Istanbul Technical University
Faculty of Computer and Informatics
Computer Engineering Department

BLG 317E
Database Project Report

Group 18
Talha Şahin - 150190114
Cemalettin Celal Toy - 150190091
Zehra Asan - 150190008
Hilal Erdoğan - 150190093
Mert Arabacı - 150190084

December 26nd, 2022

Contents

Chapter 1

Introduction

Our app is a website where users can view information about Covid in different countries and dates. Also, data can be added or changed by admins. In order to use the website users need to sign up first as a normal user or admin.

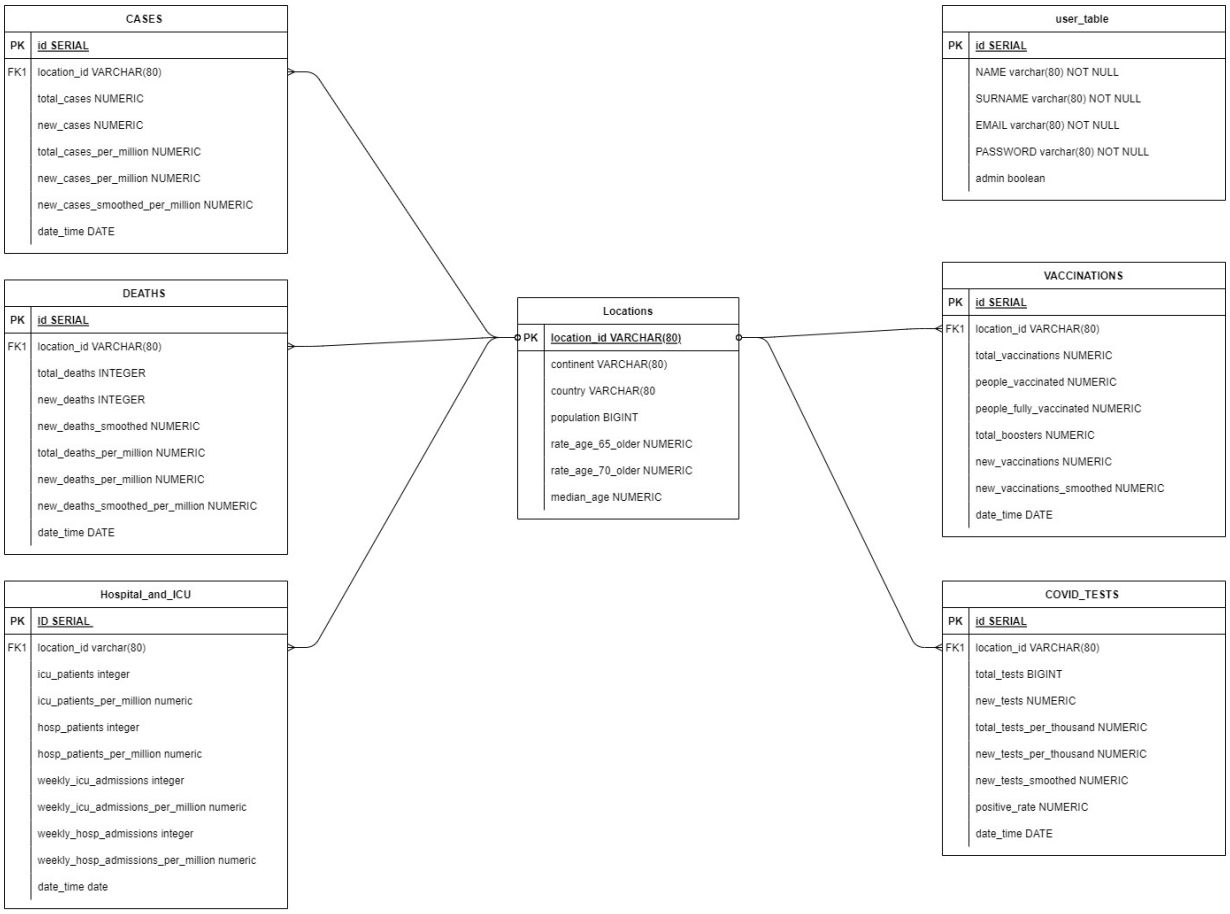


Figure 1.1: ER Diagram

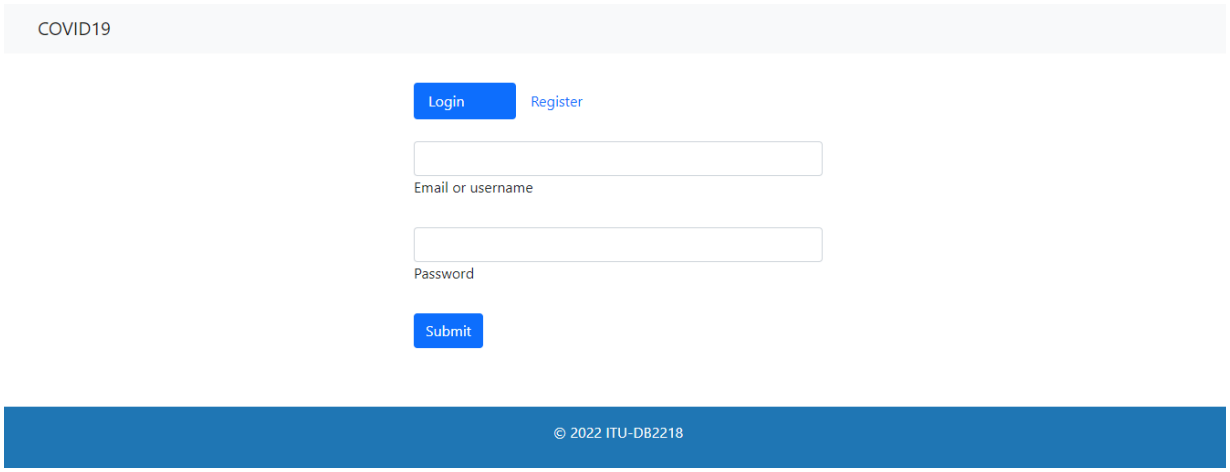
Chapter 2

User Guide

2.1 Login/Register - Implemented by Zehra Asan, Cemalettin Celal Toy and Mert Arabacı

2.1.1 Login Part

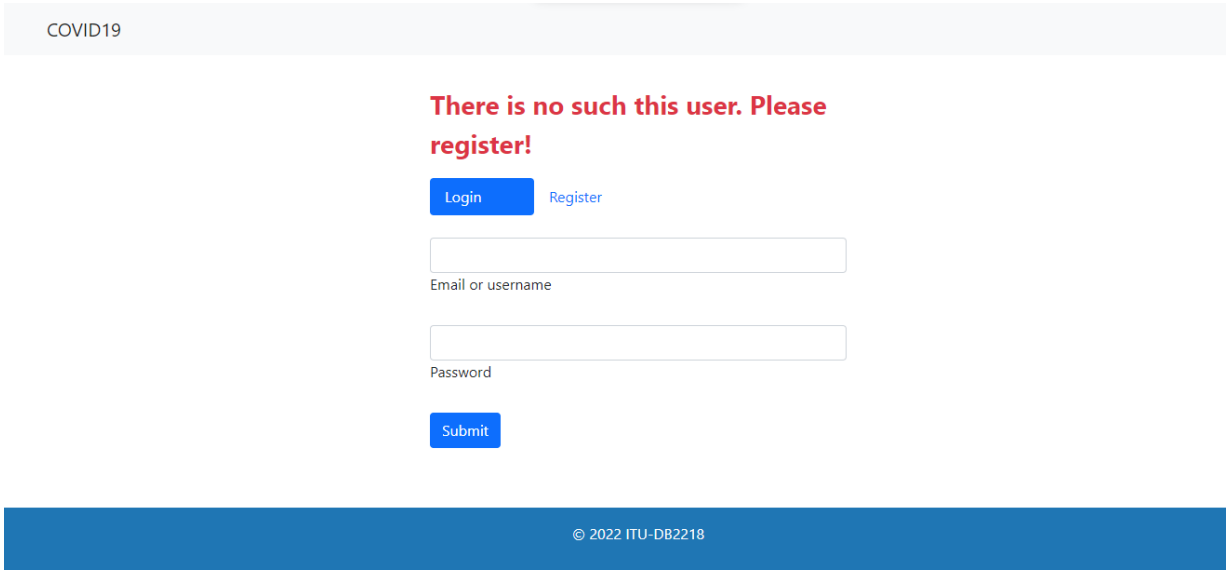
Figure 2.1: Login Page



Users who have previously registered to the application can log in from this page.

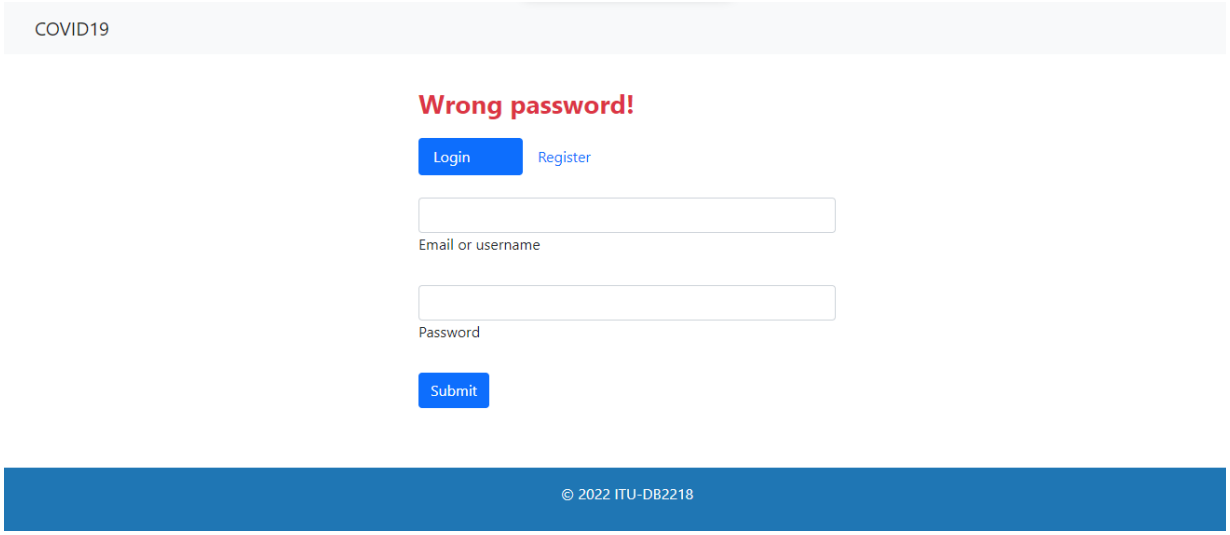
If a user has not registered to the application before, they will receive the following warning:

Figure 2.2: warning 1



If the user enters the password incorrectly, the following warning will appear:

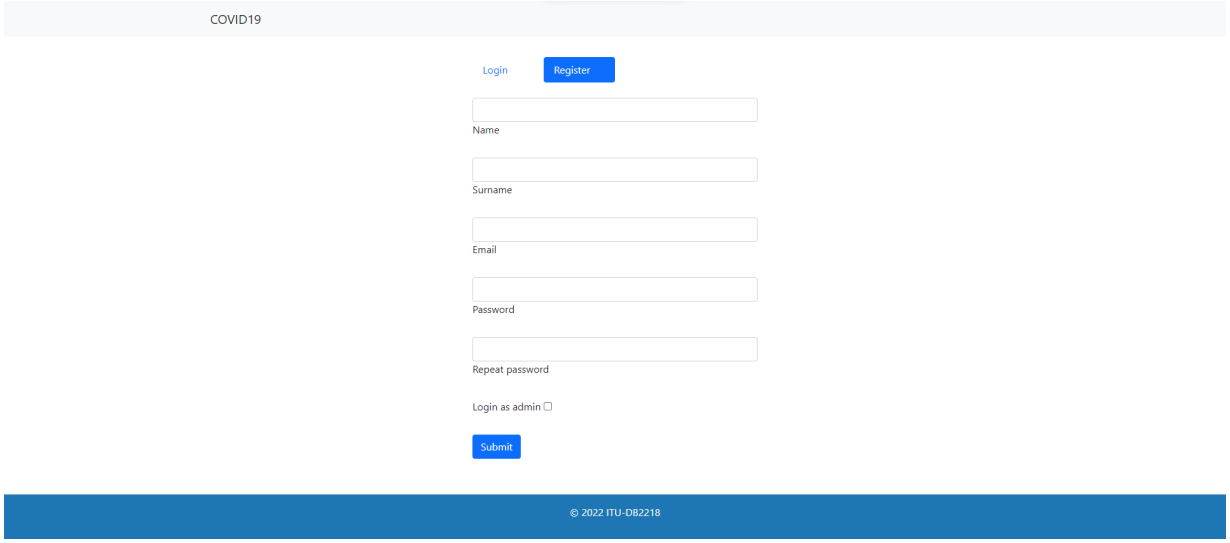
Figure 2.3: warning 2



2.1.2 Register Part

In this page, A user must enter all information completely when trying to register. In addition, each user must have a unique mail, otherwise an error message will appear.

