

# תיעוד תהליך פיתוח "Covid Vaccination App"

קישור לפרויקט (GitHub): [Final Project](#)

**מגישים:**

אביגיל חגי 318201688

נדב מלול 206762973

## תוכן העניינים

3	..... תחילת עבודה עם ChatGPT ועזרה בבניית תוכנית עבודה
4	..... יישום תוכנית העבודה שניתנה על ידי ה-ChatGPT
4	..... שלב ראשון: הכנת מסד הנתונים
5	..... שלב שני: יצירת ה-Backend
7	..... שימוש ב-CoPilot
8	..... שלב שלישי: יצירת ה-Frontend
8	..... שימוש בפלאג DeployScript
9	..... יישום האפשרות לבחירת עיר באמצעות Dropdown
10	..... עיצוב האתר:
10	..... שימוש בפלאג SummarizeAnything
13	..... הוספת האפשרות של ייצוא הטבלה ל-Excel
15	..... הרצת הפרויקט:
15	..... לסיכום:

## תחילת עבודה עם ChatGPT ועזרה בבניית תוכנית עבודה

תחילה פנינו בבקשה ל-GPT על מנת שייתן לנו תוכנית עבודה מסודרת לבניית האפליקציה ע"פ שלבים.

תוכנית העבודה שהתקבלה היא:

### 1. **Setting Up Your Database (Postgres SQL):**

You'll need to install Postgres and set up your database schema. This will involve creating tables to store the registration information you collect from your users. This might include fields like firstName, lastName, dateOfBirth, address, city, zipCode, landLine, cellPhone, covidInfected, preexistingConditions, etc.

If you need to write SQL commands, you can ask me to generate them based on your requirements. For instance, to create a table, you could provide the table name and fields, and I could generate a `CREATE TABLE` command for you.

### 2. **Creating the Backend (Java):**

For Java, you might consider using a framework like Spring Boot, which simplifies setup and includes built-in tools for creating REST APIs and connecting to databases.

You'll need to write code to handle API requests from your frontend, interact with the database, and return appropriate responses. This will include operations like inserting new records, querying existing ones, etc.

If you're unsure how to write a certain piece of code, you can give me a description of what you want to do, and I could generate a Java code snippet for you.

### 3. **Creating the Frontend (React):**

For the frontend, you'll need to create your React app, which will involve setting up your project using Create React App, Next.js, or a similar tool.

The frontend will include the form for user registration and the summary page. The form should collect the required information from the user and make a POST request to your backend API to submit it. The summary page should make GET requests to your backend API to retrieve and display data.

If you're not sure how to write a particular React component or make an API request, you can describe what you want to do, and I could generate a React code snippet for you.

### 4. **Connecting Everything Together:**

Once you have your database, backend, and frontend set up, you'll need to connect everything together. This will involve configuring your backend to connect to your database and your frontend to make requests to your backend.

