Advanced Programming

COSC1295

Assignment 2: Data Analytics Hub

Assessment Type	**Individual assignment; no group work.** Submit online via Canvas → Assignments → Assignment 2. Marks are awarded for meeting requirements as closely as possible according to assignment specifications and the supplied rubric. Clarifications or updates may be made via announcements.
Due Date	**Week 13, Friday 20 October 11:59pm.** Late submission penalty will be applied unless special consideration has been granted. **There will be 2 milestones: week 12 in-lab milestone check and week 14 post submission final interview. You will describe the key concepts adopted in your code, your overall application design, demonstrate code execution, and explain your code at the milestones. No extensions to the Week 12 milestone will be given unless special consideration has been granted.**
Marks	45 marks out of 100 for Assignment 2 5 marks for week 12 in-lab milestone check in addition to 45 marks.

1. Overview

NOTE: Carefully read this document. In addition, regularly follow the Canvas assignment discussion board for assignment related clarifications and discussion.

- In this assignment, you are required to work individually to develop a GUI Application, named **Data Analytics Hub**, which allows user to import, analyze, and get insights of social media data. You will practice your knowledge of object-oriented design principles, design patterns, and GUI-frontend application design. You will use Java SE 8 or later and JavaFX. A suite of suitable JUnit tests should be provided to demonstrate correct functionality.

The assignment can be seen as an extension of Assignment 1. Hence, you can re-use your code for Assignment 1 if you wish.

Task specification

Stage 1 - Basic functional requirements.

- The application can have multiple users.
- Each user can create a profile, with a username, password (may be unencrypted), first name, and last name. The username should be a unique identifier of the user in the application.
- Once the username and password are created, the user can log in.
- Each user has a dashboard after login. The dashboard should display a welcome message containing the user's full name.
- Each user can keep a collection of social media posts and analyze it. Specifically, a user can perform the following actions:
    - Edit the profile (i.e., change the first name, last name, username, and the password).
    - Add a social media post to the collection. A social media post should contain ID, content, author, #likes, #shares, and date-time (See Assignment 1 -> Task specification for more details). The user will provide

the details of one post. Afterwards, the post can be added to the collection. For each post, validation should be done to ensure the post has correct format, such as #likes being a number, ID being unique, etc.

- Retrieve a post from the collection based on post ID. The user is allowed to enter a post ID, and the application should retrieve the corresponding post if it exists.
- Remove a post from the collection based on post ID.
- Retrieve the top N posts with the most likes, for a given user or over ALL users, and show retrieved posts in descending order of #likes. The user is allowed to specify the value of N.
- Export a post to a file based on post ID. The user should be allowed to choose a folder for storing the file, and specify the file name to be exported. The saved file should be in csv format.
- Log out

Stage 2 – VIP functionalities

When a new user is registered, the user is considered as a non-VIP user. Once a non-VIP user logs in, the user has the option of upgrading to VIP user. To upgrade, the application asks if the user agrees to subscribe to the application:

```
Would you like to subscribe to the application for a monthly fee of $0?
```

The user will become a VIP user once they agree:

```
Please log out and log in again to access VIP functionalities.
```

For simplicity, you do not need to collect payment information in this assignment. You can assume that the user agrees by clicking "yes". A VIP user has advanced functionalities as follows.

- Data visualization - Draw a pie chart showing the distribution of #shares. The application can display a pie chart that illustrates the distribution of #shares by categorizing all posts into three groups: #shares ranging from 0 to 99, #shares between 100 and 999, and #shares equal or exceeding 1000. Generate a pie chart that displays the proportions of these three categories.
- Bulk import social media posts. Import multiple social media posts from a csv file altogether. A posts.csv file is provided for testing. This functionality should capture potential exceptions occurred during import, such as the case where the csv file does not follow the correct format.

Once a user has become VIP, the same user should always have access to advanced functionalities when logging in.

***Disclaimer:** the specification in this assignment is intended to represent a simplified version of a system in real life and thus is not meant to be a 100% accurate simulation of any real system or service that is being used commercially.*

4. Assessment details

In this assignment, you will incrementally build a GUI application, named **Data Analytics Hub**. The assignment is structured into 2 milestones which must be demonstrated to your lab tutor.

Milestone 1

In Milestone 1, you are expected to complete the following basic functionalities:

- Sign up
- Log in
- User profile editing
- Add a social media post to the application
- Retrieve a post from the collection based on post ID.

Milestone 2

You are expected to complete functionalities of Stage 1 and Stage 2. In addition, you are required to incorporate object-oriented concepts to design your program (including abstraction, encapsulation, inheritance, and polymorphism). You are **required to** follow MVC design pattern to connect the frontend with the backend. You are also **required to** use another design patterns such as Singleton pattern, besides the MVC pattern.

5. Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods,
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to https://www.rmit.edu.au/students/student-essentials/rights-and-responsibilities/academic-integrity

6. Marking Guidelines

6.1 Milestone check 1

- In-lab milestone check in week 12 (5 marks): Show a basic class design and clearly explain how the code follows MVC; complete required functionalities and demo via code execution (see Assessment Details -> Milestone 1).

6.2 Milestone check 2 (Final milestone)

The marking guideline for Milestone check 2 is as follows.

Final submission marking guidelines:

- GUI (7/45)
- Functionality (20/45)
- Object oriented design and choice of data structures (10/45) (include a basic design diagram in your submission)
- Source code quality (5/45)
- Regular incremental updates in Github (3/45)

GUI: The application is easy-to-navigate and consistent and responds clearly to every user action.

- Your program should include appropriate JavaFX components to make the application easy-to-navigate. For example, each window should have a window title; menu options (if applicable) are selected from a menu bar.
- Your program should respond clearly to every user action. For example, if the user types a wrong password, the login window should display a clear error message and ask the user to type again.

- Your program should provide visual consistency. Avoid using different styles and labels for similar elements on different windows of the application.
- The UI design of your program should serve the purpose of the application.

Note marks for GUI are checked during the post-submission interview. If you do not attend the final interview, you will be awarded ZERO marks for GUI.

Functionality: All functions required by specification are implemented correctly covering **different cases** that might happen during the usage of the program.

Note marks for Functionality are checked during the post-submission interview. If you do not attend the final interview, you will be awarded ZERO marks for Functionality.

Object oriented design and choice of data structures: Appropriate choice of data structures that serves the purpose of the application. Class designs that adhere to the SOLID principles.

- You are **required to** follow MVC design pattern to connect the frontend with the backend.
- You are **required to** use one of design patterns such as Singleton pattern, besides the MVC pattern.
- You are **required to** apply SOLID principles whenever suitable in order to (1) enhance the maintainability and extensibility of your program; (2) decrease the coupling between classes; (3) minimize the repetition of codes/methods across classes.
- You are **required to** choose appropriate data structures to enhance the running efficiency of the program.

Source code quality: Adequately documented and properly indented code; appropriate class/method/variable names.

6.3 Other important notes

- You **MUST** use JavaFX.
- Your program should save the user profile data and allow the application to be restarted in the same state it was in at the end of the previous execution. For example, if the program is ended and restarted, an existing user should still be able to log in and view his/her own data collection. You may store the data in a database (using JDBC) when the application closes and restore the data when the application starts again or in files and read from disk on restart.
- You will need to document the code properly by following Code Conventions for the Java Programming Language by Oracle: https://www.oracle.com/java/technologies/javase/codeconventions-introduction.html
- You MUST regularly save your code to GitHub to demonstrate regular incremental development of your code