PROG 2200 FINAL PROJECT

ALERTS NOTIFICATION SYSTEM

Assignment Value: 40% of overall the course mark.

Due Date: **29th Nov (**See due date designated on the assignment on D2L.)
Late submissions will receive the standard late submission penalty as stated in the course outline. (5% overall deduction per day late)

Assignment Instructions:

Use IDE to create applications (Java files) in which you'll code the solution for the given problem.

Submissions:

You will submit your work for this assignment via GitHub. A GitHub Repo should include all required Java files along with console output. **You must upload the solution to the public GitHub repo.**

Evaluation:

To ensure the greatest chance of success on this assignment, be sure to check the marking rubric contained at the end of this document or in D2L. The rubric contains the criteria your instructor will be assessing when marking your assignment.

Final Project: Alerts Notification System

The project you're going to be working on is the SafetyNet Alerts application. Its main purpose is to send emergency information to emergency responders.

For instance, if there is a fire, we need SafetyNet Alerts to bring up information about the people in the house that is on fire, such as their phone numbers. Another example is if there is a severe storm warning, we want SafetyNet Alerts to notify all of the people in the area of the storm via text message. To do this we need SafetyNet Alerts to get the phone numbers of every resident who lives in homes near the storm area. In the event of a flood alert, we want to dispatch responders with specialised information about everyone in the area. We need to know every person who could be in the flood, their ages, and any specific medical information about them, such as their medications and allergies. With SafetyNet Alerts, we hope to dispatch more prepared and informed first responders.

Requirements: -

SafetyNet Alerts need to have Springboot endpoints that yield information about its status.
Once it reads the data file containing the names and addresses, we will need SafetyNet Alerts to produce JSON output from the corresponding URLs (Try to implement at least 4 APIs).

- **http://localhost:8080/firestation?stationNumber=<station_number>**

This URL should return a list of people serviced by the corresponding fire station. So if station number = 1, it should return the people serviced by station number 1. The list of people should include these specific pieces of information: first name, last name, address, and phone number. As well, it should provide a summary of the number of adults in the service area and the number of children (anyone aged 18 or younger).

- **http://localhost:8080/childAlert?address=<address>**

This URL should return a list of children (anyone under the age of 18) at that address. The list should include the first and last name of each child, the child's age, and a list of other persons living at that address. If there are no children at the address, then this URL can return an empty string.

- **http://localhost:8080/phoneAlert?firestation=<firestation_number>**

This URL should return a list of phone numbers of each person within the fire station's jurisdiction. We'll use this to send emergency text messages to specific households.

- **http://localhost:8080/fire?address=<address>**

This URL should return the fire station number that services the provided address as well as a list of all of the people living at the address. This list should include each person's name, phone number, age, medications with dosage, and allergies.

- **http://localhost:8080/flood/stations?stations=<a list of station_numbers>**

This should return a list of all the households in each fire station's jurisdiction. This list needs to group people by household address, including name, phone number, and age of each person, and any medications (with dosages) and allergies beside each person's name.

- **http://localhost:8080/personInfo?firstName=<firstName>&lastName=<lastName>**

This should return the person's name, address, age, email, list of medications with dosages and allergies. If there is more than one person with the same name, this URL should return all of them.

- **http://localhost:8080/communityEmail?city=<city>**

This will return the email addresses of all of the people in the city

For all of these endpoints, the results should be JSON, and if there is an address or fire station number not found within our file, it should return an empty JSON object. We'll also need a set of unit tests that test each of the requirements and validate that the application working correctly. Also, don't forget about logging. The application should log every request and every response.

Also, SafetyNet Alerts need to be architected following the Model-View-Controller design pattern. If you follow the SOLID principles and separate your model from your controllers, you'll be right in line with our coding best practices. When coding up the application, please commit often, so that we can validate little by little; this will help us piece together the overall flow of development, and give us an idea of lines of code per day and other metrics that will be helpful in gauging the timeline for the next phase of SafetyNet Alerts.

