Persistence and Analytics Fundamentals Assessment

Date: Friday April 29 2022
Reading Time: 0900 - 0915
Assessment Time: 0915 - 1730 (including meal breaks)

Overview

In this assessment you will be writing a Spring Boot application to save a user's task list into a database.

There are **11 tasks** in this assessment. Complete all tasks.

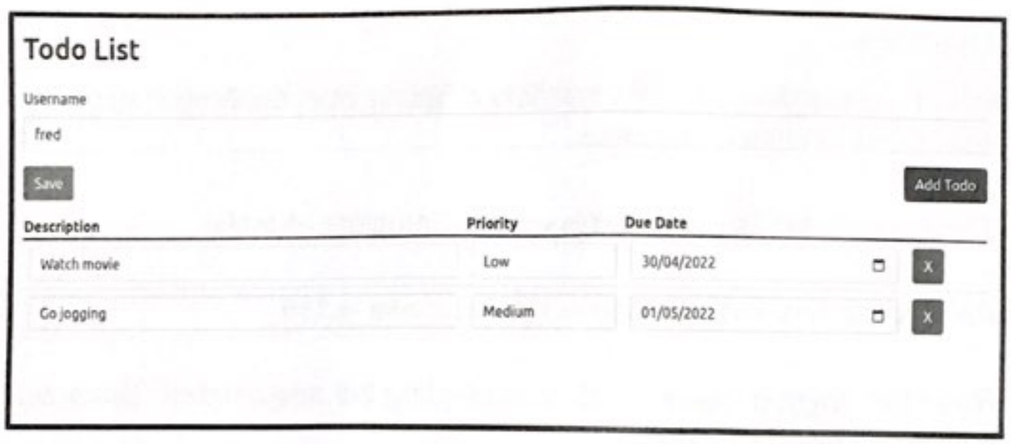
Passing mark is **70% (80 marks)**. Total marks is **115**.

Read this entire document before attempting the assessment. There are 9 pages in this document.

Application Overview

This application has only a single view (View 1) shown in Figure 1. Users enter their username and a list of tasks to be saved to the database.

Each task consists of a description of the task, its priority and the due date. A delete button allows the user to remove the task from the list.



Username: Fred

Description	Priority	Due Date
Watch movie	Low	30/04/2022
Go jogging	Medium	01/05/2022

Figure 1

The 'Add Todo' button adds a new task to the list. The 'Save' button saves the task to the server. After successfully saving the task, all data from the view will be cleared.

Details of the view will be provided in subsequent tasks below.

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, Thymeleaf, MySQL and JaCoCo library. 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 the repository has been created, add Kenneth (`kenken64`) and Chuk (`chukmunnlee`) as collaborators. See https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-github-user-account/managing-access-to-your-personal-repositories/inviting-collaborators-to-a-personal-repository (Or Google 'inviting collaborators to a personal repository').

Add/link the Maven project to the remote repository that you have just created.

You should now perform an initial commit and push of your project to the remote repository.

IMPORTANT: this repository is PRIVATE and is only accessible to yourself, Kenneth (`kenken64`) and Chuk (`chukmunnlee`); nobody else should have access to it. If your work is plagiarised by others, you will be considered as a willing party in the aiding and abetting of the dishonest act.

Task 2 (5 marks)

In the `database` directory, you will find a file called `data.csv` which contains the record of 4 users.

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

Field name	Description
`user_id`	This is the unique identifier of the record. It is a randomly generated 8 character long hex string
`username`	Username name without spaces or special characters
`name`	Friendly name, may contain spaces. This is an optional field

Create a database base; using the above specifications and columns from `data.csv`, create a table, called `user` with the provided field names. Write all the SQL statements for creating the database and the table in a file called `schema.sql` in the `database` directory.

Use the `schema.sql` to create the database in the MySQL database hosted in Digital Ocean. You can use an existing MySQL database or create new instances.

Give a MySQL user (a non database administrator/root) access to the database you have just created. Your application should use this user to access the database.

Task 3 (5 marks)
Populate the database you have created in **Task 2** with data from `data.csv`.

Create a file called `data.sql` in the `database` directory; write SQL insert statements in `data.sql` to insert records from `data.csv` into the database.

Execute `data.sql` to populate the database.

Task 4 (15 marks)
Write the following 2 methods to manipulate the database in the provided `UserService.java` file under `services` directory.

`public Optional<User> findUserByUserId(String userId)`
Find the user with the `user_id`. Return an `Optional` object to indicate if the search was successful.

`public String insertUser(User user)`
Insert a new user into the database. `insertUser()` method should generate a random `user_id` for this new user. Return the generated `user_id` from the `insertUser()` method when the user has been successfully added to the database.

Hint: you can use the `UUID` class (https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/UUID.html) to generate a random 8 character long hex string.

