FINAL PROJECT - ALERTS NOTIFICATION SYSTEM

PROG2200

ALICIA SEKIGUCHI (GUPONY)
Github: https://github.com/Fall2024-NSCC-ECampus/final-project-alerts-notification-system-gupony

Final Project Summary: Alerts Notification System

For my final project in the PROG 2200 course, I worked on the **SafetyNet Alerts** system. The objective of the project was to implement a Spring Boot application that sends emergency notifications, such as fire, storm, and flood alerts, to both first responders and the general public. The system retrieves vital information about individuals in affected areas—such as their phone numbers, addresses, medications, and allergies—to help prepare emergency responders.

I followed the **Model-View-Controller (MVC)** design pattern and adhered to **SOLID principles** to ensure the application was scalable and maintainable. The system includes several RESTful endpoints that provide various types of emergency information.

The core functionality of the application is as follows:

1. **Fetching details of fire stations and the people they serve** based on station number or address.
2. **Identifying children in a specific household** or area, which is critical in emergency situations.
3. **Providing phone numbers for emergency alert systems** based on fire station jurisdictions.
4. **Retrieving specific details about residents in affected areas**, such as their medical information, to ensure responders are better prepared.

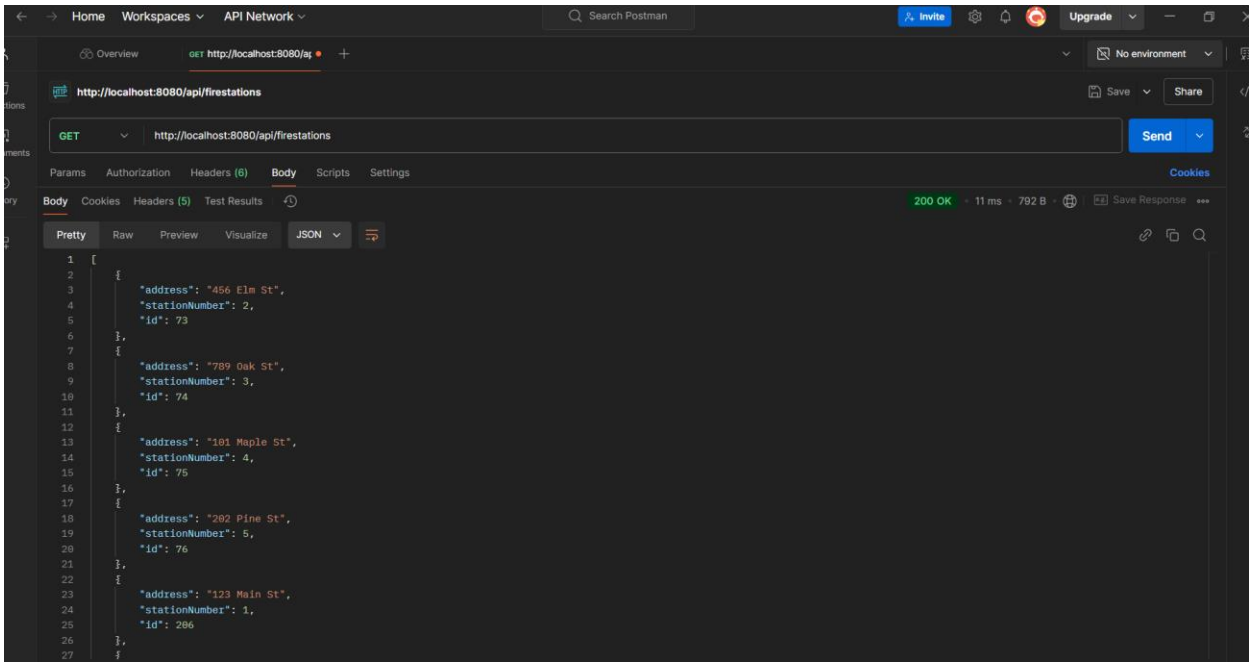
Test Results Summary for SafetyNet Alerts API

To verify that the application met the requirements and worked as expected, I tested the API endpoints using **Postman**. Here's a breakdown of the results for each API:

Get All Fire Stations

URL: http://localhost:8080/api/firestations

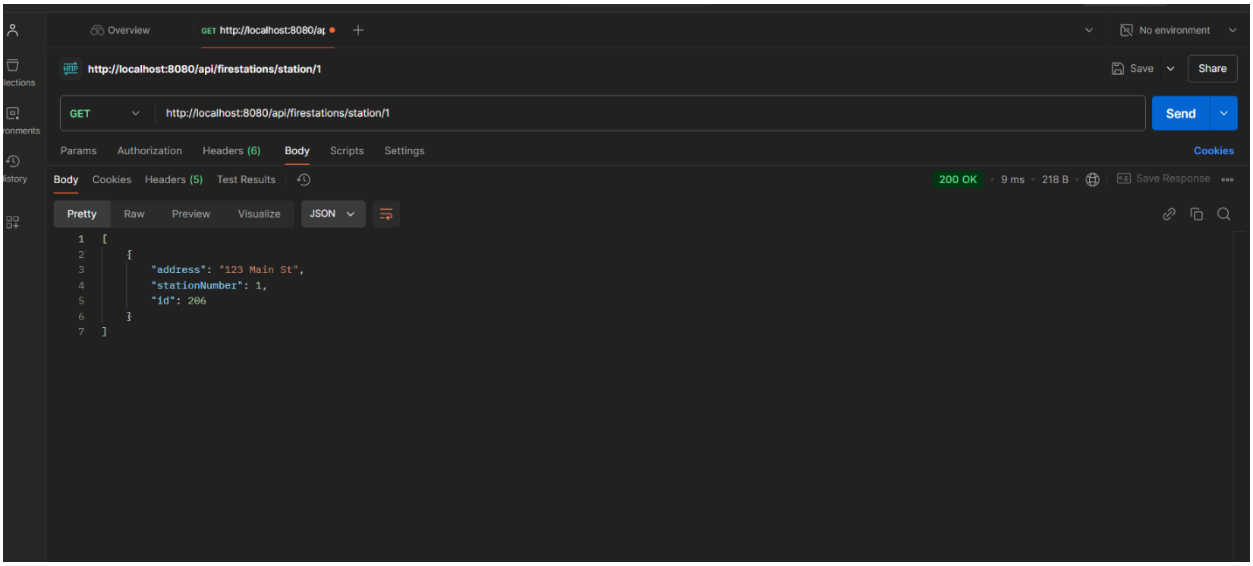
- Returns a full list of fire stations, including station numbers and addresses.



Get Fire Stations by Station Number

URL: http://localhost:8080/api/firestations/station/1

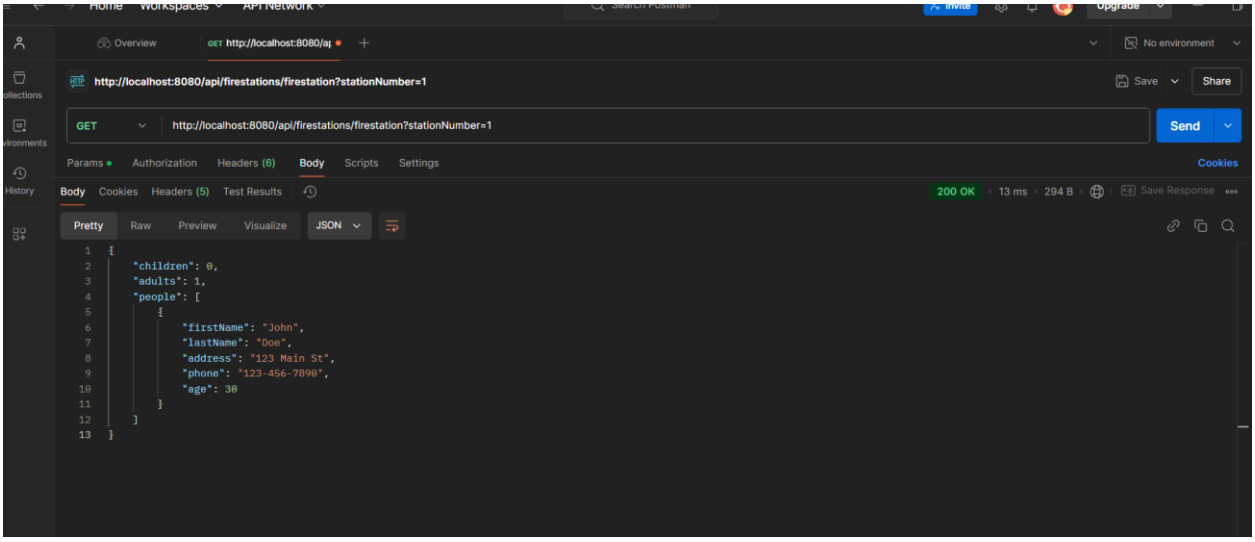
- Returns fire stations with station number 1, including their addresses.