### 5. **Deployment:**

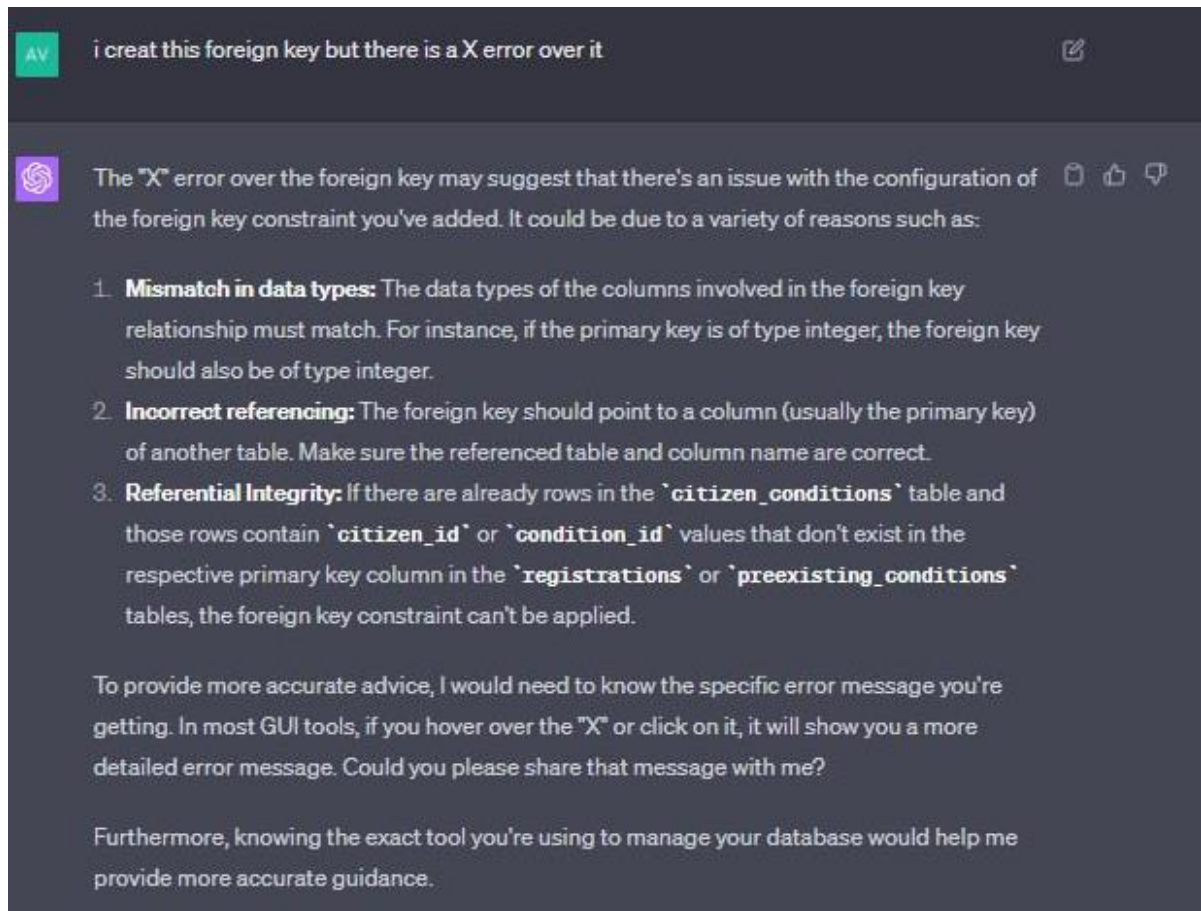
Finally, once you have your application working locally, you'll need to deploy it so that others can use it. This might involve deploying your backend and frontend to a server or cloud platform, and setting up your database on a cloud database service or a server.

## יישום תוכנית העבודה שניתנה על ידי ChatGPT

שלב ראשון: הכנת מסד הנתונים.

נעזרנו במדריך להתקנת PgAdmin ע"פ חוברת התקנה שקיבלנו בקורס "מסדי נתונים",  
[ניתן למצוא את המדריך כאן](#).

בנוסף כאשר הגדרנו את מסד הנתונים ותכולתו הייתה לנו בעיה בהגדרת Foreign key של טבלה מסוימת ובמובן שנעזרנו בChatGPT על מנת לפתור את בעיה זו. (זוהי דוגמא לדרכים בהם נעזרנו בGPT לטובת סיוע בתקלות ובאגים בתהליך כתיבת הקוד)



לבסוף בנינו טבלה במסד הנתונים עם השדות הבאים :

Create - Table

General

Columns

Advanced

Constraints

Partitions

Parameters

Security

SQL

Columns

		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
		id	serial			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		first_name	text			<input type="checkbox"/>	<input type="checkbox"/>	
		last_name	text			<input type="checkbox"/>	<input type="checkbox"/>	
		date_of_birth	date			<input type="checkbox"/>	<input type="checkbox"/>	
		address	text			<input type="checkbox"/>	<input type="checkbox"/>	
		city	text			<input type="checkbox"/>	<input type="checkbox"/>	
		zip_code	bigint			<input type="checkbox"/>	<input type="checkbox"/>	
		land_line	text			<input type="checkbox"/>	<input type="checkbox"/>	
		cell_phone	bigint			<input type="checkbox"/>	<input type="checkbox"/>	
		covid_infected	boolean			<input type="checkbox"/>	<input type="checkbox"/>	

