Introduction

To develop a project of this size, we can use only ChatGPT to ask it how and what to do. We start by telling it, in general, what the given task is, then ask it to break it down into smaller tasks and give us a to-do list, then for each task it gives us, we ask it to break it into smaller tasks until we reach a point that the tasks it gives us are so simple that we can do it ourselves without the need for any help.
By the way, when I say “the tasks it gives us are so simple that we can do it ourselves without the need for any help”, I mean for real, just copying and pasting. Also, sometimes I copy the code or commands it gives me but I don’t know where to paste it, like for example, it wrote me a java file once and I didn’t where should I put it in the project so I asked it where to paste the code it gave me.

We don’t really need any tutorial, why would we need a tutorial if we are chatting with whom read all the good tutorials on the internet anyway? We can just ask it and it will give us the juice. In every following step, I explain in general what we should do in each step, for more details, please open the links that point to documentation or in-depth tutorials. These links might be helpful to you if you didn’t know how to chat with ChatGPT.

Requirements

For the client side, we need to install Node.js, which allows us to run JavaScript on our local machine, it also comes with NPM the JavaScript package manager, we can use it to install React.js and other packages later on.
For the server side, we need to install JDK, which will be used to run our Java server. We also need to install PostgreSQL the database we are going to use.

Setting Up The Client-Side

- Using your computer’s command line tool, create React application using create-react-app, the created folder will contain all the related client-side files
- Navigate to the created folder using your command line tool, then write “npm i axios bootstrap react-router-dom xlsx”. This command will install all the packages we need for the client application
    - Axios - used to send requests easily to any server
    - Bootstrap - for styling the user interface without the need to write CSS
    - React-router-dom - manages the application’s pages and navigation
    - Xlsx - helps us export data in xlsx format (Microsoft Excel)

Setting Up The Server-Side

Using your favorite IDE (I used VSCode), create a spring-boost project with these three essential dependencies:

- Spring-boot-starter-web
- Postgresql
- Spring-boot-starter-data-jpa

You can watch this tutorial if you don't completely understand how to add dependencies to your spring-boost project.
After setting up the project, find and open the porm.xml file and make sure that it contains all three dependencies. Here is an example:

pom.xml

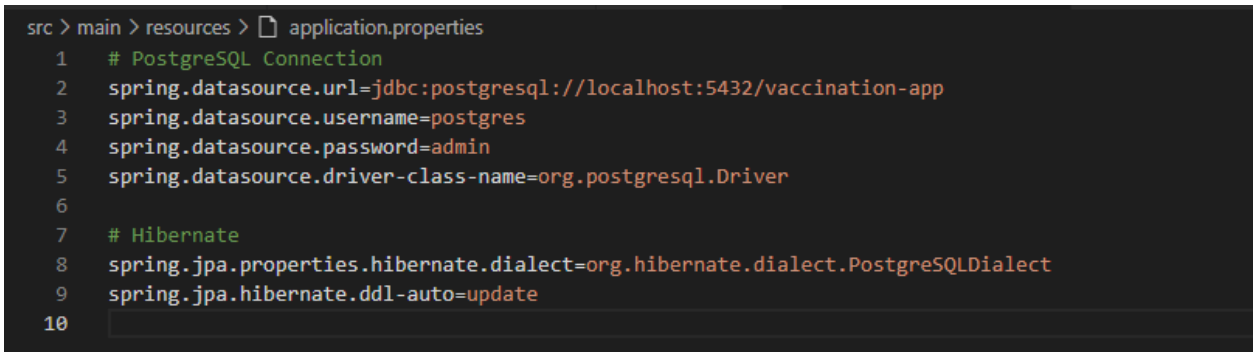
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.1.0</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.vaccination</groupId>
    <artifactId>api</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>api</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>20</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Setting Up The Database

Assuming that you successfully installed PostgreSQL on your machine, you can now search for “pgadmin” on your computer and open it. This software is a friendly user interface which helps us create and manage our PostgreSQL database easily. In case you find the user interface complicated, here is a quick guide on how to create a database using pgadmin.