Get People Served by Fire Station

URL: http://localhost:8080/api/firestations/firestation?stationNumber=1

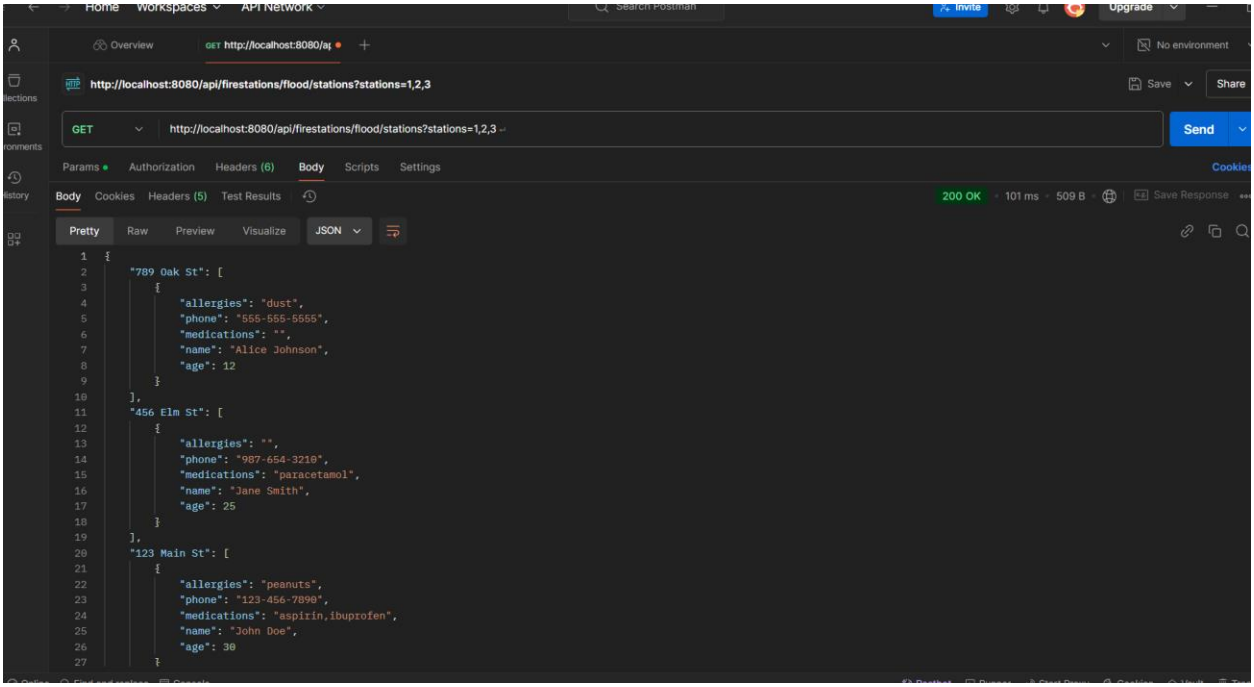
- Provides a list of people served by fire station 1, with their names, addresses, phone numbers, and summaries of adults and children