## שלב שני: יצירת הBackend.

נעזרנו ב ChatGPT שהנחה אותו ליצירת התשתית, ע"פ השלבים הבאים:

### Step 2: Creating the Backend (Java):

We'll use Spring Boot for this, as it simplifies the process significantly.

- **Set Up a Spring Boot Project:**

Start by creating a new Spring Boot project. You can do this through the [Spring Initializer](#) web interface, selecting the options you need. At the very least, you should select the following dependencies: Spring Web, Spring Data JPA, and PostgreSQL Driver. After generating the project, you can import it into your favorite Java IDE.

- **Configure Database Connection:**

Once your project is set up, you need to configure Spring to connect to your database. In your **application.properties** file (or **application.yml** if you're using YAML format), add the following settings:

### propertiesCopy code

spring.datasource.url=jdbc:postgresql://localhost:5432/mydatabase

spring.datasource.username=postgres

spring.datasource.password=yourpassword spring.jpa.hibernate.ddl-auto=update

Replace **mydatabase**, **postgres**, and **yourpassword** with your actual database name, username, and password.

- **Define the Entity:**

In your Java code, define a class to represent a Citizen. This class should be annotated with **@Entity** to tell Spring that it corresponds to a database table, and its fields should correspond to the columns of your table.

```
import javax.persistence.*;

@Entity
@Table(name = "citizens")
public class Citizen {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @Column(name = "first_name")
    private String firstName;

    // other fields...

    // Getters and setters...
}
```

- **Create a Repository:**

Now create an interface for your **Citizen** repository. This interface should extend **CrudRepository** or **JpaRepository**, which will give you methods for basic CRUD operations. You can also add custom query methods if needed.

### javaCopy code

```
import org.springframework.data.jpa.repository.JpaRepository;
```

```
public interface CitizenRepository extends JpaRepository<Citizen, Long> { }
```

- **Create a Controller:**

Finally, create a **CitizenController** class to handle HTTP requests. This class should use your **CitizenRepository** to perform operations.

Here's a basic example of what your **CitizenController** might look like:

```
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/v1/citizens")
public class CitizenController {

    private final CitizenRepository citizenRepository;

    // Constructor injection
    public CitizenController(CitizenRepository citizenRepository) {
        this.citizenRepository = citizenRepository;
    }

    @GetMapping
    public List<Citizen> getCitizens() {
        return citizenRepository.findAll();
    }

    @PostMapping
    public Citizen addCitizen(@RequestBody Citizen citizen) {
        return citizenRepository.save(citizen);
    }

    // other methods...
}
```

In the above code, **getCitizens()** is used to fetch all the citizens from the database, and **addCitizen()** is used to save a new citizen to the database.

Proceed with these steps, and let me know if you encounter any issues or when you're ready for the next steps.

מסומנות בכחול השורות שמחברות את שרת הספרינג למסד הנתונים.

## שימוש ב-CoPilot

להשלמת שלב 2 התקנו על סביבת העבודה שלנו (IntelliJ), כלי AI ששמו **GitHub CoPilot** והינו כלי מבית OpenAI שמטרתו לעזור למפתחים לכתוב קוד. הוא מונע באמצעות מודלי למידת מכונה ומבוסס על גרסאות מותאמות אישית של מודל GPT-3.

ה-CoPilot מאפשר **השלמת קוד אוטומטית**: הכלי מציע השלמות שורת קוד או קטעי קוד שלמים בהתאם להקשר. למשל, אם אתה מתחיל לכתוב פונקציה, ה-CoPilot - ימשיך את מה שהתחלת, תוך התחשבות בנתונים שהוזנו בשלב זה.