Figure 2.4: Register Page



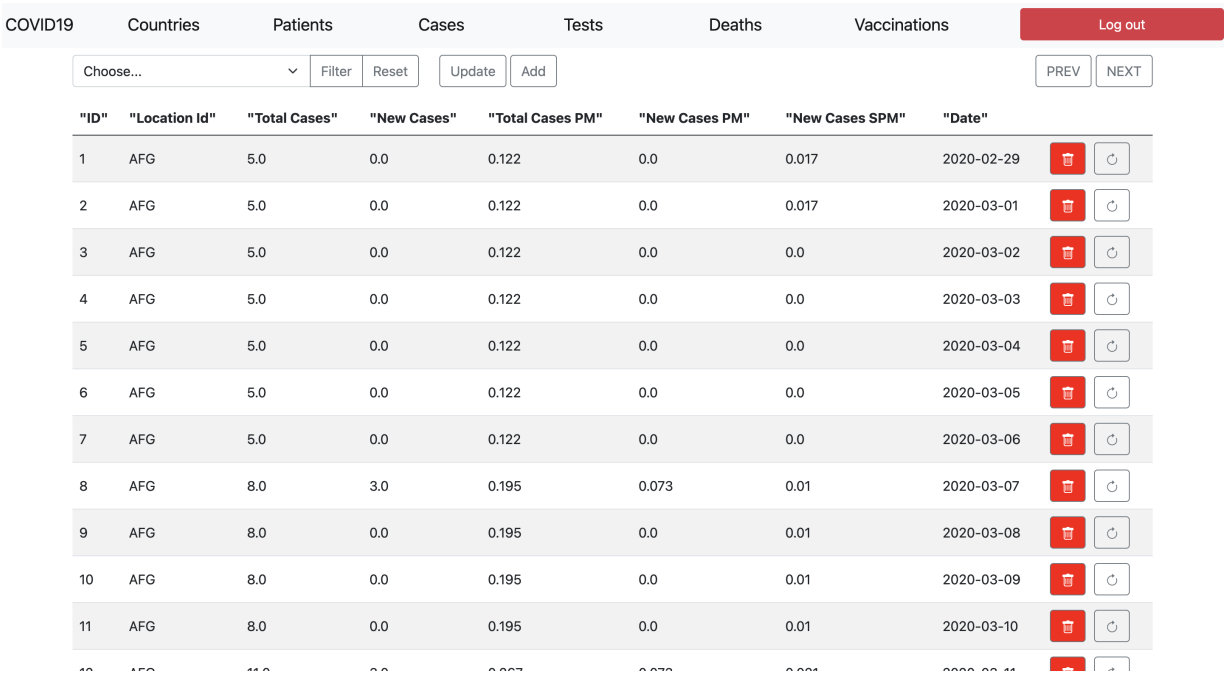
If a user wants to register as an admin, the admin password must be entered correctly. (Hint: Admin Password is "Bu grup 100 alacak")

Figure 2.5: Admin Password

2.2 Cases - Implemented by Talha şahin

2.2.1 Cases Page

Figure 2.6: Cases page

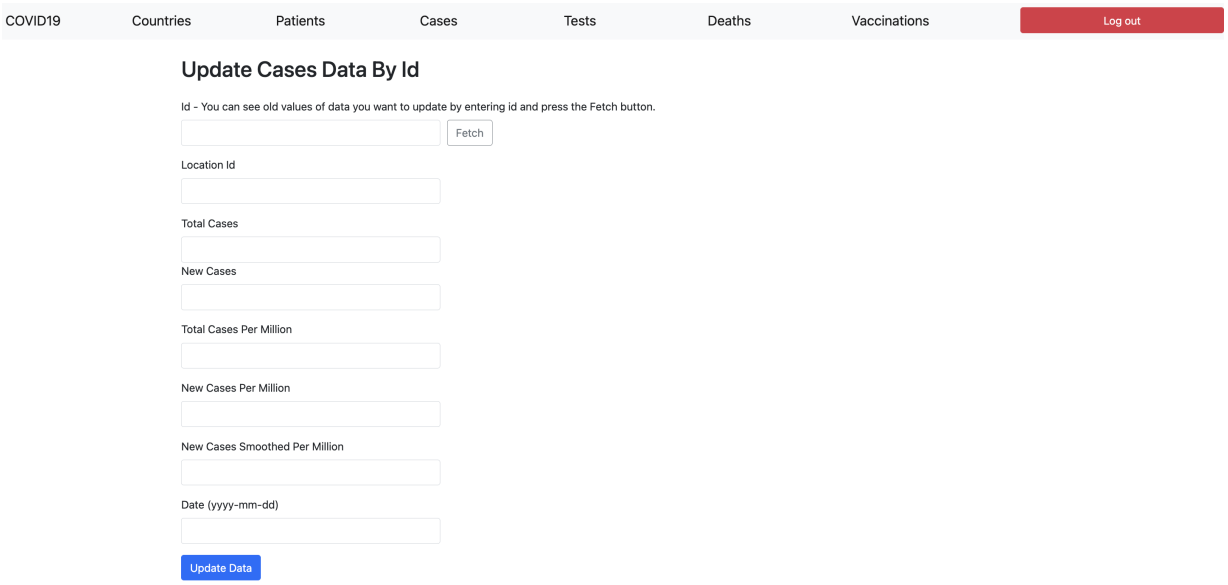


"ID"	"Location Id"	"Total Cases"	"New Cases"	"Total Cases PM"	"New Cases PM"	"New Cases SPM"	"Date"
1	AFG	5.0	0.0	0.122	0.0	0.017	2020-02-29
2	AFG	5.0	0.0	0.122	0.0	0.017	2020-03-01
3	AFG	5.0	0.0	0.122	0.0	0.0	2020-03-02
4	AFG	5.0	0.0	0.122	0.0	0.0	2020-03-03
5	AFG	5.0	0.0	0.122	0.0	0.0	2020-03-04
6	AFG	5.0	0.0	0.122	0.0	0.0	2020-03-05
7	AFG	5.0	0.0	0.122	0.0	0.0	2020-03-06
8	AFG	8.0	3.0	0.195	0.073	0.01	2020-03-07
9	AFG	8.0	0.0	0.195	0.0	0.01	2020-03-08
10	AFG	8.0	0.0	0.195	0.0	0.01	2020-03-09
11	AFG	8.0	0.0	0.195	0.0	0.01	2020-03-10

As seen in the figure, we have some buttons under the header and data below it. All the attributes of our data are shown to the user in the form of a table.
Buttons at the top of the page:
Choose: In the Choose section, we select the country whose data we want to see. Filter: After selecting the country, we press the Filter button and filter the data with the data of the selected country.
Reset: If we want to see all country data, we use this button to reset the filtering.
Add: The Add button leads to the data add page.
Update: The Update button redirects to the data update page.
NEXT AND PREV Buttons: We see the next 100 data with Next. With prev we see the previous 100 data.
Buttons next to data: As you can see, there are two buttons on the far right of each row (data).
Delete: It is the delete button on the left (with the trash icon) that deletes the data in the row it is in.
Update: The one on the right (with the refresh icon) is the button that leads to the page that updates the relevant data.

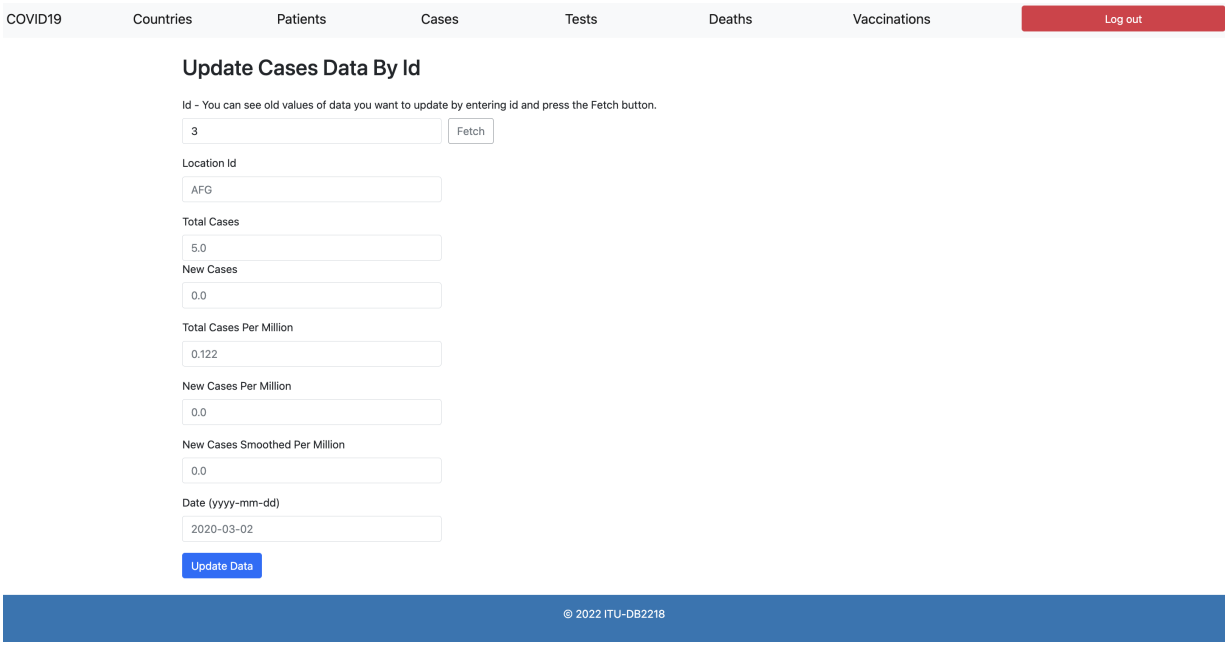
2.2.2 Cases Update Page

Figure 2.7: Cases Update Page



On the update page, there is an input for each attribute of the cases table, and that value of the data related to the input entered here can be updated.
The first part to be entered is the id of the data to be updated. After entering the Id, what you need to do is to press the Fetch button next to this field. After pressing the Fetch button, the value of all attributes of the relevant data will appear in the places where the value will be entered. In this way, reference values will be for the user.

Figure 2.8: Cases Update Page Fetched Data



The user does not need to enter all the values of the relevant data to update the data. He/she only enters the data he/she wants to update, the remaining data will be preserved in the same

way.
After all the values are entered, the Update Data button at the bottom is pressed.

After pressing the Update button, the page will return a warning to you. This alert will let you know if your update was successful or not. You can see example warnings in the two figures below.