Get Flood Information by Station Numbers

URL: http://localhost:8080/api/flood/stations?stations=1,2,3

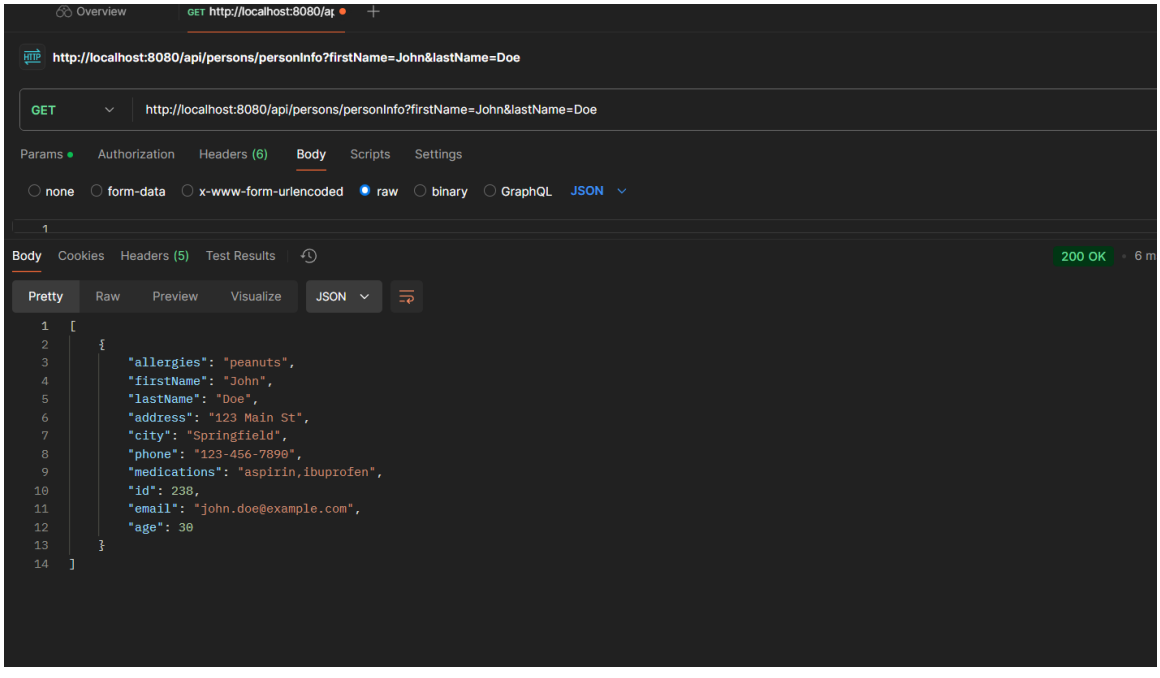
- Returns households for fire stations 1, 2, 3, grouped by address, including names, phone numbers, ages, and medical details



Get Person Information by Name

URL: http://localhost:8080/api/persons/personInfo?firstName=John&lastName=Doe

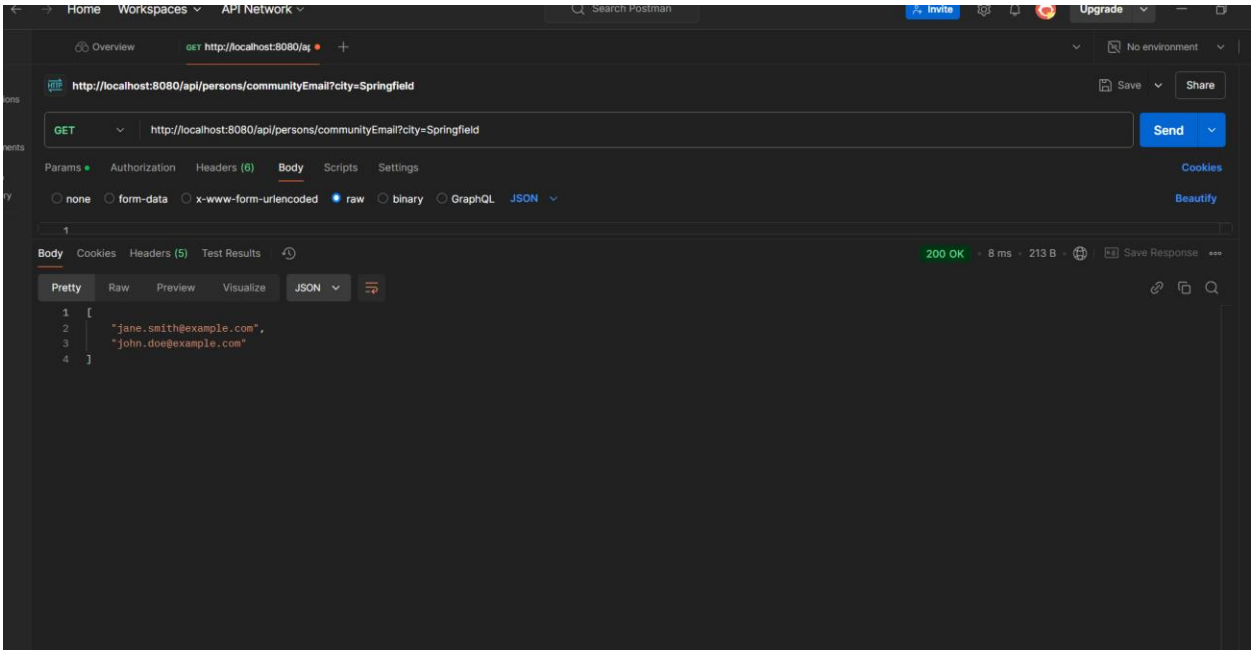
- Returns details for individuals named John Doe, including their addresses, ages, email addresses, and medical data.