למשל בדוגמא הנוכחית כתבנו בהערה getter and setter וה-CoPilot השלים לנו את הקוד אוטומטית:

```
// getters and setters
public long getId() { return id; }
```

כלי זה חסך לנו זמן רב בכתיבת הקוד.

## שלב שלישי: יצירת הFrontend.

תחילה התקנו ע"פ הוראות ההתקנה של ChatGPT את react באופן הבא:

**Step 3: Creating the Frontend (React)**

1. **Setting up your React project:** Start by setting up a new React application. If you have Node.js and npm installed, you can use the following command to create a new React application:

```
bash
```

Copy code

```
npx create-react-app covid-vaccination-app
```

## שימוש בפלאג DeployScript

התקנו את **פלאג DeployScript**, DeployScript הוא כלי שמאפשר ל-ChatGPT ליצור קוד HTML, CSS, ו-JavaScript לאפליקציות אינטרנט. הוא משמש כממשק בין ChatGPT לשירותים של פיתוח אפליקציות אינטרנט, ומאפשר ל-ChatGPT ליצור ולעדכן קבצים של אפליקציות אינטרנט באופן דינמי.

העתקנו את תיאור התרגיל ל-GPT וקיבלנו את 2 קבצי ה-HTML הנ"ל:

Used DeployScript

Used DeployScript

Used DeployScript

I have created the basic structure for the Registration and Summary pages using HTML. You can view them at the following URLs:

- [Registration Page](#)
- [Summary Page](#)

לאחר מכן ביקשנו מכלי זה גם עיצוב בסיסי לדפים הללו.



בשלב הבא, ביקשנו מ-ChatGPT שישתמש בדפים הללו על מנת לייצור לנו 2 קומפוננטות: RegistrationForm ו-SummaryPage.

אלו חלק מהוראות ChatGPT ליצירת הקומפוננטות: (בהמשך נצרף לינק לשיחה המלאה)

2. **Creating the Registration Form:** Once you have your React project set up, let's create a registration form. Here's a very simple example:

```
jsx
import React, { useState } from 'react';

const RegistrationForm = () => {
  const [formData, setFormData] = useState({
```

לאחר מכן יצרנו את תקשורת שרת API הפנימי מפורט 8080(צד שרת הספרינג) לפורט 3000(לקוח), כלומר – בקשות שנשלחו מפורט 3000 ישלחו לפורט 8080 באמצעות הוספת השורה הבאה ל-package-lock.json:

*"proxy": http://localhost: 8080*

בבקשות ה-API לשרת הפנימי נעזרנו בCopilot באמצעות הוספת תיעוד עם שמות משמעותיים, כשבצד שרת נעזרנו ב-GPT בהתבסס על בקשת הלקוח.

### יישום האפשרות לבחירת עיר באמצעות Dropdown

על מנת ליישם את אפשרות בחירת העיר מבין ערים שונות, נעזרנו בChatGPT וביצענו את השלבים הבאים:

1. Sign up for a free account at GeoNames: <https://www.geonames.org/login>
2. After signing in, go to the "Account" section and find your API key.
3. Modify your `RegistrationForm` component to fetch city options from the GeoNames API using the `axios` library:

```
jsx
import React, { useState, useEffect } from 'react';
import axios from 'axios';

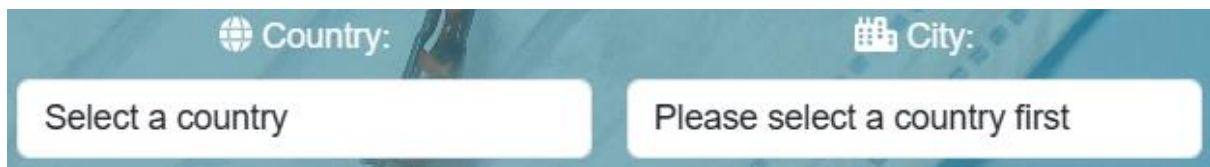
const RegistrationForm = () => {
  const [cityOptions, setCityOptions] = useState([]);

  useEffect(() => {
    const fetchCityOptions = async () => {
      try {
        const response = await axios.get(
          `http://api.geonames.org/searchJSON?username=YOUR_USERNAME&type=json`
        );
      }
    }
  });
```