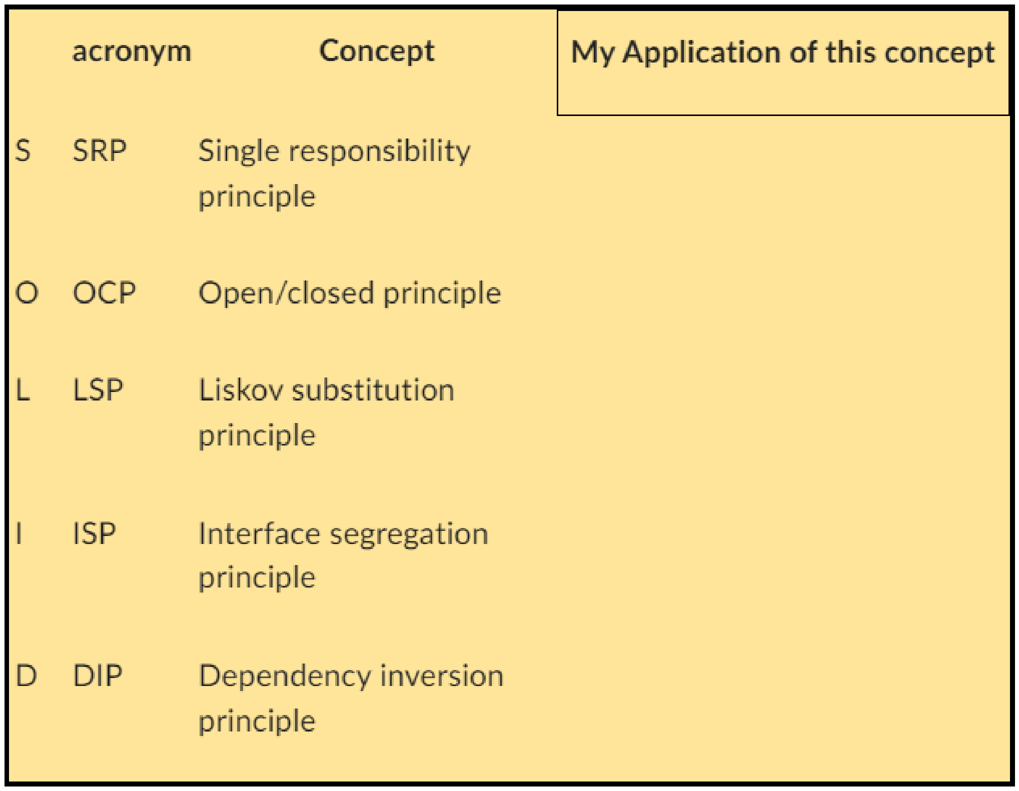
When you've finished SafetyNet Alerts, you prepare a presentation with a description of your solution. This document will serve as additional documentation to other team members who will be working on SafetyNet Alerts during our next phase of development.

Your solution must contain examples demonstrating your understanding of the appropriate use of Java language concepts.

Submission: -

As part of this final project, please follow the below checklist while submitting.

- Source Code on GitHub
- Screen Recording to explain the system workflow.
- Presentation to explain your solution.
- SOLID Table:
    - o Copy the table below into your PDF. For at least one concept, describe how you implemented it.



	acronym	Concept	My Application of this concept
S	SRP	Single responsibility principle	
O	OCP	Open/closed principle	
L	LSP	Liskov substitution principle	
I	ISP	Interface segregation principle	
D	DIP	Dependency inversion principle	

Program 1 – Alerts Notification System

Criteria	Insufficient (0 pts)	Needs Development (3-5 pts)	Sufficient (7 pts)	Excellent (10 pts)	Mark
Submissions: GitHub Source Code & Screen Recording	• Little to no effort was made or contains too many errors/omissions.	• A reasonable effort was made, but there are multiple areas for improvement.	• A good effort was made, but at least one error or omission exists.	• An extended effort was made, and go beyond the mentioned requirement.	/10
In-code Documentation & Code Quality	• Little to no effort was made or contains too many errors/omissions.	• A reasonable effort was made, but there are multiple areas for improvement.	• A good effort was made, but at least one error or omission exists.	• An extended effort was made and go beyond expectations. Also demonstrated a strong understanding of the in-code documentation and code quality.	/10
System Design & Solution: Dynamic Input/Output, Fulfill all the mentioned requirement	• Little to no effort was made or contains too many errors/omissions.	• A reasonable effort was made, but there are multiple areas for improvement.	• A good effort was made, but at least one error or omission exists.	• An extended effort was made and go beyond the mentioned requirement. Also demonstrated a strong understanding of the solution design.	/10
Java/Spring Concepts: Variables, Datatypes, Logic control statements, Arrays, Restful API , File I/O, and a lot.	• Little to no effort was made or contains too many errors/omissions.	• A reasonable effort was made, but there are multiple areas for improvement.	• A good effort was made, but at least one error or omission exists.	• An extended effort was made and demonstrated a strong understanding of the Java Language concepts.	/10
				Total:	**/40**