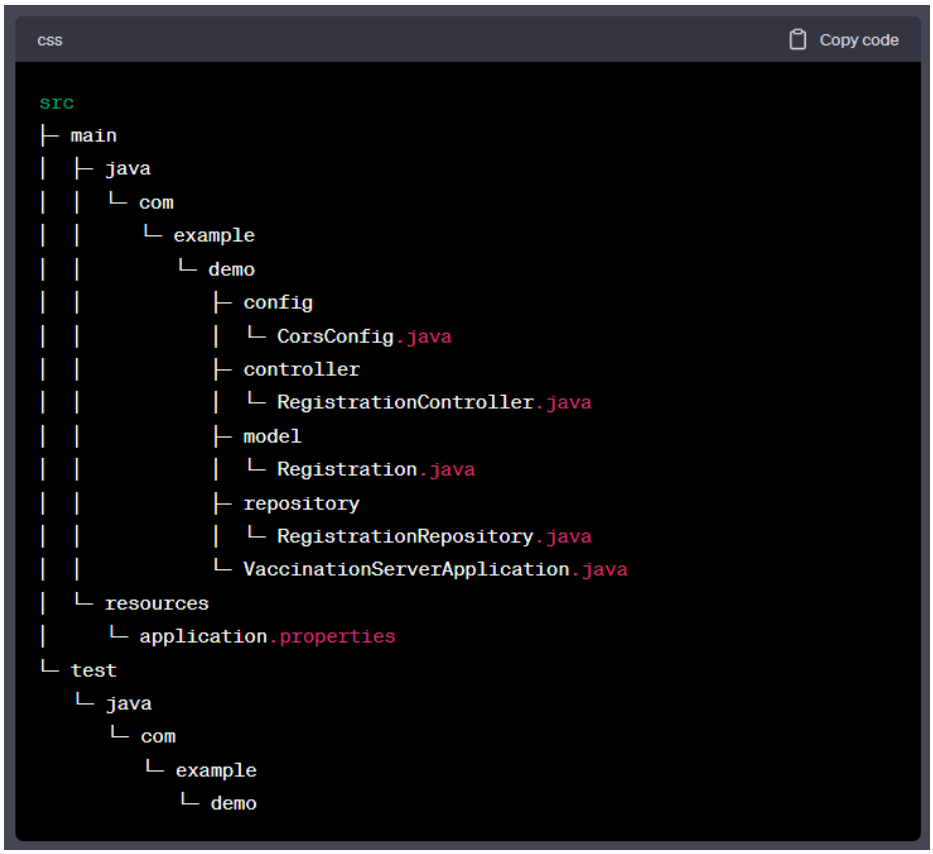
Connecting The Server-Side To The Database

Find and open the application.properties file in your server-side project folder, and make sure that it looks like the image below. Don’t forget to change the values and the keys according to your database name, postgres user name, and password.



Creating Java Classes

Now we want to create the Java files needed on the server side. In total, we need 5 Java files each one has one class. The text below represents the structure of our project, this text is generated by ChatGPT by the way.



Honestly, I didn't write any line of code, you can watch whatever YouTube tutorial if you really need that but for me I just asked ChatGPT to write for me the code. So I want to give some examples of how to ask ChatGPT to write the logic, see the following page.

Aha okay, I want to remind you that this what the client app needs:

1- Registration Page: Citizens will enter their information in a form as follows:
a- First Name
b- Last Name
c- Date of birth
d- Address
e- City (a dropdown)
f- Zip code (optional)
g- Land line
h- Cellular phone
i- Checkbox whether they have been infected by COVID-19 before
j- multi-check options for previous conditions like (Diabetes, Cardio-Vascular problems, Allergies..etc) with a text box for the user to enter "Other" conditions

2- Summary Page: A tabular view that shows all the existing registration info available in the system. With a search grid to search by a)Date of birth (date range) b) City

Thank you for the reminder. Now that we have set up the server-side components, let's continue with the implementation of the API endpoints to handle the client's requests.