לאחר מכן פנינו לדוקומנטציה של אתר [geonames](#) שם ראינו שישנה אפשרות לחפש ערים על פי מדינה מסוימת, ולכן בעזרת ChatGPT הוספנו שדה לבחירת מדינה, ובהתבסס על המדינה שאותו יבחר המשתמש הוא יקבל בעזרת בקשה מהשרת את הערים שבאותה המדינה. כלומר סה"כ ישנן 2 בקשות לשרת החיצוני והשנייה מתבססת על הראשונה.

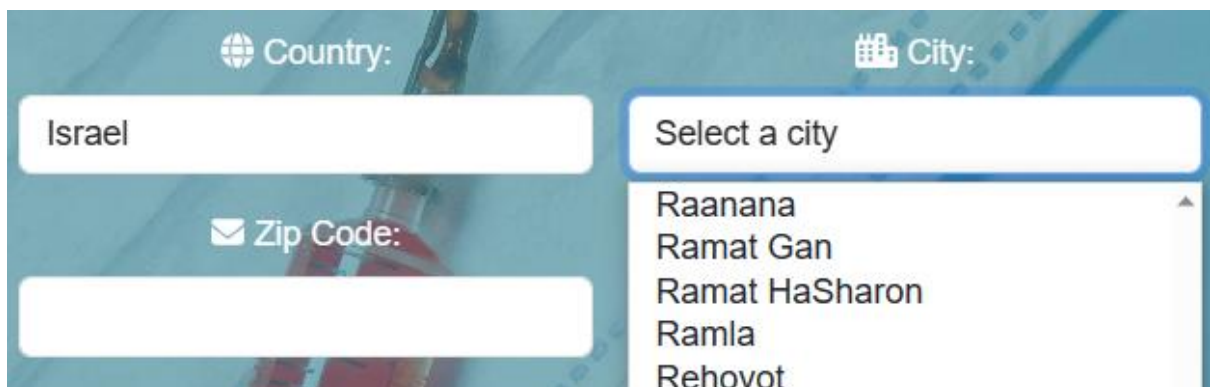
### המחשה

לפני בחירת ארץ:



The screenshot shows a web form with two input fields. The first field, labeled 'Country:' with a globe icon, contains the text 'Select a country'. The second field, labeled 'City:' with a city icon, contains the text 'Please select a country first'.

אחרי בחירת ארץ:



The screenshot shows the same web form after the country 'Israel' has been selected. The 'Country:' field now displays 'Israel'. The 'City:' field has a dropdown menu open, showing a list of cities: Raanana, Ramat Gan, Ramat HaSharon, Ramla, and Rehovot. Below the 'Country:' field, there is a new field labeled 'Zip Code:' with an envelope icon and an empty input box.