Figure 2.9: Cases Successful Update Data Alert

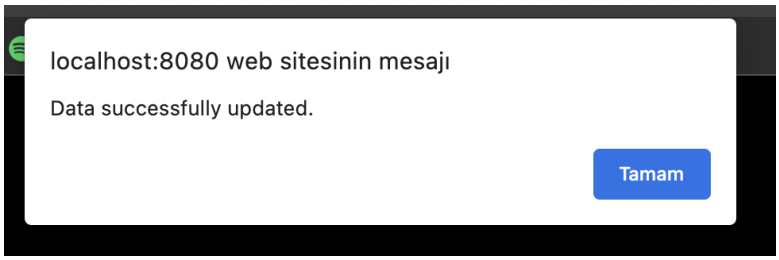
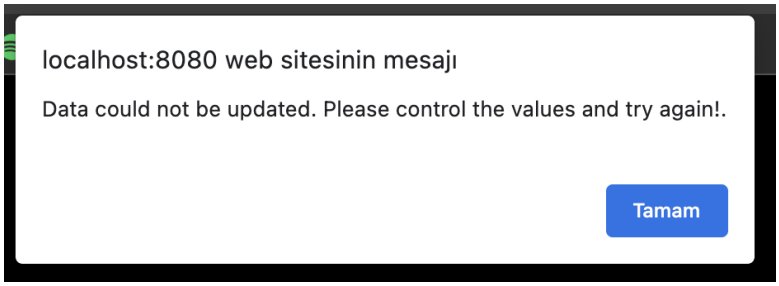
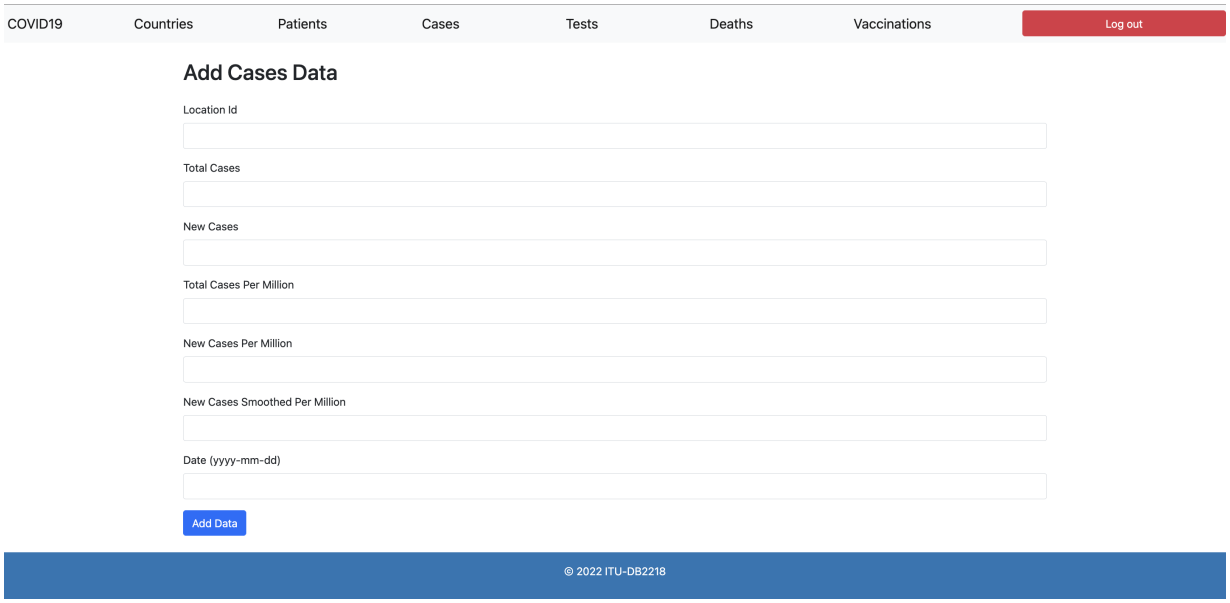


Figure 2.10: Cases Failed Update Data Alert



2.2.3 Cases Add Page

Figure 2.11: Cases Add Page



On the add page, there is an input for each attribute of the cases table. You should write your values of your data will be added.
After all the values are entered, the Add Data button at the bottom is pressed.

After pressing the Add Data button, the page will return a warning to you. This alert will let you know if your add was successful or not. You can see example warnings in the two figures below.

Figure 2.12: Cases Successful Add Data Alert

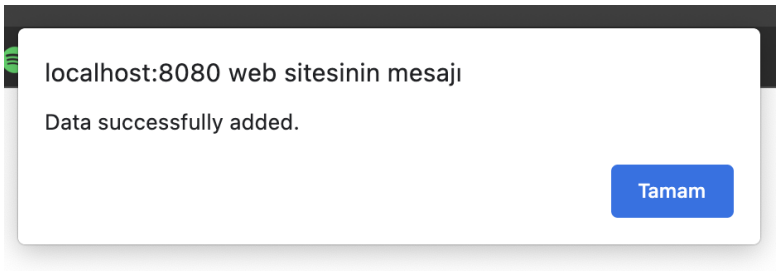
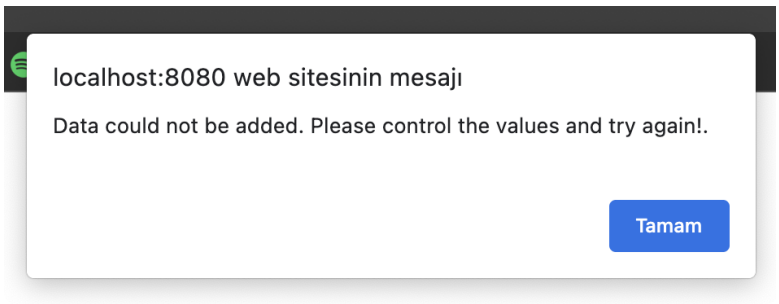


Figure 2.13: Cases Failed Add Data Alert



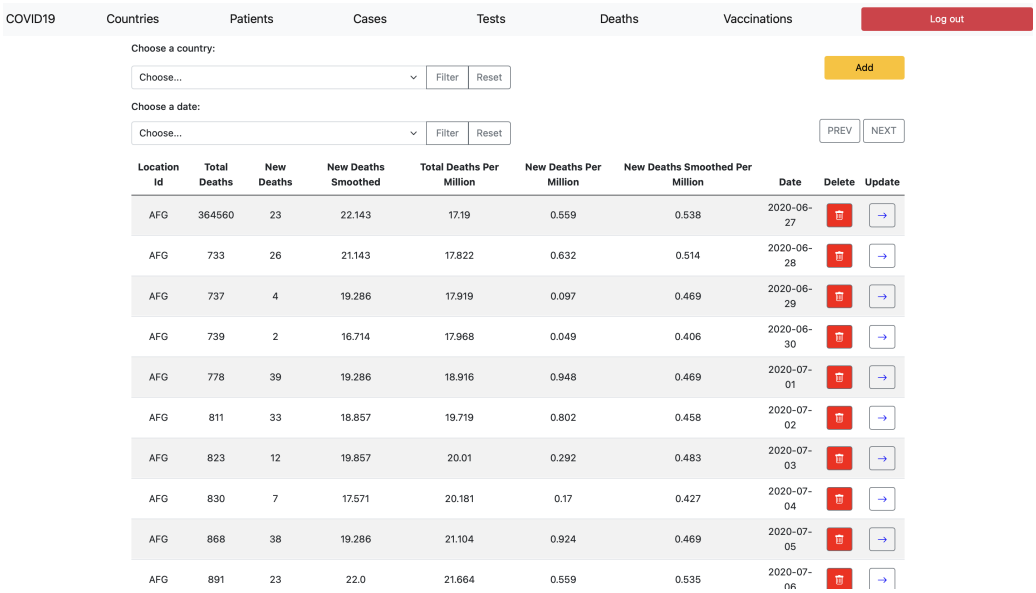
2.3 Deaths - Implemented by Zehra Asan

On the Deaths page, the user can access the number of deaths for different countries and their distribution in different ways.

2.3.1 Deaths Page

When the Deaths page is opened, the options on the page and their permissions are shaped differently according to two different users. There are two different page designs for someone who logs in as an admin in the login section and someone who can log in without admin authority.

While someone logging in without an admin can only filter the data in the table, someone logging in as an admin can modify, delete or add new data on these tables. Buttons with these different authorizations appear on the page according to the way they are entered. Delete, update and add buttons become visible with admin login. To delete a data, it is necessary to click on the trash icon next to each row.



Location Id	Total Deaths	New Deaths	New Deaths Smoothed	Total Deaths Per Million	New Deaths Per Million	New Deaths Smoothed Per Million	Date	Delete	Update
AFG	364560	23	22.143	17.19	0.559	0.538	2020-06-27		
AFG	733	26	21.143	17.822	0.632	0.514	2020-06-28		
AFG	737	4	19.286	17.919	0.097	0.469	2020-06-29		
AFG	739	2	16.714	17.968	0.049	0.406	2020-06-30		
AFG	778	39	19.286	18.916	0.948	0.469	2020-07-01		
AFG	811	33	18.857	19.719	0.802	0.458	2020-07-02		
AFG	823	12	19.857	20.01	0.292	0.483	2020-07-03		
AFG	830	7	17.571	20.181	0.17	0.427	2020-07-04		
AFG	868	38	19.286	21.104	0.924	0.469	2020-07-05		
AFG	891	23	22.0	21.664	0.559	0.535	2020-07-06		

Figure 2.14: Deaths Page for Admins

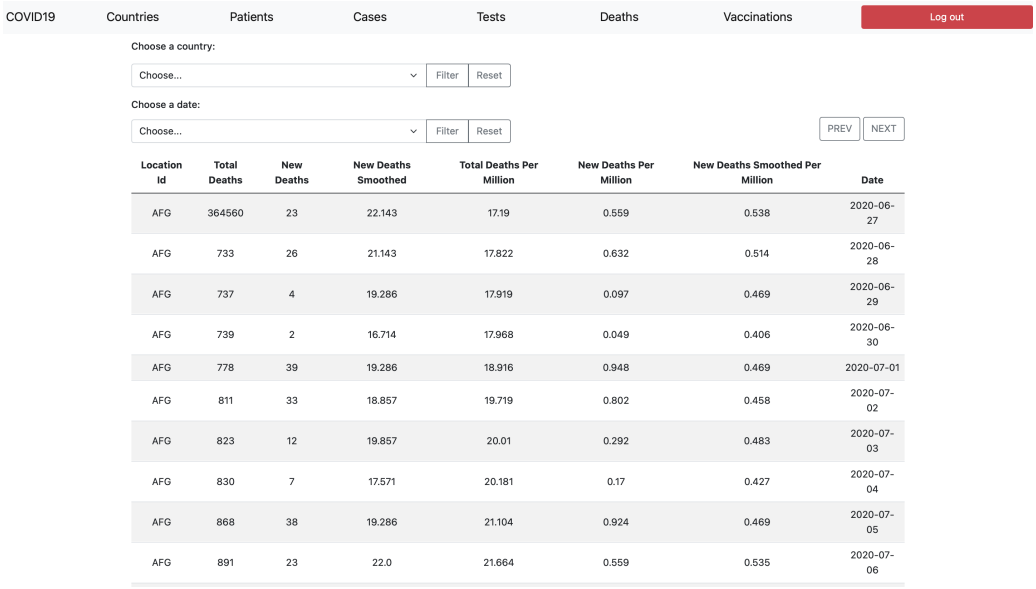


Figure 2.15: Deaths Page for Not an Admin Users

Either way, the user who logs in has access to the ability to filter data by country or by date. With these filters, data on specific dates or countries can be examined separately. With the reset button next to the filter button, the table can be returned to its unfiltered state.

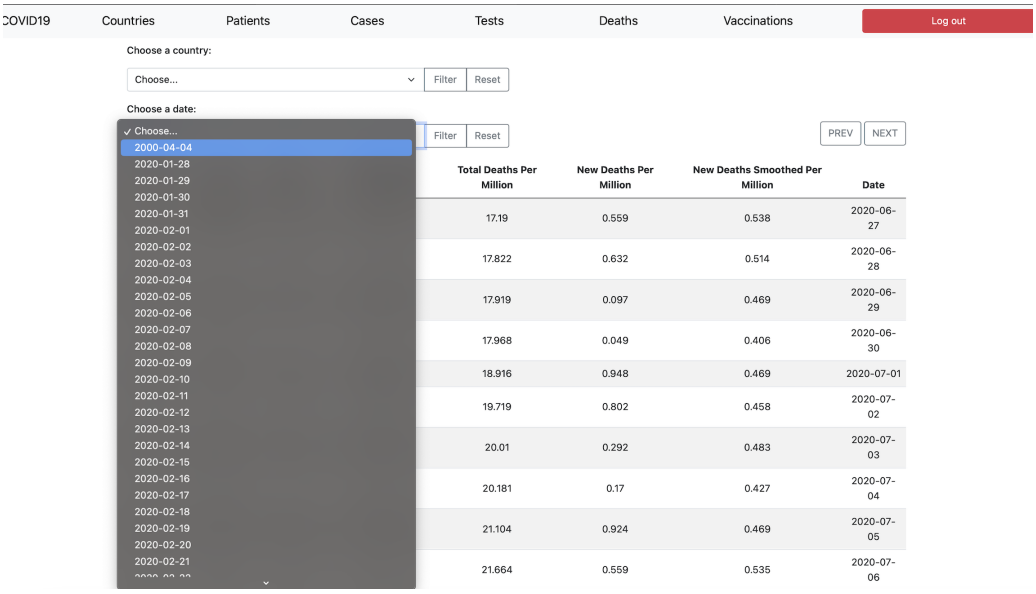


Figure 2.16: Deaths Page Date Filter

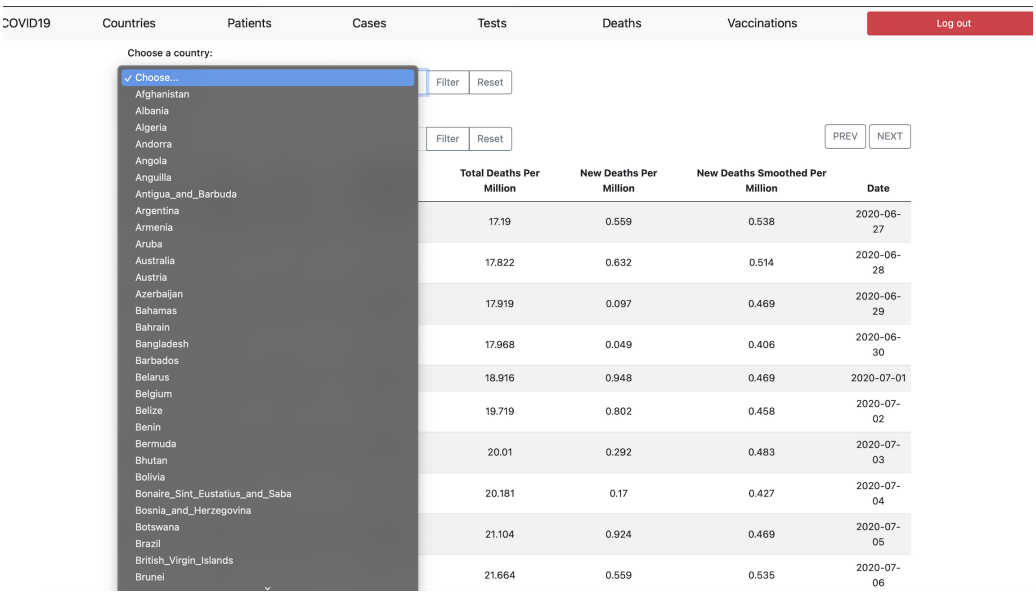


Figure 2.17: Deaths Page Location Filter

Finally, we can view the data page by page with the prev and next buttons. In this way, we can easily follow the data in order without scrolling down the page for a long time.

