Marc Remillard

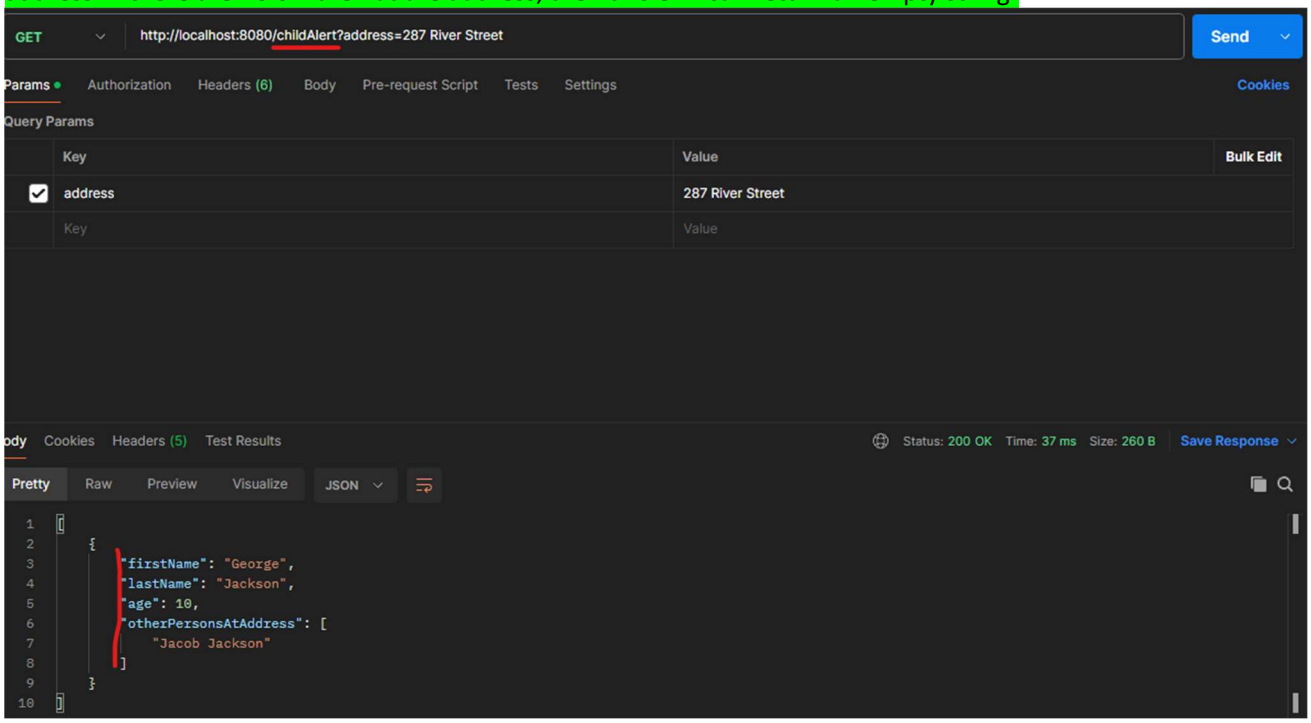
Final Project: Alerts Notification System

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).

1) 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.



```java
@GetMapping("/childAlert")
public List<ChildAlertDTO> getChildrenByAddress(@RequestParam String address) {
    // Find all persons by address
    List<Person> personsAtAddress = personRepository.findByAddress_AddressLine(address);  // Adjust according to

    // Filter children (age < 18)
    List<Person> childrenAtAddress = personsAtAddress.stream()
            .filter(person -> person.getAge() < 18)
            .collect(Collectors.toList());

    // If no children are found, return an empty list or empty string
    if (childrenAtAddress.isEmpty()) {
        return List.of();  // Returning an empty list if no children are found
    }

    // For each child, get the other persons living at the same address
    return childrenAtAddress.stream()
            .map(child -> {
                // Get the list of other persons at the address
                List<String> otherPersons = personsAtAddress.stream()
                        .filter(person -> !person.getId().equals(child.getId()))  // Exclude the current child
                        .map(person -> person.getFirstName() + " " + person.getLastName())
                        .collect(Collectors.toList());

                // Create a DTO for each child
                return new ChildAlertDTO(
                        child.getFirstName(),
                        child.getLastName(),
                        child.getAge(),
                        otherPersons
                );
            })
            .collect(Collectors.toList());
}
```

2) 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.