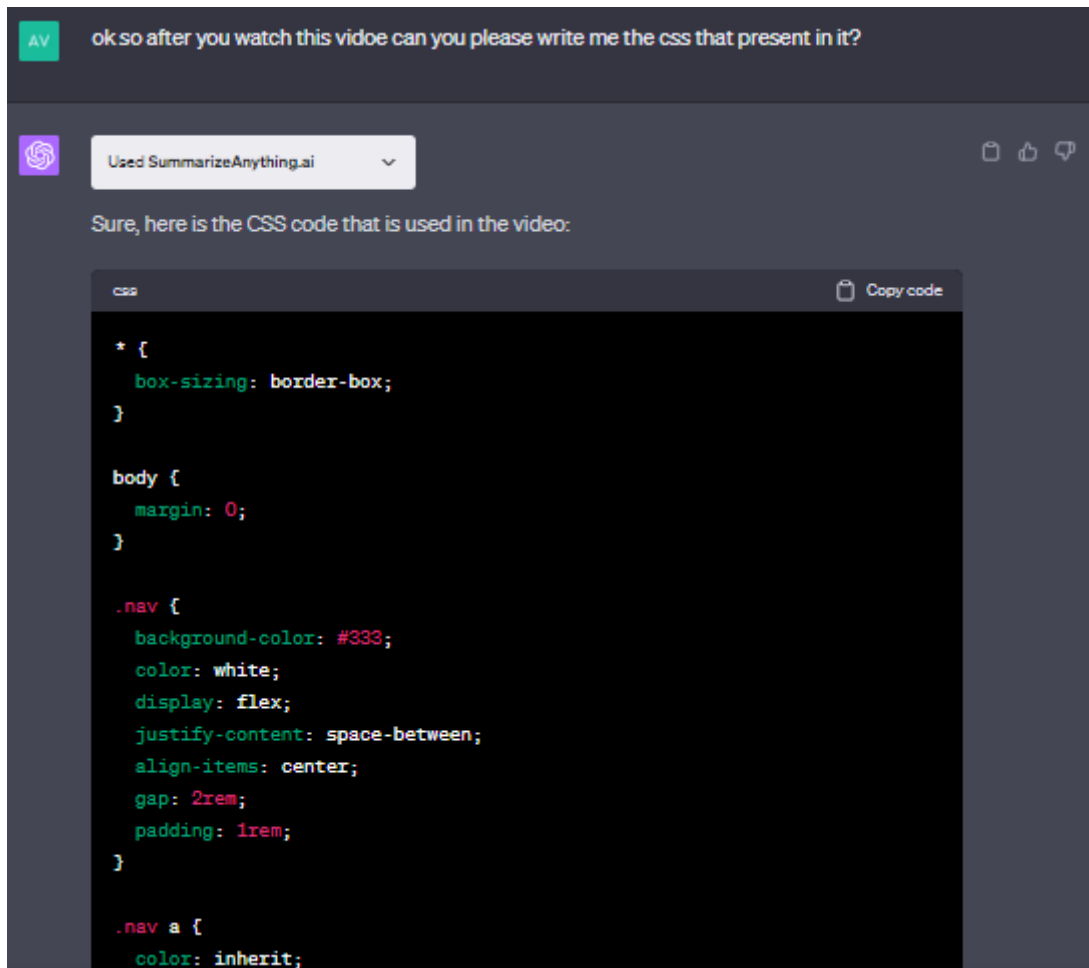
כמובן שאת הבקשות ואופן הטיפול בהן עשינו בעזרת ChatGPT. טיפול בשגיאות ביצענו באמצעות צד שרת על פי חומר שלמדנו בקורס "תכנות אינטרנט" על ידי הספריות Valid ו-BindingResult.

### **עיצוב האתר:**

#### **שימוש בפלאג SummarizeAnything**

לאחר יישום השלבים שניתנו על ידי ChatGPT כאשר האפליקציה כבר עבדה נותר עלינו לעצב אותה. לטובת כך התקנו את **SummarizeAnything**, הפלאג **SummarizeAnything**, הוא כלי שמאפשר לChatGPT לסכם תוכן ממגוון מקורות. הכלי מאפשר לטעון ולסכם תוכן מסרטוני YouTube.

למשל, בעיצוב navbar נעזרנו בסרטון [המופיע ביוטיוב](#), שאת הקישור שלנו שלחנו ל-GPT, ChatGPT באמצעות הפלאג הביא לנו את העיצוב המלא! (css) המופיע בסרטון.