Figure 2.18: Prev and Next Buttons

2.3.2 Deaths Update Page

The update page, which can be accessed by a user logging in as an administrator, opens with the update button next to the lines. On this page, the data to be updated can be displayed in a faint form. The user can submit a single data or submit by making changes to more than one data.

Figure 2.19: Deaths Page Update Form

If the data is successfully updated, a confirmation message is received as in the figure:

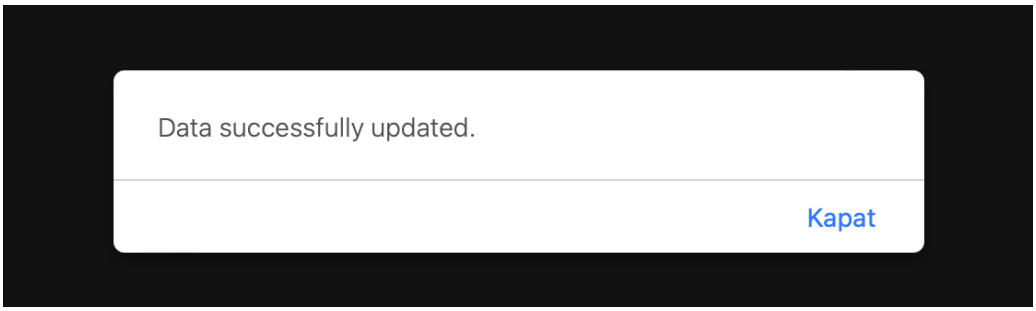


Figure 2.20: Deaths Page Update Success

If appropriate values cannot be entered, a failed message is given:

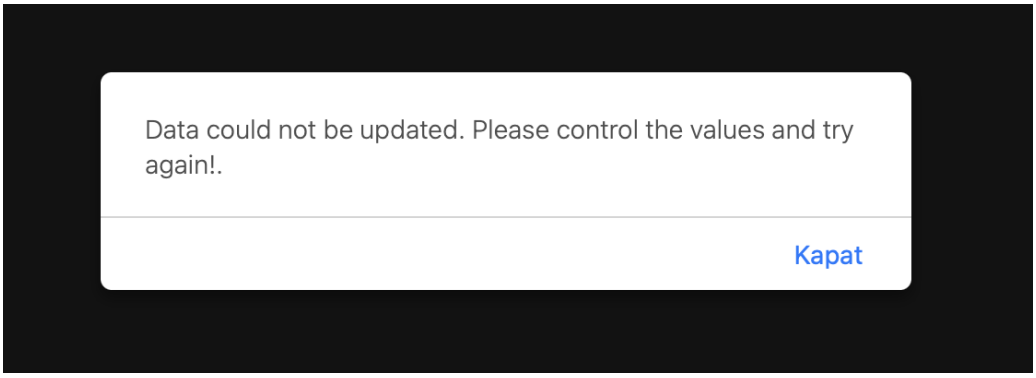


Figure 2.21: Deaths Page Update Fail

2.3.3 Deaths Add Page

A user logged in with admin permissions can access this page by clicking the Add button. In this form, the user can enter the necessary data and add data to the table. If the data is successfully added, a pop-up message will be displayed stating this.

Figure 2.22: Deaths Page Add Form

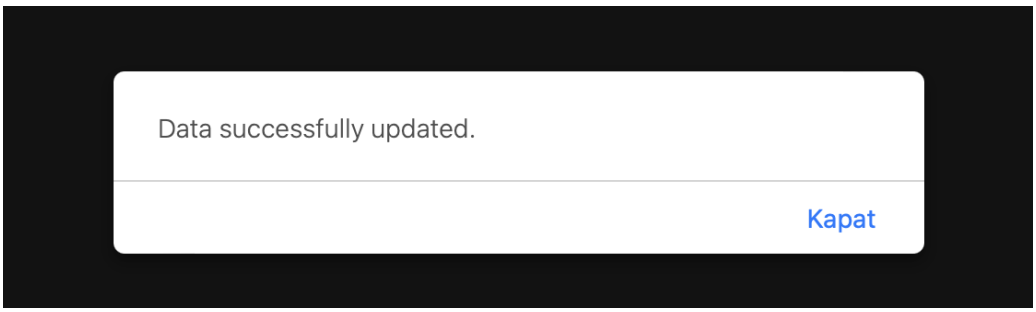


Figure 2.23: Deaths Page Add Success

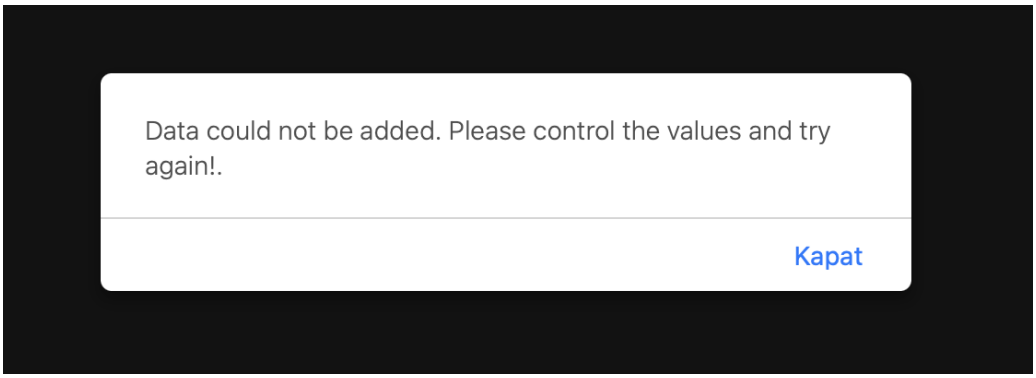


Figure 2.24: Deaths Page Add Fail

2.4 Tests - Implemented by Mert Arabacı

In the Tests Page, users can see information about Covid tests such as total tests, new tests per day in each country.

2.4.1 Tests Page

When an user opens the Tests page, the content of this page changes depending on admin status of the user. If the user is an admin, delete and update buttons show up in each row of the table and the yellow add button becomes available. Admins can easily remove or change the rows they wanted by clicking the delete and update buttons However if the user is not an admin, these buttons are hidden.

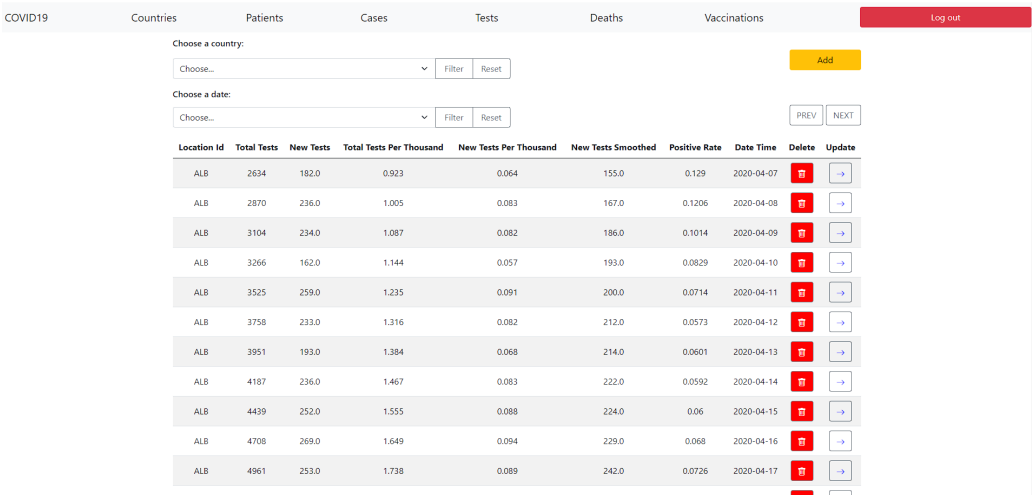


Figure 2.25: Tests Page for Admins

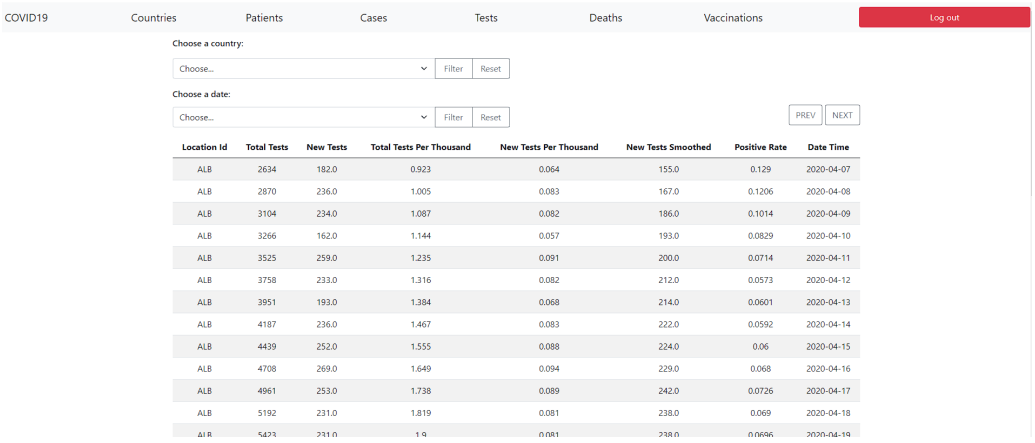
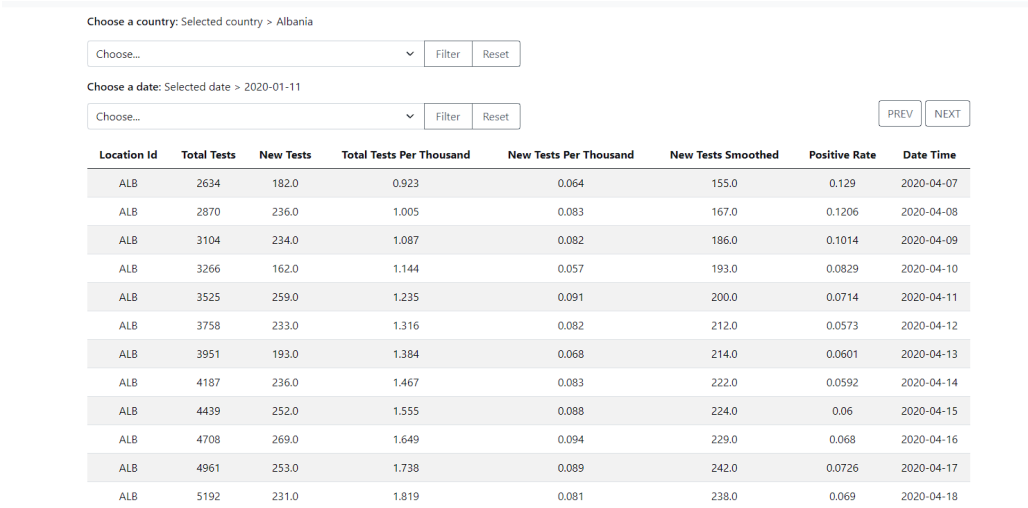


Figure 2.26: Tests Page for Normal Users

Tests table can be filtered by country and date. Users can select a specific country to view and choose the date to limit the starting date of the table. When the table is filtered by the country or date, current selection is shown above selection boxes. If the user want to remove the filter, reset button can be used to remove country or date filters.