1. Implement the Registration API Endpoint:
   Create a new controller class, `RegistrationController.java`, in the `com.example.demo.controller` package. This controller will handle the registration-related requests. Here's an example of how the code might look:

java Copy code

```java
package com.example.demo.controller;

import com.example.demo.model.Registration;
import com.example.demo.repository.RegistrationRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
```

Regenerate response

Send a message

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version

```java
import java.util.List;

@RestController
@RequestMapping("/api/registrations")
public class RegistrationController {

    private final RegistrationRepository registrationRepository;

    @Autowired
    public RegistrationController(RegistrationRepository registrationReposit
        this.registrationRepository = registrationRepository;
    }

    @PostMapping
    public Registration createRegistration(@RequestBody Registration registr
        return registrationRepository.save(registration);
    }

    @GetMapping
    public List<Registration> getAllRegistrations() {
        return registrationRepository.findAll();
    }
}
```

In this example, the `RegistrationController` class handles the POST request for creating a new registration and the GET request for retrieving all registrations. The `@PostMapping` annotation is used for the createRegistration() method, and the `@GetMapping` annotation is used for the getAllRegistrations() method.

Make sure to adjust the package structure and imports accordingly.

2. Configure CORS (Cross-Origin Resource Sharing):
   To allow requests from your React client application to the Spring server, you need to configure CORS. Add the following configuration class, `CorsConfig.java`, in the `com.example.demo.config` package:

Regenerate response

Send a message

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version

2. Configure CORS (Cross-Origin Resource Sharing):
   To allow requests from your React client application to the Spring server, you need to configure CORS. Add the following configuration class, `CorsConfig.java`, in the `com.example.demo.config` package:

java Copy code

```java
package com.example.demo.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;
import org.springframework.web.filter.CorsFilter;

@Configuration
public class CorsConfig {

    @Bean
    public CorsFilter corsFilter() {
        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurati
        CorsConfiguration config = new CorsConfiguration();
        config.addAllowedOrigin("*");
        config.addAllowedHeader("*");
        config.addAllowedMethod("*");
        source.registerCorsConfiguration("/**", config);
        return new CorsFilter(source);
    }
}
```

This configuration allows requests from any origin (`*`), with any headers and methods.

Make sure to adjust the package structure and imports accordingly.

With these changes, the server-side is ready to handle the registration requests from the client application.

Regenerate response

Send a message

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version

With these changes, the server-side is ready to handle the registration requests from the client application.

What about the RegistrationRepository you mentioned earlier? should we add things to it or not

Apologies for the oversight. Yes, we need to update the `RegistrationRepository` to include additional query methods for searching registrations based on date of birth and city. Here's an updated version of the `RegistrationRepository.java` file:

java Copy code

```java
package com.example.demo.repository;

import com.example.demo.model.Registration;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.time.LocalDate;
import java.util.List;

@Repository
public interface RegistrationRepository extends JpaRepository<Registration,
    List<Registration> findByDateOfBirthBetween(LocalDate startDate, LocalDa

    List<Registration> findByCity(String city);
}
```

In this example, we added two methods to the `RegistrationRepository` interface:

- `findByDateOfBirthBetween`: This method retrieves registrations with a date of birth between the given start date and end date.
- `findByCity`: This method re

Regenerate response

a specific city.

Send a message

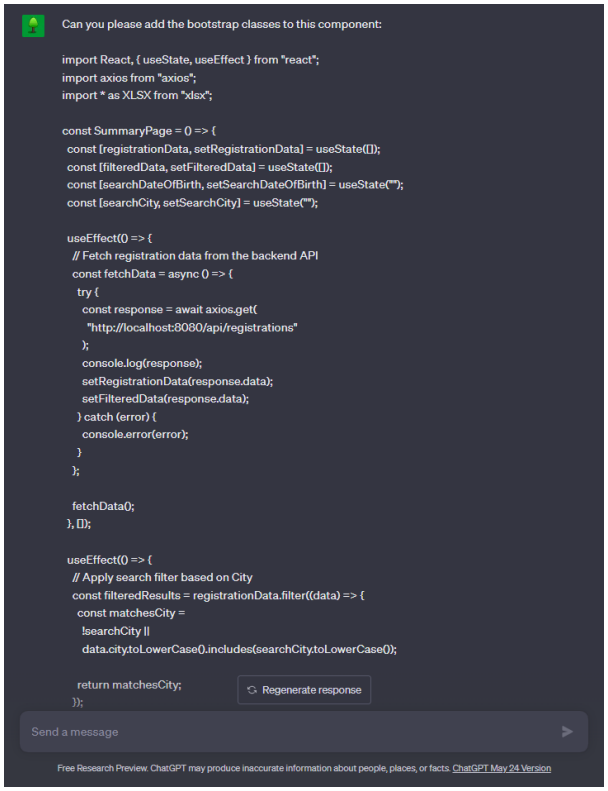
Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version