Task 5 (10 marks)
Run the Spring Boot application; open `localhost:8080`. You should see Figure 1 as the landing page.

Write a handler to process the request from the landing page. Use the provided class `TaskController.java` to write the request handler.

The task from the request should be saved into a model called `Task` defined in `Task.java` file in `models` directory. Fill in the `Task` model with the appropriate members and also their corresponding getters and setters.

Task 6 (5 marks)
Create a table to save tasks in your database created in Task 2. Call this table `task`; `task` table should have the following fields

Field name	Description
`task_id`	This is an auto generated sequential number
`description`	Task's description up to a maximum of 255 characters
`priority`	As an integer; bigger value indicates higher priority. Currently there are only 3 levels of priority: 1 to 3 (inclusive)
`due_date`	The task's due date

The task table (`task`) has a one to many (1:m) relationship with the user table (`user`) created in Task 2.

Write the SQL statements to create the task table in the `schema.sql` file according to the above specifications.

Execute the relevant SQL statements to create the `task` table.

Your database should now have 2 tables.

Task 7 (marks 15)
Write a method in the `TaskRepository.java` file to insert a task into the `task` table. Write the method according to the following specification

- Method name should be called `insertTask`
- You are free to decide on the method's parameters; but one of these parameters should be the `Task` model.
- You are free to decide if the method returns any value.

Task 8 (marks 20)
Create a method in `TodoService` class (in `services` package) according to the following specification

- Method name should be called `upsertTask`
- You are free to decide the parameters and if the method should return any values

The `upsertTask()` method is used to insert a list of tasks for a particular user into the database. If the user does not exist, `upsertTask()` will first create the user before inserting the tasks.

The following flowchart in Figure 2, summarises the `upsertTask()` algorithm.

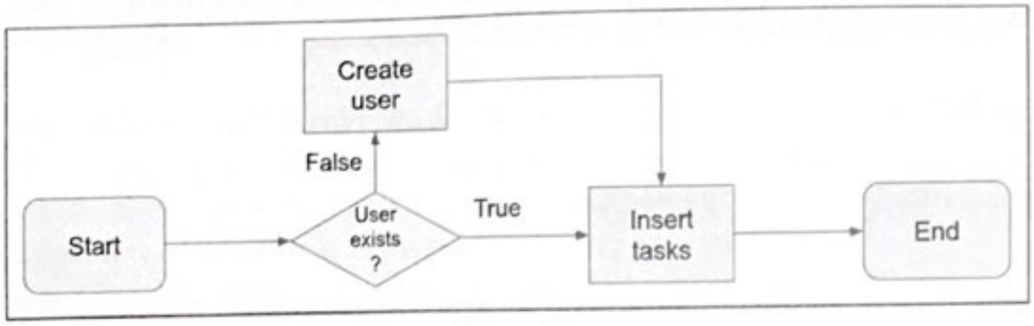


Figure 2

Since `upsertTask()` is updating multiple tables (and inserting multiple records), ensure that multiple updates to the database either succeed together or are rolled back if any one of the insert operations fails.

Task 9 (10 marks)
Complete the controller from Task 5. Use the `upsertTask()` method to insert tasks from the request into the database.

If the insert operation is successful, return the `result.html` view (in `templates` directory) and a 200 HTTP status code. Hint: examine the `results.html` Thymeleaf template.

If the insert operation is not successful or the request handler encounters any errors/exceptions, return the `error.html` view (in `templates` directory) and a 500 HTTP status code.

Task 10 (20 marks)
Write one or more tests to validate the correctness of `upsertMethod()`.

IMPORTANT: Your tests must cover a minimum of 80% of `TodoService`.

Task 11 (10 marks)
Deploy the application to Heroku. The deployment should be based on any commits on or before **1730 Apr 29 2022 SGT**. Do not undeploy your application before **2359 Friday May 5 2022 SGT**.

Submission

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

Only commits on or before **1730 Apr 29 2022 SGT** will be accepted. Any commits after **1730 Apr 29 2022 SGT** will not be accepted. No other form of submission will be accepted (eg. ZIP file).

Remember to make your repository private and add Kenneth (`kenken64`) and Chuk (`chukmunnlee`) as the repository's collaborators.

After committing your work, post the following information to Slack channel `#02-ssf-submission`

1. Your name (as per NRIC)
2. Your deployed application's URL on Heroku
3. Git repository URL - Is the repository private and have you added the `kenken64` and `chukmunnlee` as collaborators?

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 3 items mentioned above is missing, and/or
2. you information did not comply with the submission requirements eg. not providing your full name as per your NRIC, incorrect email subject line, forgetting to post the Heroku URL, etc, and/or
3. the repository is not private and/or Kenneth or Chuk cannot access the repository.

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.