Choose a country: Selected country > Albania

Choose... Filter Reset

Choose a date: Selected date > 2020-01-11

Choose... Filter Reset PREV NEXT

Location Id	Total Tests	New Tests	Total Tests Per Thousand	New Tests Per Thousand	New Tests Smoothed	Positive Rate	Date Time
ALB	2634	182.0	0.923	0.064	155.0	0.129	2020-04-07
ALB	2870	236.0	1.005	0.083	167.0	0.1206	2020-04-08
ALB	3104	234.0	1.087	0.082	186.0	0.1014	2020-04-09
ALB	3266	162.0	1.144	0.057	193.0	0.0829	2020-04-10
ALB	3525	259.0	1.235	0.091	200.0	0.0714	2020-04-11
ALB	3758	233.0	1.316	0.082	212.0	0.0573	2020-04-12
ALB	3951	193.0	1.384	0.068	214.0	0.0601	2020-04-13
ALB	4187	236.0	1.467	0.083	222.0	0.0592	2020-04-14
ALB	4439	252.0	1.555	0.088	224.0	0.06	2020-04-15
ALB	4708	269.0	1.649	0.094	229.0	0.068	2020-04-16
ALB	4961	253.0	1.738	0.089	242.0	0.0726	2020-04-17
ALB	5192	231.0	1.819	0.081	238.0	0.069	2020-04-18

Figure 2.27: Tests Page After Country and Date Filters

Lastly, prev and next buttons at the right hand side are used to change table's pages. If there are not any more data, a pop-up window shows up and warns the user.

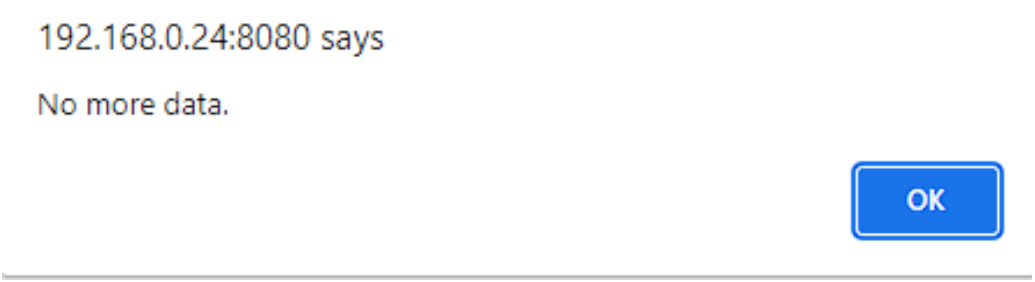


Figure 2.28: The Warning About The Absence of Next Page

2.4.2 Tests Page — Update Data

When admins click the edit button in a row, they are directed into an update page that changes the corresponding row of the table. In the update page, current values of the row are shown inside each input box and admins can change any column in that row. Admins have to enter appropriate date value to date box and numerical values to other input boxes.

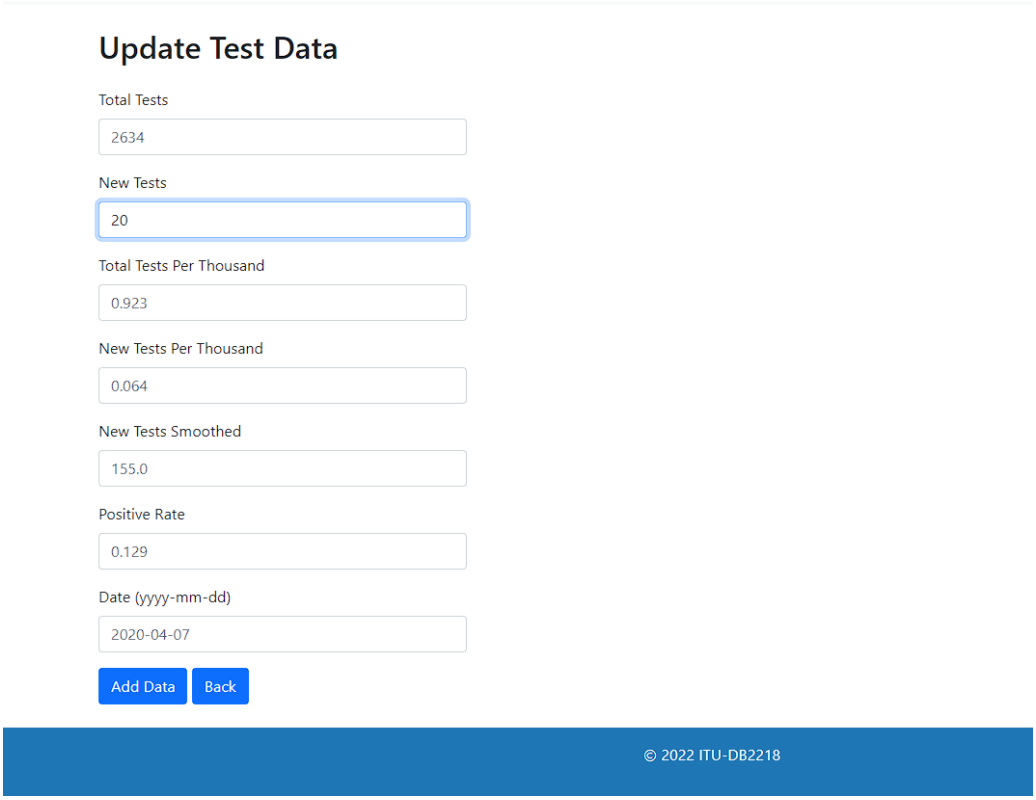


Figure 2.29: Update Page

If update operation is successful 'Data updated' pop-up will be shown. However, if admins fail to update the data, for example if they write string in the box where they have to write a number, data will not be updated and another pop up will be shown.

Figure 2.30: Successful Update Operation

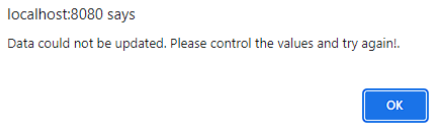


Figure 2.31: Failed Update Operation

2.4.3 Tests Page — Add New Data

When admins click the add button, they are directed into an add page. In the add page, admins have to enter appropriate date value to date box, iso code of a country to location id box and numerical values to other input boxes. Lastly, total tests and new tests input cannot be empty.

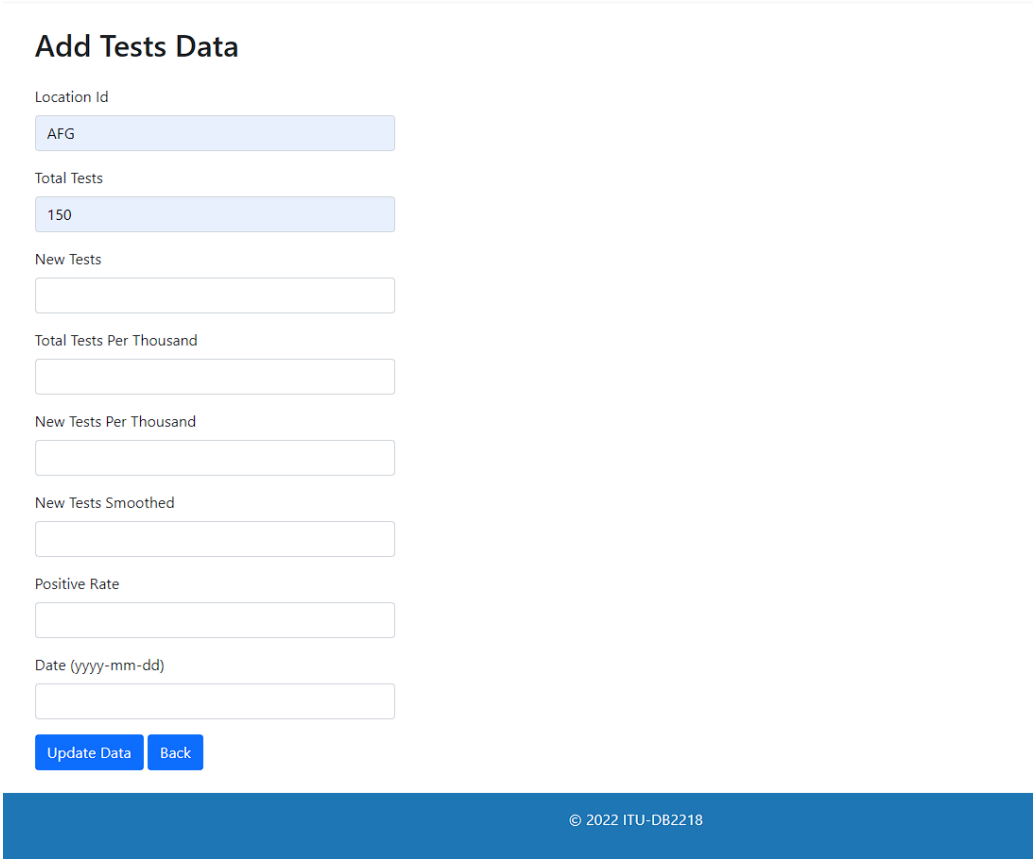


Figure 2.32: Update Page

If add operation is successful 'Data added' pop-up will be shown. However, if admins fail to add data, for example if both total tests and new tests inputs are not filled, data will not be added and another pop up will be shown.

Figure 2.33: Successful Add Operation

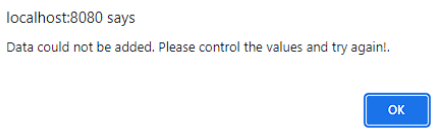


Figure 2.34: Failed Add Operation

2.5 Patients - Implemented by Cemalettin Celal Toy

In the Patients part of Application, hospital and intensive care unit data are displayed. Also, this data is taken from the Hospital_and_ICU table in the dataset.

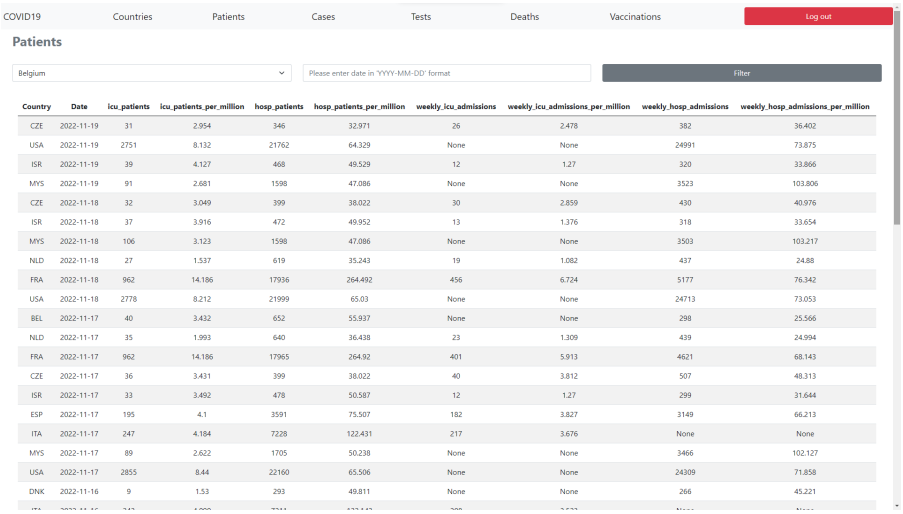
2.5.1 Patients Page

On this page, there is a table showing the data, a country and date filter, and a button that directs you to the edit page if you are logged in as an Admin.

Edit button can be seen by only Admins.