GET http://localhost:8080/phoneAlert?stationNumber=1 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
stationNumber	1	
Key	Value	

Body Cookies Headers (5) Test Results Status: 200 OK Time: 25 ms Size: 255 B Save Response

Pretty Raw Preview Visualize JSON

```
[
    "613-768-2841",
    "902-458-9779",
    "519-898-9082",
    "519-600-0508",
    "519-898-9082",
    "519-626-1456"
]
```

```
@GetMapping("/phoneAlert")
public List<String> getPhoneNumbersByFireStation(@RequestParam int stationNumber) {
    // Fetch persons associated with the given station number
    List<Person> people = personRepository.findByFireStationStationNumber(stationNumber);

    // Extract phone numbers from the people list
    return people.stream()
            .map(Person::getPhone) // Assuming you have a `getPhone` method in the Person class
            .collect(Collectors.toList());
}
```

GET http://localhost:8080/phoneAlert?stationNumber=8

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value
stationNumber	8
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
[]
```

NONE

3) 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.

GET http://localhost:8080/personInfo?firstName=Edna&lastName=Taylor

Params • Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value
☑ firstName	Edna
☑ lastName	Taylor
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "firstName": "Edna",
        "lastName": "Taylor",
        "email": "edna.taylor@outlook.com",
        "address": "7 King Street",
        "medicationsAndDosages": "Arthritis Medication - 1 pill per day",
        "age": 1,
        "allergies": "Nuts"
    }
]
```


```java
@GetMapping("/personInfo")
public List<PersonInfoDTO> getPersonInfo(@RequestParam String firstName, @RequestParam String lastName) {
    // Fetch persons by first and last name
    List<Person> persons = personRepository.findByFirstNameAndLastName(firstName, lastName);

    // Log the persons and their medications (for debugging purposes)
    persons.forEach(person -> {
        System.out.println("Person: " + person.getFirstName() + " " + person.getLastName());
        if (person.getMedications() != null && !person.getMedications().isEmpty()) {
            String[] medications = person.getMedications().split( regex: ",");
            String[] dosages = person.getDosages().split( regex: ",");
            for (int i = 0; i < medications.length; i++) {
                System.out.println("Medication: " + medications[i] + " - " + dosages[i]);
            }
        } else {
            System.out.println("No medications found for this person.");
        }
    });

    // Map the person data to PersonInfoDTO, including age and allergies
    return persons.stream() Stream<Person>
            .map(person -> new PersonInfoDTO(
                    person.getFirstName(),
                    person.getLastName(),
                    person.getEmail(),
                    person.getAddress() != null ? person.getAddress().getAddressLine() : "",  // Check if address
                    person.getMedications() != null && !person.getMedications().isEmpty() ?
                            person.getMedications() + " - " + person.getDosages() :
                            "No medications found",
                    person.getAge(),  // Add age to the DTO
                    person.getAllergies()  // Add allergies to the DTO
            )) Stream<PersonInfoDTO>
            .collect(Collectors.toList());
}
```