The screenshot shows a ChatGPT chat window. At the top, a user message says: "ok so after you watch this vidoe can you please write me the css that present in it?". Below this, a system message indicates "Used SummarizeAnything.ai". The AI's response is: "Sure, here is the CSS code that is used in the video:". Below the text is a code block containing the following CSS code:

```
css

* {
  box-sizing: border-box;
}

body {
  margin: 0;
}

.nav {
  background-color: #333;
  color: white;
  display: flex;
  justify-content: space-between;
  align-items: center;
  gap: 2rem;
  padding: 1rem;
}

.nav a {
  color: inherit;
```

ה-Navbar שהתקבל:

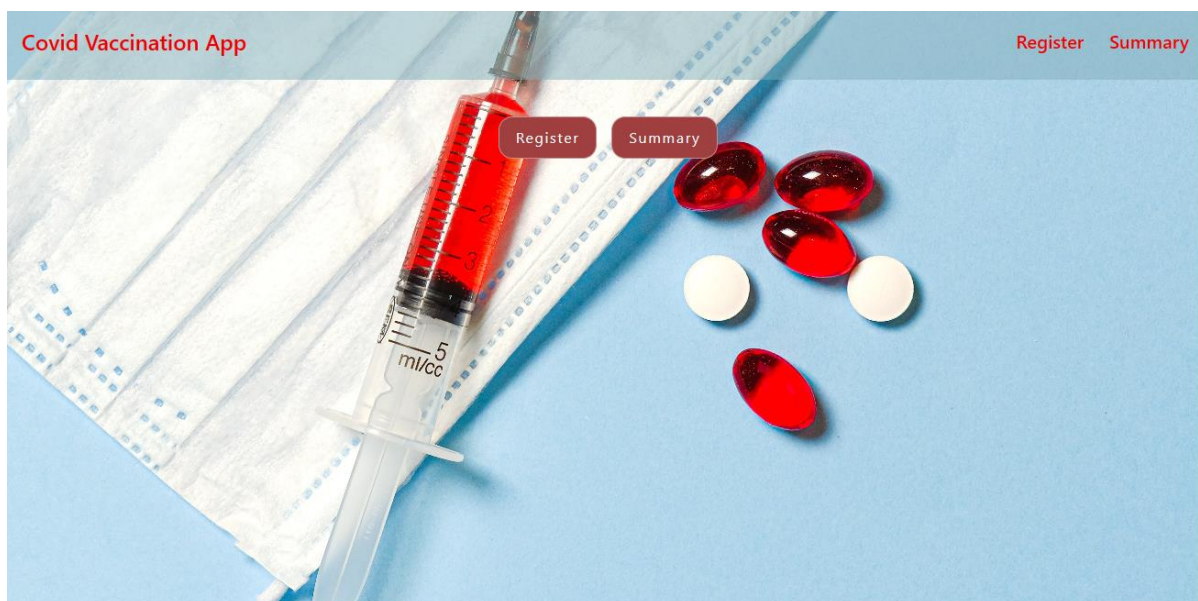


ה-Navbar הוא דוגמא אחת מתוך הדוגמאות הרבות שבהן השתמשנו בפלאג זה.

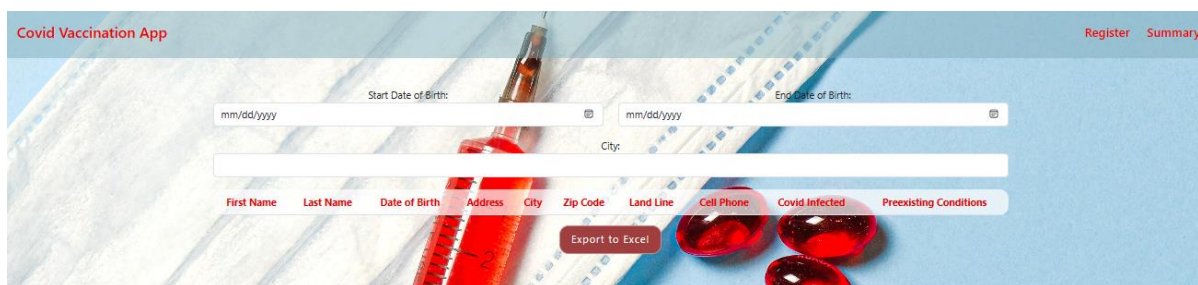
בנוסף נעזרנו בChatGPT על מנת להוסיף אייקונים לטופס ההרשמה (Registration form) והוספת תמונת רקע. כמו כן במטלת העיצוב נעזרנו במידע מקדים שהיה לנו מקורסי "תכנות אינטרנט א'", "תכנות אינטרנט ב'" שם למדנו כיצד לעצב אתר רספונסיבי באמצעות CSS, Bootstrap.

