	Description
`posting_id`	Unique identifier representing the order id
`posting_date`	The date when the posting is created, not null
`name`	Customer's name, not null
`email`	Customer's email, max length of 128 characters
`phone`	Customer's phone. Defaults to an empty string if not set
`title`	Posting title, max length of 256 characters
`description`	Posting description. Use a `text` for this column
`image`	URL to the bucket of the image from Task 1, max length of 256 characters

Table 2

Use the `schema.sql` file to create your MySQL database.

Task 4 (10 marks)

Deploy the application to Railway. You may choose to deploy the application as

- A single application where the Angular frontend is served by the Spring Boot backend application (same origin style), or
- Two separate applications where the Angular frontend is hosted on a separate platform (eg. Vercel) accessing Spring Boot on Railway (cross origin style)

Remember to set all your sensitive data as environment variables and not hardcoded into the source code or the application property file.

Do not undeploy your application until after **0000 (12AM) Saturday January 21 2023**.

Submission

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

Only commits on or before **1700 Jan 12 2023** will be accepted. Any commits after **1700 Jan 12 2023** will not be accepted. No other form of submission will be accepted (eg. ZIP file).

Your application deployment (**Task 4**) must be from a commit of your repo on or before **1700 Jan 12 2023**.
Remember to **make your repository public after 1700 Jan 12 2023** so the instructors can review your submission.

After committing your work, post the following information to Slack channel `#05-submission`

1. Your name (as it is shown in your NRIC)
2. Your email (as your LumiNUS)
3. Git repository URL. Remember to make it public after **1700 Jan 12 2023**.
4. Railway deployment URL. Please do not undeploy your application from Railway until after **0000 (12AM) Saturday January 21 2023**. If you are deploying your application as 2 separate deployments (cross origin style) then post your Angular deployment URL (eg Vercel URL) and your backend Railway deployment URL

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 `#05-submission` channel, and/or
2. Your 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 1700 Jan 12 2023**.

You should post the submission information to the Slack channel `#05-submission` **no later than 1715 Jan 12 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.