Get Community Emails by City

URL: http://localhost:8080/api/persons/communityEmail?city=Springfield

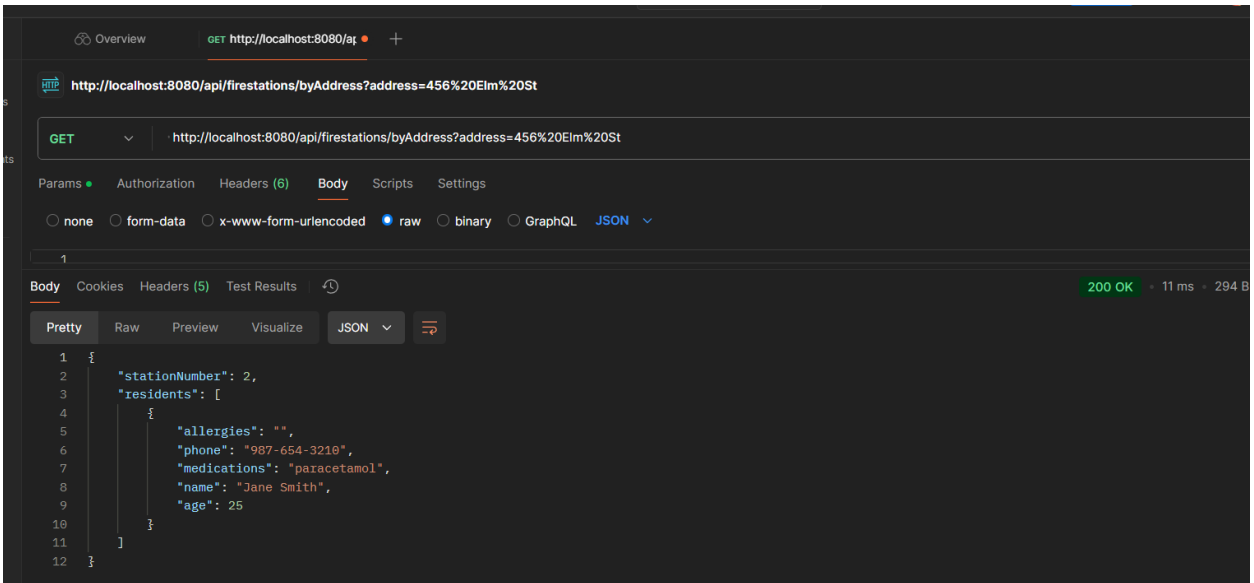
- Returns the email addresses for all people living in Springfield, useful for community-wide alerts.



Get Fire Station by Address

URL: http://localhost:8080/api/firestations/byAddress?address=456%20Elm%20St

- Returns the fire station serving the address 456 Elm st, with resident details, including medical information.



SOLID Table

SOLID Principle	Description	Implementation in Project
S - Single Responsibility Principle	A class should have one reason to change, meaning it should only have one job.	In the PersonService class, I separated concerns by having methods that only deal with specific tasks like fetching, saving, or updating person data. For example, getPersonById focuses only on fetching a specific person, while savePerson focuses only on saving data to the database.
O - Open/Closed Principle	Software entities should be open for extension but closed for modification.	I used interfaces and inheritance to allow the system to extend without modifying existing code. For instance, the FireStationService and PersonService are designed to easily extend their functionality without changing the core logic.
L - Liskov Substitution Principle	Objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program.	The inheritance structure allows for flexible modifications. For instance, if we had different types of fire stations or persons in the future, we could easily extend the existing classes without breaking the application.
I - Interface Segregation Principle	Clients should not be forced to implement interfaces they do not use.	Instead of creating a large, generic interface, I broke down the interfaces into smaller ones that serve specific functionalities. For example, the fire station repository and person repository have specific methods for managing only the relevant data.
D - Dependency Inversion Principle	High-level modules should not depend on low-level modules. Both should depend on abstractions.	I used dependency injection throughout the services by relying on Spring's @Autowired annotation to inject the necessary repositories. This way, the services do not depend on specific implementations but instead on the abstractions provided by the repository interfaces.

	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	