## נראות האתר

דף נחיתה:



דף הסיכום(בשהמסד ריק):



דף ההרשמה:

**Registration**

First Name:  Last Name:

Date of Birth:

Address:

Country:  City:

Zip Code:  Land Line:

Cell Phone:

☐ Covid Infected:

☐ Preexisting Conditions:

- Diabetes
- Cardio-Vascular problems
- Allergies
- Asthma
- Cancer
- HIV
- Other

## הוספת האפשרות של ייצוא הטבלה לExcel


על מנת ליישם את הבונוס היה עלינו למצוא תיעוד על האפשרויות בהן ניתן לייצא מReact לקובץ Excel, נעזרנו בChatGPT שהוסיף ביעילות את הייצוא הנ"ל לקומפוננטה הרלוונטית. את השלבים שניתנו על ידי ChatGPT ניתן לראות כאן:



### Step 1: Install the package

First, you need to install the `react-html-table-to-excel` package. Open your terminal and go to your project's root directory and run the following command:


bash

 Copy code

```
npm install react-html-table-to-excel --save
```

or

bash


 Copy code

```
yarn add react-html-table-to-excel
```

### Step 2: Import the package

Once the package is installed, you need to import it into the component file where the table is located:

javascript

 Copy code

```
import ReactHTMLTableToExcel from 'react-html-table-to-excel';
```


### Step 4: Create the Export to Excel button

Next, use the `ReactHTMLTableToExcel` component to create an "Export to Excel" button.

You need to pass a few properties:

- `id`: Any unique ID for the component.
- `className`: Any CSS class name for styling the button.
- `table`: The ID of the table we want to convert to Excel. It should match the ID we gave to our HTML table.
- `filename`: The name of the Excel file after it's downloaded.
- `sheet`: The name of the sheet inside the Excel file.
- `buttonText`: The text that will appear on the button.

javascript

 Copy code

```
<ReactHTMLTableToExcel  
  id="test-table-xls-button"  
  className="download-table-xls-button"  
  table="table-to-xls"  
  filename="tablexls"  
  sheet="tablexls"  
  buttonText="Download as XLS"/>
```

## הרצת הפרויקט:

על מנת להריץ את הפרויקט יש לבצע את השלבים הבאים:

### חיבור האפליקציה למסד הנתונים

1. צור טבלה בשם "citizens" במסד הנתונים של PostgreSQL.
2. הרץ את שרת ה-PostgreSQL על פורט 5432.

### הרצת צד לקוח

כתוב את הפקודות הבאות בטרמינל:

1. `cd covid-vaccination-app`
2. `npm start`

האפליקציה תרוץ בעת בפורט 3000 😊.

## לסיכום:

בתרגיל זה התבקשנו להשתמש בטכנולוגיות AI על מנת להשיג את המטרה, זאת בכדי להמחיש את האופנים הרבים בהם מתכנתים יכולים לנצל את טכנולוגיות אלו על מנת לייצר אפליקציה בדרך קלה ויעילה. החל מקבלת תכנית עבודה ועד להתקנת פלאגים נוספים כגון `Deploy Script` ו `Summarize Anything`.

כמו כן בנוסף לעזרה שקיבלנו מ-ChatGPT נעזרנו ב-Co-pilot הנמצא בסביבת העבודה IntelliJ כפי שכבר תיארנו קודם לכן.

מיותר לציין כי התייעוד ארוך מאוד ולכן ביצענו את כל הפרומפט באותו צ'אט עם ChatGPT על מנת שיזכור את השיחה הנ"ל וידע להתייחס לפרנסים שקרו קודם לכן. את הקישור לשיחה המלאה ניתן למצוא [באן](#).