Normal users can see only this page:

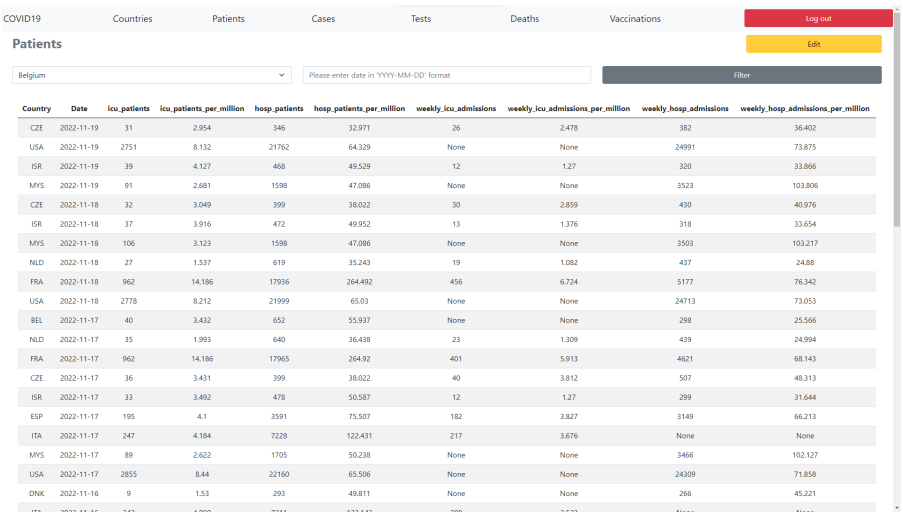
Figure 2.35: Patients Page for Normal Users



Country	Date	icu_patients	icu_patients_per_million	hosp_patients	hosp_patients_per_million	weekly_icu_admissions	weekly_icu_admissions_per_million	weekly_hosp_admissions	weekly_hosp_admissions_per_million
CZE	2022-11-19	31	2.954	346	32.971	26	2.478	382	36.402
USA	2022-11-19	2751	8.132	21762	64.329	None	None	24991	73.875
ISR	2022-11-19	39	4.127	468	49.529	12	1.27	320	33.866
MYS	2022-11-19	91	2.681	1598	47.086	None	None	3523	103.806
CZE	2022-11-18	32	3.049	399	38.022	30	2.859	430	40.976
ISR	2022-11-18	37	3.916	472	49.952	13	1.376	318	33.654
MYS	2022-11-18	106	3.123	1598	47.086	None	None	3503	103.217
NLD	2022-11-18	27	1.537	619	35.243	19	1.082	437	24.88
FRA	2022-11-18	962	14.186	17936	264.492	456	6.724	5177	76.342
USA	2022-11-18	2778	8.212	21999	65.03	None	None	24713	73.053
BEL	2022-11-17	40	3.432	652	55.937	None	None	298	25.566
NLD	2022-11-17	35	1.993	640	36.438	23	1.309	439	24.994
FRA	2022-11-17	962	14.186	17965	264.92	401	5.913	4621	68.143
CZE	2022-11-17	36	3.431	399	38.022	40	3.812	507	48.313
ISR	2022-11-17	33	3.492	478	50.587	12	1.27	299	31.644
ESP	2022-11-17	195	4.1	3591	75.507	182	3.827	3149	66.213
ITA	2022-11-17	247	4.184	7228	122.431	217	3.676	None	None
MYS	2022-11-17	89	2.622	1705	50.238	None	None	3466	102.127
USA	2022-11-17	2855	8.44	22160	65.506	None	None	24309	71.858
DNK	2022-11-16	9	1.53	293	49.811	None	None	266	45.221

Admins see this page:

Figure 2.36: Patients Page for Admins



Country	Date	icu_patients	icu_patients_per_million	hosp_patients	hosp_patients_per_million	weekly_icu_admissions	weekly_icu_admissions_per_million	weekly_hosp_admissions	weekly_hosp_admissions_per_million
CZE	2022-11-19	31	2.954	346	32.971	26	2.478	382	36.402
USA	2022-11-19	2751	8.132	21762	64.329	None	None	24991	73.875
ISR	2022-11-19	39	4.127	468	49.529	12	1.27	320	33.866
MYS	2022-11-19	91	2.681	1598	47.086	None	None	3523	103.806
CZE	2022-11-18	32	3.049	399	38.022	30	2.859	430	40.976
ISR	2022-11-18	37	3.916	472	49.952	13	1.376	318	33.654
MYS	2022-11-18	106	3.123	1598	47.086	None	None	3503	103.217
NLD	2022-11-18	27	1.537	619	35.243	19	1.082	437	24.88
FRA	2022-11-18	962	14.186	17936	264.492	456	6.724	5177	76.342
USA	2022-11-18	2778	8.212	21999	65.03	None	None	24713	73.053
BEL	2022-11-17	40	3.432	652	55.937	None	None	298	25.566
NLD	2022-11-17	35	1.993	640	36.438	23	1.309	439	24.994
FRA	2022-11-17	962	14.186	17965	264.92	401	5.913	4621	68.143
CZE	2022-11-17	36	3.431	399	38.022	40	3.812	507	48.313
ISR	2022-11-17	33	3.492	478	50.587	12	1.27	299	31.644
ESP	2022-11-17	195	4.1	3591	75.507	182	3.827	3149	66.213
ITA	2022-11-17	247	4.184	7228	122.431	217	3.676	None	None
MYS	2022-11-17	89	2.622	1705	50.238	None	None	3466	102.127
USA	2022-11-17	2855	8.44	22160	65.506	None	None	24309	71.858
DNK	2022-11-16	9	1.53	293	49.811	None	None	266	45.221

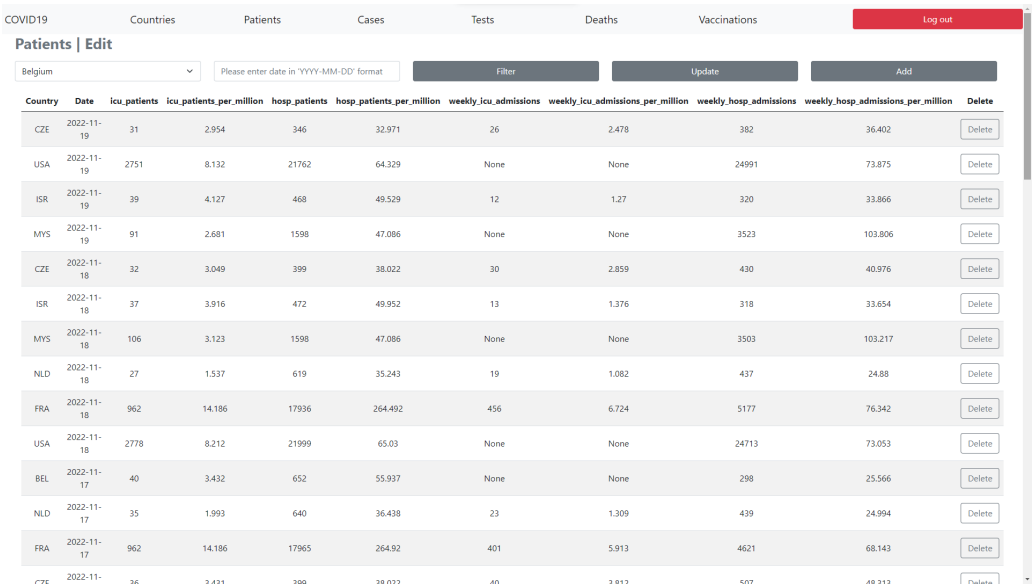
If a user select a country and push the filter button, only the data of that country is displayed on the screen. Moreover, if a user also selects a date as an extra, only the data of that country on that date will appear.

When a user clicks the Edit button, the page is redirected to the edit page.

2.5.2 Patients—Edit Page

In this page, if a user is an admin, they can delete data, and also go to add and update data pages.

Figure 2.37: Patients Edit Page



Country	Date	icu_patients	icu_patients_per_million	hosp_patients	hosp_patients_per_million	weekly_icu_admissions	weekly_icu_admissions_per_million	weekly_hosp_admissions	weekly_hosp_admissions_per_million	Delete
CZE	2022-11-19	31	2.954	346	32.971	26	2.478	382	36.402	Delete
USA	2022-11-19	2751	8.132	21762	64.329	None	None	24991	73.875	Delete
ISR	2022-11-19	39	4.127	468	49.529	12	1.27	320	33.866	Delete
MYS	2022-11-19	91	2.681	1598	47.086	None	None	3523	103.806	Delete
CZE	2022-11-18	32	3.049	399	38.022	30	2.859	430	40.976	Delete
ISR	2022-11-18	37	3.916	472	49.952	13	1.376	318	33.654	Delete
MYS	2022-11-18	106	3.123	1598	47.086	None	None	3503	103.217	Delete
NLD	2022-11-18	27	1.537	619	35.243	19	1.082	437	24.88	Delete
FRA	2022-11-18	962	14.186	17936	264.492	456	6.724	5177	76.342	Delete
USA	2022-11-18	2778	8.212	21999	65.03	None	None	24713	73.053	Delete
BEL	2022-11-17	40	3.432	652	55.937	None	None	298	25.566	Delete
NLD	2022-11-17	35	1.993	640	36.438	23	1.309	439	24.994	Delete
FRA	2022-11-17	962	14.186	17965	264.92	401	5.913	4621	68.143	Delete

A user can filter the data as in patients page, and can delete this data as pressing the delete button next to the data row.

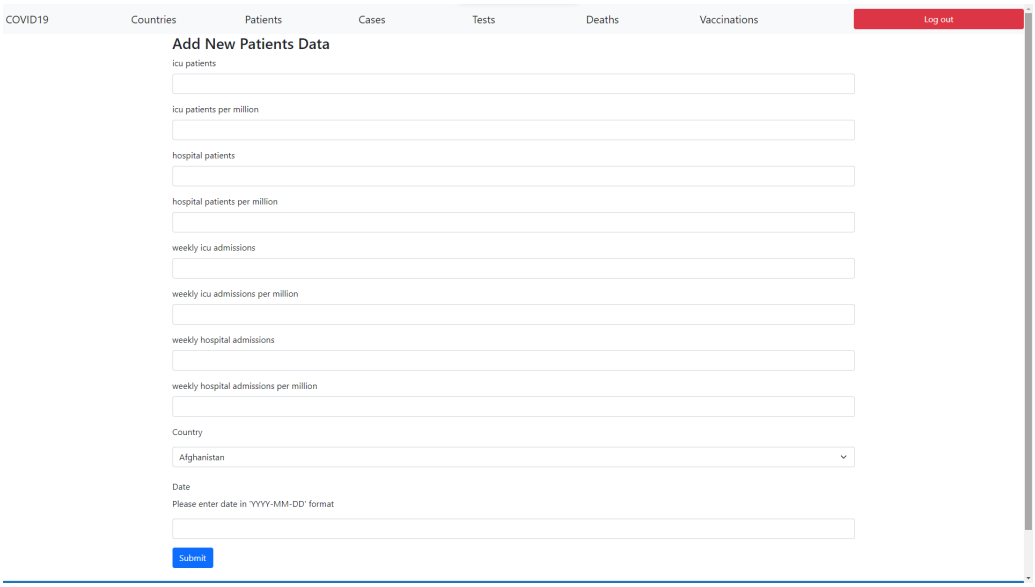
If User wants to add a new data, they should press the add button. This button redirect the current page to adding page.

If User wants to update the data, they should press the update button. This button redirect the current page to update page.

2.5.3 Patients—Add Page

In this page, A user can add a new record.