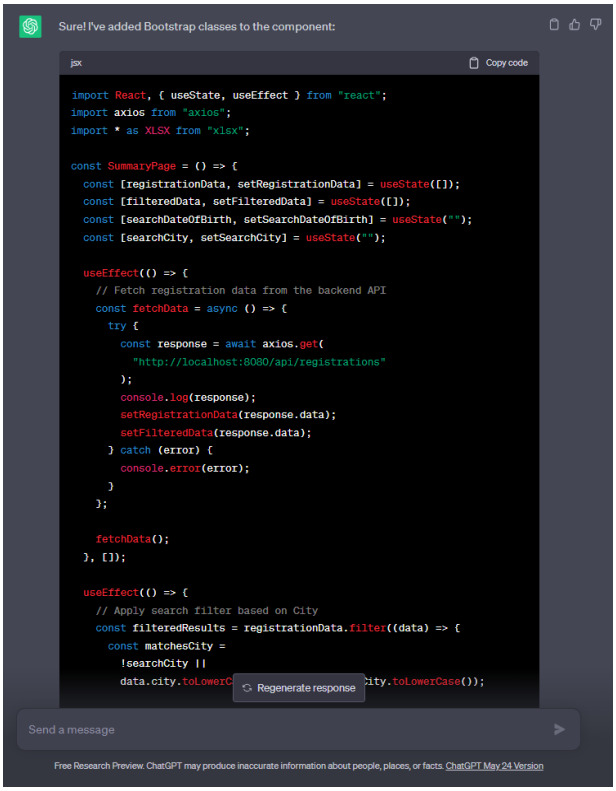
Creating The Client-Side Components

Here also I used ChatGPT to write all the code, I simply asked it to do things for me after explaining to it what I need. See this video I uploaded to YouTube if you are interested in knowing how to prompt ChatGPT efficiently.

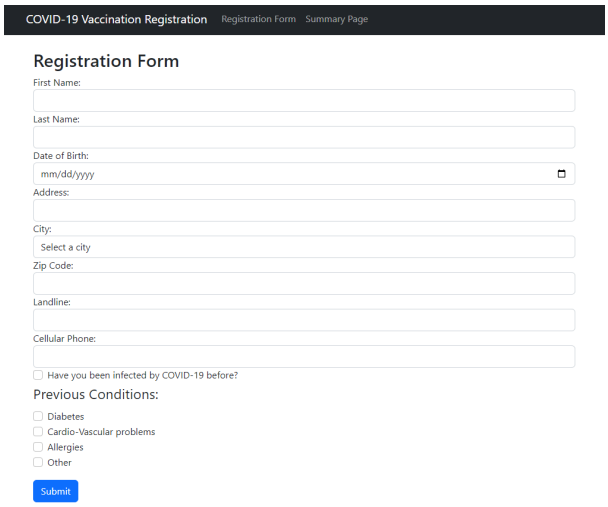
Now the client side looks something like this:

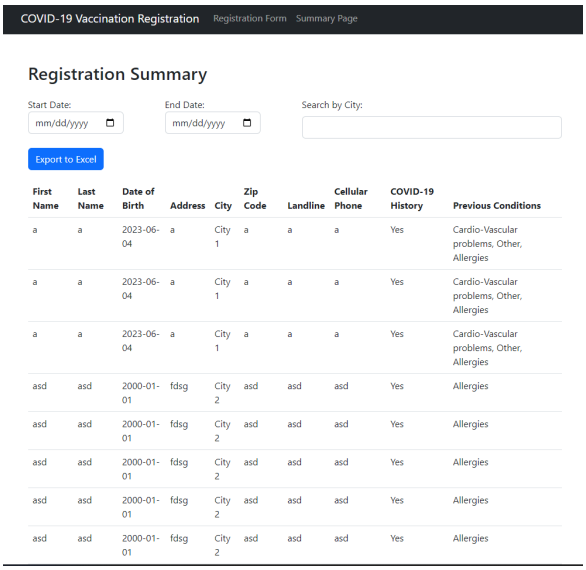
This is complete garbage, ley's copy-paste each component we have to ChatGPT and ask it to style it using Bootstrap:





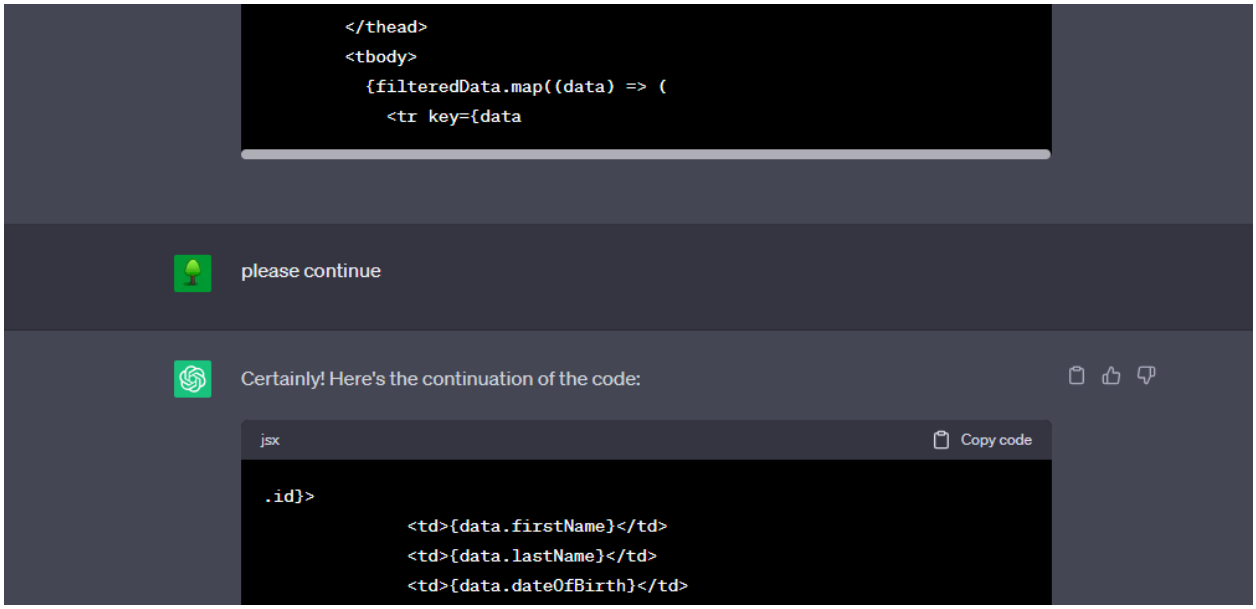
Now It will look something like this:





Tips On How To Chat With ChatGPT

- Sometimes when it is generating code or any long text, it suddenly stops generating code, don't worry this is normal, you can simply say "Please continue" and it will continue writing the code



- Sometimes when you run the code after pasting the code it gave you an error appears. Don't worry, you can simply copy and paste the error you got to ChatGPT and ask it to resolve it for you

- Sometimes you notice that even though the code is compliable without any error, there are still bugs in the logic and functionality itself. Don’t worry, just describe the bug for ChatGPT and maybe try to describe the behavior you were expecting to ChatGPT and it will fix it for you
- Don’t forget to watch this video I upload to YouTube, and feel free to pause it any second if you found something interesting