4) http://localhost:8080/communityEmail?city=<city>

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

GET http://localhost:8080/communityEmail?city=Halifax

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

Key	Value
city	Halifax
Key	Value

Body Cookies Headers (5) Test Results

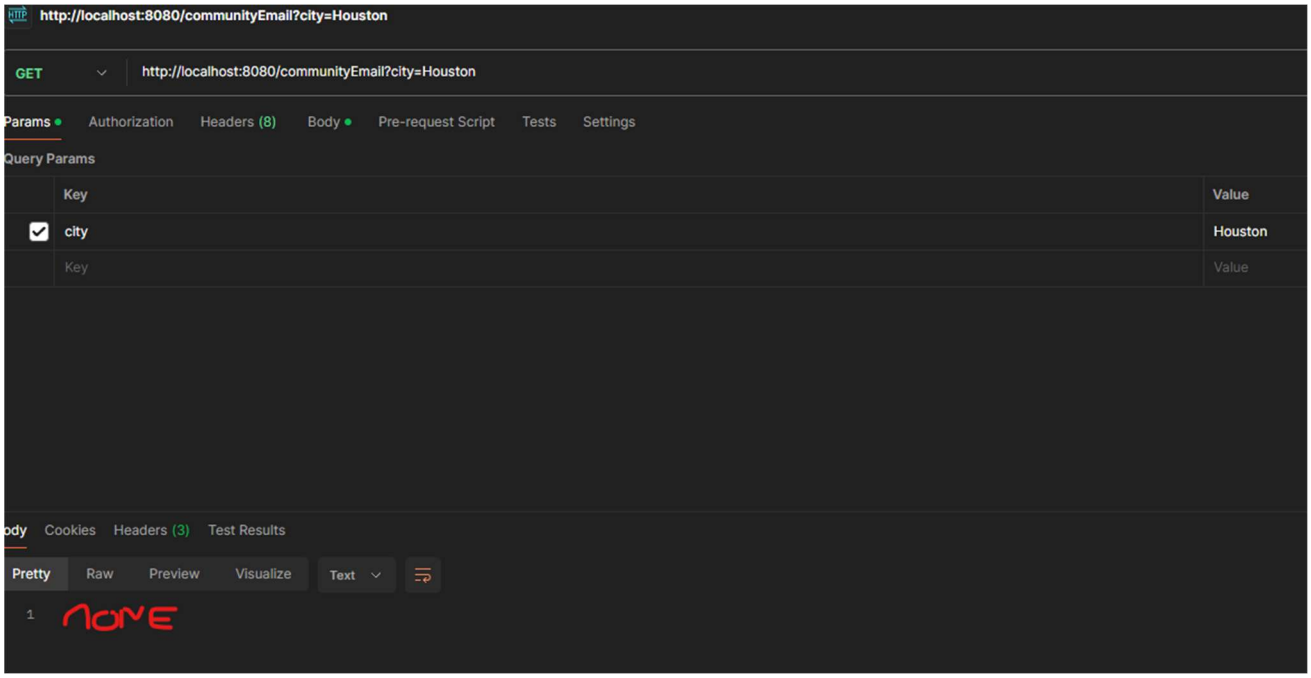
Pretty Raw Preview Visualize JSON


```
[
    "bob.doe@outlook.com",
    "emily.miller@outlook.com",
    "michael.jones@outlook.com",
    "george.wilson@outlook.com",
    "david.johnson@outlook.com",
    "emily.johnson@outlook.com",
    "bob.smith@outlook.com",
    "george.kim@outlook.com",
    "george.kim@outlook.com",
    "john.wilson@outlook.com",
    "william.miller@outlook.com",
    "william.miller@outlook.com",
    "sarah.taylor@outlook.com",
    "sarah.taylor@outlook.com",
    "david.miller@outlook.com",
    "sophie.anderson@outlook.com"
]
```


```java
@GetMapping("/communityEmail")
public ResponseEntity<List<String>> getCommunityEmail(@RequestParam String city) {
    List<String> emails = personService.getEmailsByCity(city);

    if (emails.isEmpty()) {
        return ResponseEntity.noContent().build();
    }

    return ResponseEntity.ok(emails);
}
```



SOLID Table

	Acronym	Concept	My Application of this concept
S	SRP	Single Responsibility principle	ChildAlertController - Only responsible for GetMapping of /childAlert CommunityEmailController - Only responsible for GetMapping of /community Email PersonInfoController - Only responsible for GetMapping of /personInfo PhoneAlertController - Only responsible for GetMapping of /phoneAlert
O	OCP	Open/closed principle	
L	LSP	Liskov substitution principle	
I	ISP	Interface segregation principle	Use of interfaces - Only methods which are needed are implemented
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**