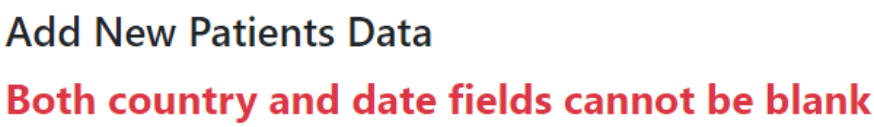
Figure 2.38: Patients Add Page



While recording, the date, country, icu patients and hospital patients sections should not be empty. If a user try these, the following warnings appear:

Figure 2.39: Blank data

Figure 2.40: Blank country or Date

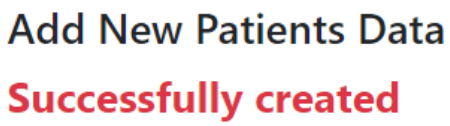


Also, a user cannot add a record has same country and date. If a user try these, the following warnings appear:

Figure 2.41: Exist record message

If the user can add the data this text will appear on the screen:

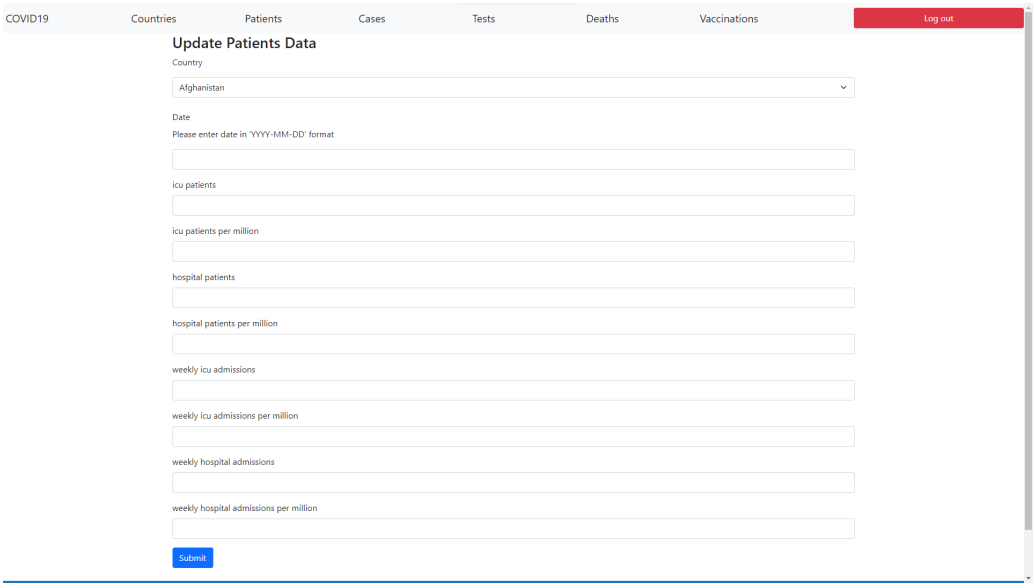
Figure 2.42: Successfully Created message



2.5.4 Patients—Update Page

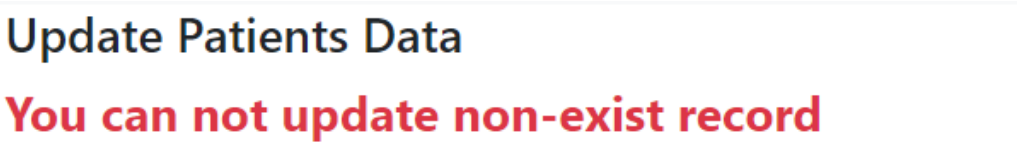
In this page, a user can update a exist record.

Figure 2.43: Update Page



If a user try to update non-exist record, the following warnings appear:

Figure 2.44: Non-exist record



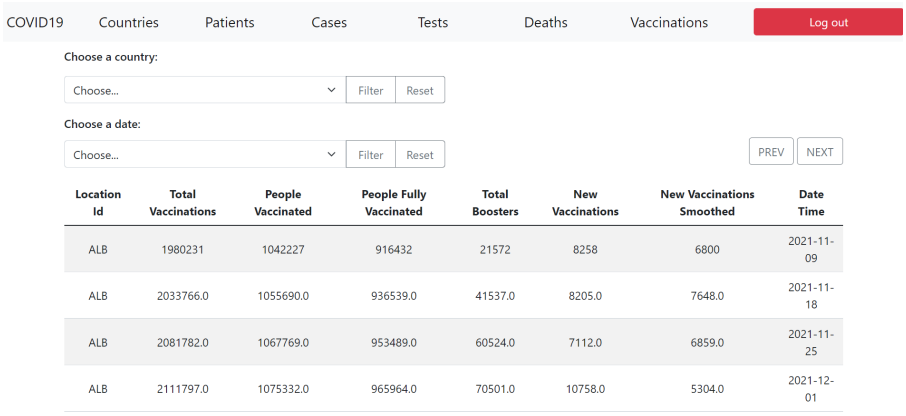
2.6 Vaccinations - Implemented by Hilal Erdoğan

2.6.1 Vaccinations Page

The table on this page shows the number of total vaccinations, individuals who have received vaccinations, those who have completed the full vaccination, booster doses, new vaccinations

given, and a 7-day smoothed version of new vaccinations from December 31, 2020 to November 20, 2022. The data can be filtered by country and date.

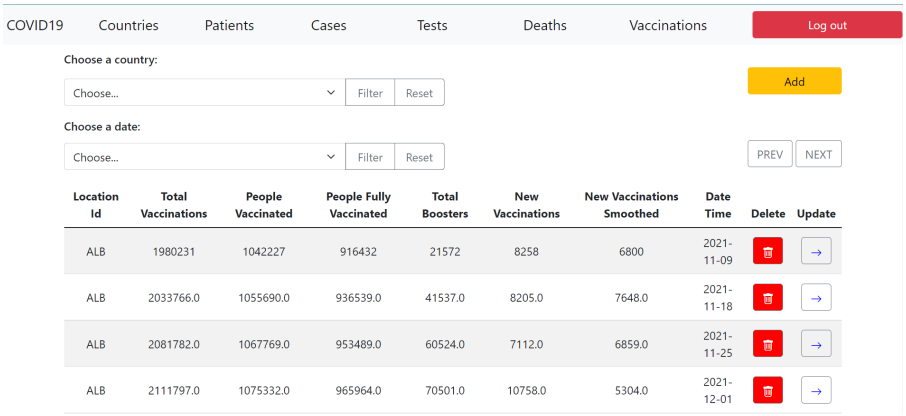
Figure 2.45: Vaccinations Page for Normal Users



Location Id	Total Vaccinations	People Vaccinated	People Fully Vaccinated	Total Boosters	New Vaccinations	New Vaccinations Smoothed	Date Time
ALB	1980231	1042227	916432	21572	8258	6800	2021-11-09
ALB	2033766.0	1055690.0	936539.0	41537.0	8205.0	7648.0	2021-11-18
ALB	2081782.0	1067769.0	953489.0	60524.0	7112.0	6859.0	2021-11-25
ALB	2111797.0	1075332.0	965964.0	70501.0	10758.0	5304.0	2021-12-01

The following page appears if and only if user is an admin. Admin can change the values by clicking "update" and giving new values, or delete with "delete" button. Also using the "Add" button totally new data can be integrated.

Figure 2.46: Vaccinations Page for Admin Users

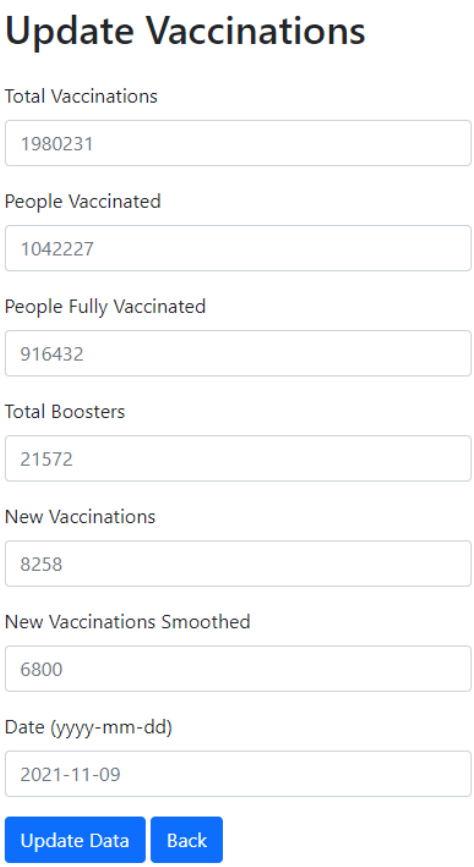


Location Id	Total Vaccinations	People Vaccinated	People Fully Vaccinated	Total Boosters	New Vaccinations	New Vaccinations Smoothed	Date Time	Delete	Update
ALB	1980231	1042227	916432	21572	8258	6800	2021-11-09		
ALB	2033766.0	1055690.0	936539.0	41537.0	8205.0	7648.0	2021-11-18		
ALB	2081782.0	1067769.0	953489.0	60524.0	7112.0	6859.0	2021-11-25		
ALB	2111797.0	1075332.0	965964.0	70501.0	10758.0	5304.0	2021-12-01		

2.6.2 Vaccinations Update Page

After clicking "Update" button Update Vaccinations Page appears. Here user can update any attribute, admin don't have to fill every blank.

Figure 2.47: Vaccinations Update for Admin Users



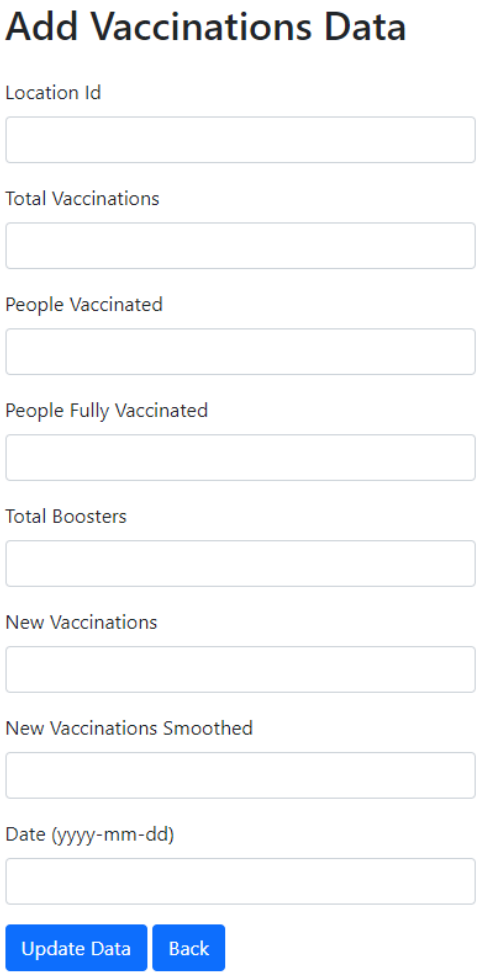
After clicking "Update Data" if process is done without a problem "data updated successfully" is appear as message.

Figure 2.48: Vaccinations Update Success

2.6.3 Vaccinations Add Page

After clicking "Add" button Add Vaccinations Page appears. Here user can add new data. To successfully add new data all blanks has to be filled.

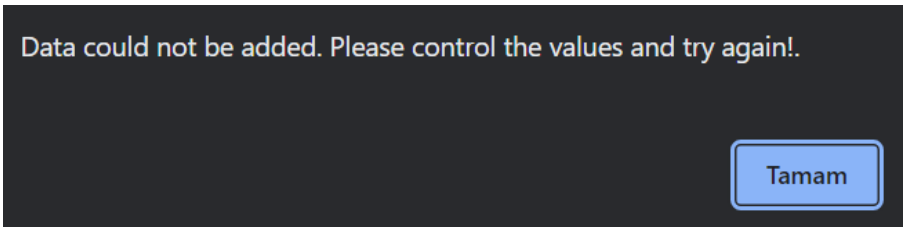
Figure 2.49: Vaccinations Add Page



User gets a message after clicking "Update" whether or not data is added.

Figure 2.50: Vaccinations Add Success

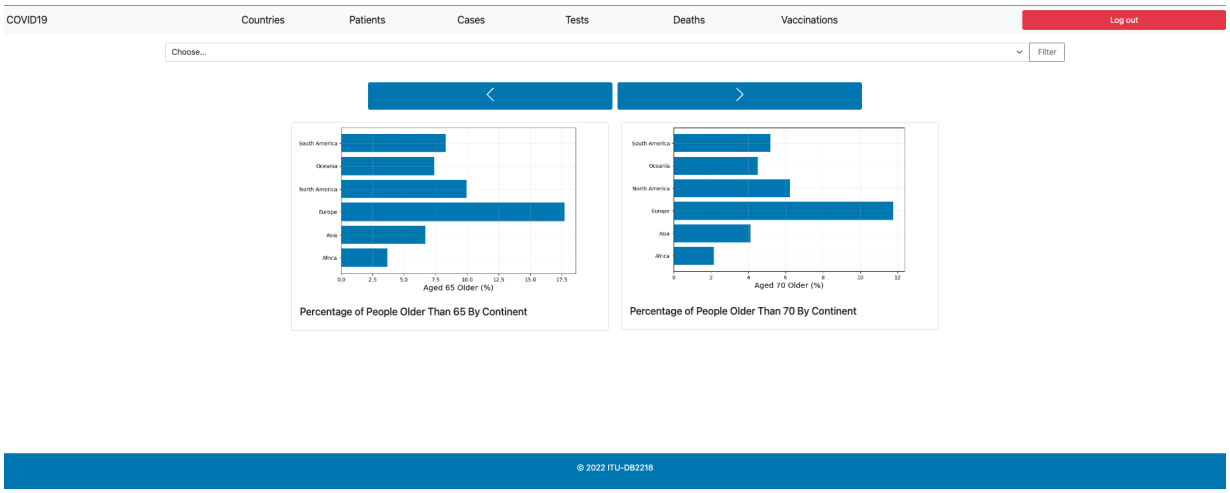
Figure 2.51: Vaccinations Add Error



2.7 Locations - Implemented by all team members together

2.7.1 Locations Home Page

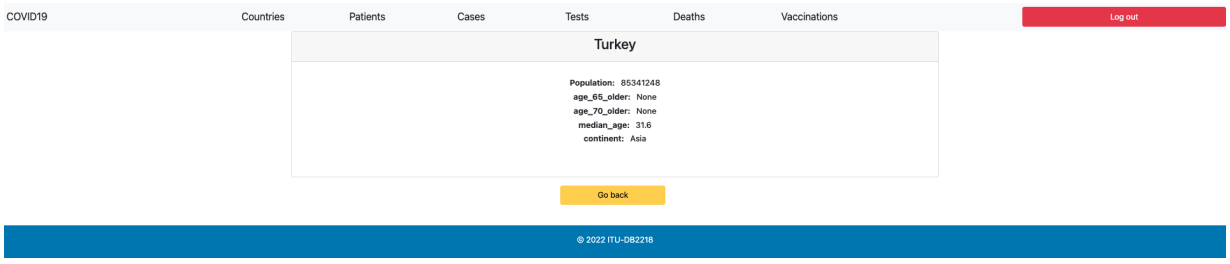
Figure 2.52: Locations Home Page



A visualized version of some values by continent can be seen on the homepage. You can navigate between these graphics with two buttons on the right and left.
On the Location page, there is a select option below the header and a filter button next to it. You need to select the country whose details you want to see from the drop-down tab and press the Filter button. Then there will be a redirection to the page where the information of the selected country is written.

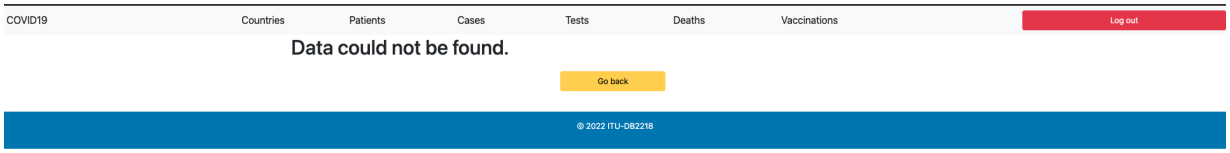
2.7.2 Location Information Page

Figure 2.53: Locations Information Page



The information of the selected country can be seen on the information page. With the Go Back button at the bottom, you can return to the Locations main page and select a different country. Regarding the selected country. If there is no information, a warning will appear as 'Data could not be found'.

Figure 2.54: Locations Not Found Information Page



Chapter 3

Developers Guide

3.1 Database Design

In the database, the data of the application and users are kept. It consists of seven separate tables in total. These tables are cases, covid_tests, deaths, hospital_and_icu, locations, vaccinations, and users.

In the User table, the users' name, surname, e-mail, password and information about whether they are admin or not are kept. Email addresses are unique to each user.

The locations table consists of a unique location_id as the primary key. In addition, the table includes continent name, country name, the number of population, the rate of those over 65, the rate of those over 70, and the median age data.

One thing the rest of the tables have in common is that they all depend on the locations table. There is a foreign key that connects them all to one country. In addition, there is date data in all tables except the locations and user tables. All remaining data in these tables are specific numeric data.

3.2 Codes

3.2.1 Login and Home Page - Implemented by Cemalettin Celal Toy, Zehra Asan, and Mert Can Arabacı

3.2.1.1 View Function of Login Page

Code 3.1: View Function of Login Page

```
from flask import Flask, render_template, flash, url_for, request,
   session
from werkzeug.utils import redirect
from model.user import *
```

```
def login_page():
    connection = User()
    session["id"] = None
    if(request.method == "POST"):
        check = request.args.get("check")
        if(check == "True"):
            mail = request.form["mailLogin"] if request.form["
               mailLogin"] !="" else None
            password = request.form["passwordLogin"]

            if mail is not None:
                information = connection.
                   selectByEmailReturnPasswordAndID(mail)
                if(information is None):
                    flash("There␣is␣no␣such␣this␣user.␣Please␣
                       register!")
                    return redirect("/")
                elif(password != information[0]):
                    flash("Wrong␣password!")
                    return redirect("/")
                else:
                    session["id"] = information[1]
                    return render_template("home.html", isHome=True
                       )
            else:
                flash("Please␣enter␣a␣mail␣for␣log-in")
                return redirect("/")
        else:
            name = request.form["name"]
            surname = request.form["surname"]
            mail = request.form["mail"]
            password = request.form["password"]
            password_2 = request.form["password2"]
            adminPassword = request.form["adminPassword"]
            check_adminPassword = "Bu␣grup␣100␣alacak"
            try:
                isAdmin = "true" if request.form["isAdmin"] == 'on'
                    else "false"
            except:
                isAdmin = "false"
            if((name !="")and(surname !="")and(mail !="")and(
               password !="")and(password_2 !="")):
                if password != password_2:
                    flash("These␣passwords␣are␣not␣same!")
                    return redirect("/")
                elif(isAdmin == "true" and adminPassword !=
                   check_adminPassword):
                    flash("Admin␣password␣is␣not␣correct!␣Nice␣try␣
                       :)")
                    return redirect("/")
```

```
                else:
                    check=False
                    emailCheck = False
                    if connection.selectByEmailReturnPasswordAndID(
                       email=mail) is not None:
                        emailCheck = True
                    else:
                        check = connection.register(name=name,
                           surname=surname,email=mail,password=
                           password,admin=isAdmin)

                    if check:
                        flash("Successfully␣registered.␣Please␣log-
                           in.")
                    else:
                        if emailCheck:
                            flash("This␣mail␣is␣used␣by␣another␣
                               user")
                        else:
                            flash("Invalid␣entries!")
                    return render_template("login.html")
            else:
                flash("These␣fields␣cannot␣be␣blank!")
                return redirect("/")

    return render_template("login.html")
```

3.2.1.2 HTML File of Login Page

Code 3.2: HTML file of Login page

```
{% extends "before_login.html" %}
{% block title %}Country{% endblock %}
{% block content %}


<div class="mx-auto␣m-3␣p-3" style="width:␣25%;">
  {% with messages = get_flashed_messages() %}
  {% if messages %}

  {% for message in messages %}
  <p class="text-danger␣text-start␣fs-3␣fw-bold">{{message}}</p>
  {% endfor %}

  {% endif %}
  {% endwith %}
  <ul class="nav␣nav-pills" id="myTab">
    <li class="nav-item" style="width:␣25%;">
```

```
      <a href="#login" class="nav-link␣active" data-bs-toggle="tab"
         >Login</a>
    </li>
    <li class="nav-item" style="width:␣25%;">
      <a href="#register" class="nav-link" data-bs-toggle="tab">
         Register</a>
    </li>
  </ul>
  <br />

  <div class="tab-content">
    <div class="tab-pane␣fade␣show␣active" id="login" role="
       tabpanel" aria-labelledby="tab-login">
      <form action="/?check=True" method="POST">

        <!-- Email input -->
        <div class="form-outline␣mb-4">
          <input type="email" id="loginName" name="mailLogin" class
             ="form-control" />
          <label class="form-label" for="loginName">Email or
             username</label>
        </div>

        <!-- Password input -->
        <div class="form-outline␣mb-4">
          <input type="password" id="loginPassword" name="
             passwordLogin" class="form-control" />
          <label class="form-label" for="loginPassword">Password</
             label>
        </div>

        <!-- 2 column grid layout -->

        <!-- Submit button -->
        <button id="loginButton" type="submit" class="btn␣btn-
           primary␣btn-block␣mb-4">Submit</button>

      </form>
    </div>
    <div class="tab-pane␣fade" id="register" role="tabpanel" aria-
       labelledby="tab-register">
      <form action="/?check=False" method="POST">

        <!-- Name input -->
        <div class="form-outline␣mb-4">
          <input type="text" id="registerName" name="name" class="
             form-control" />
          <label class="form-label" for="registerName">Name</label>
        </div>
```

```
        <!-- Username input -->
        <div class="form-outline␣mb-4">
          <input type="text" id="regiterSurName" name="surname"
             class="form-control" />
          <label class="form-label" for="regiterSurName">Surname</
             label>
        </div>

        <!-- Email input -->
        <div class="form-outline␣mb-4">
          <input type="email" id="registerEmail" name="mail" class=
             "form-control" />
          <label class="form-label" for="registerEmail">Email</
             label>
        </div>

        <!-- Password input -->
        <div class="form-outline␣mb-4">
          <input type="password" id="registerPassword" name="
             password" class="form-control" />
          <label class="form-label" for="registerPassword">Password
             </label>
        </div>

        <!-- Repeat Password input -->
        <div class="form-outline␣mb-4">
          <input type="password" id="registerRepeatPassword" name="
             password2" class="form-control" />
          <label class="form-label" for="registerRepeatPassword">
             Repeat password</label>
        </div>

        <!-- Checkbox -->

        <label class="form-label" for="loginPassword">Login as
           admin</label>
        <input type="checkbox" name="isAdmin" />
        <input type="password" id="adminPassword" name="
           adminPassword" placeholder="admin␣password" /> <br /> <
           br />

        <!-- Submit button -->
        <button id="registerButton" type="submit" class="btn␣btn-
           primary␣btn-block␣mb-3">Submit</button>
      </form>
    </div>
  </div>
  {% endblock %}
```

3.2.1.3 HTML File of Home Page

Code 3.3: HTML File of Home Page

```
{% extends "before_login.html" %}
{% block title %}Covid19 General Info{% endblock %}
{% block content %}
<div class="content-container">
    <div class="container␣mx-auto">
        <h3 class="text-center␣fw-bold␣mb-5"
            style="color:␣#173364ff;␣font-size:␣350%;␣font-style:␣
               italic;margin-top:␣5%;">General Information about
            Covid19</h3>
        <div class="row">
            <div class="col">
                <a href="/locations" style="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣text-decoration:␣none;color:␣whitesmoke;␣font-
   size:␣30px;">
                <button type="button" class="btn␣btn-secondary"
                    style="width:␣75%;height:␣200px;margin-left:␣
                       12%;␣margin-top:␣25%;margin-bottom:␣25%;␣
                       font-size:␣30px;"
                    data-toggle="button" aria-pressed="false"
                       autocomplete="off">
                    Locations
                </button></a>
            </div>
            <div class="col">
                <a href="/patients" style="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣text-decoration:␣none;color:␣whitesmoke;␣font-
   size:␣30px;">
                <button type="button" class="btn␣btn-secondary"
                    style="width:␣75%;height:␣200px;margin-left:␣
                       12%;␣margin-top:␣25%;margin-bottom:␣25%;␣
                       font-size:␣30px;"
                    data-toggle="button" aria-pressed="false"
                       autocomplete="off">
                    Patients
                </button></a>
            </div>
            <div class="col">
                <a href="/tests" style="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣text-decoration:␣none;color:␣whitesmoke;␣font-
   size:␣30px;">
                <button type="button" class="btn␣btn-secondary"
                    style="width:␣75%;height:␣200px;margin-left:␣
                       12%;␣margin-top:␣25%;margin-bottom:␣25%;␣
                       font-size:␣30px;"
                    data-toggle="button" aria-pressed="false"
                       autocomplete="off">
                    Tests
```

```
                </button></a>
            </div>
        </div>
        <div class="row␣justify-content-center">
            <div class="col">
                <a href="/cases" style="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣text-decoration:␣none;color:␣whitesmoke;␣font-
   size:␣30px;">
                <button type="button" class="btn␣btn-secondary"
                    style="width:␣75%;height:␣200px;margin-left:␣
                       12%;␣margin-top:␣25%;margin-bottom:␣50%;␣
                       font-size:␣30px;"
                    data-toggle="button" aria-pressed="false"
                       autocomplete="off">
                    Cases
                </button></a>
            </div>
            <div class="col">
                <a href="/vaccinations" style="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣text-decoration:␣none;color:␣whitesmoke;␣font-
   size:␣30px;">
                <button type="button" class="btn␣btn-secondary"
                    style="width:␣75%;height:␣200px;margin-left:␣
                       12%;␣margin-top:␣25%;margin-bottom:␣50%;␣
                       font-size:␣30px;"
                    data-toggle="button" aria-pressed="false"
                       autocomplete="off">
                    Vaccinations
                </button></a>
            </div>
            <div class="col">
                <a href="/deaths" style="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣text-decoration:␣none;color:␣whitesmoke;␣font-
   size:␣30px;">
                <button type="button" class="btn␣btn-secondary"
                    style="width:␣75%;height:␣200px;margin-left:␣
                       12%;␣margin-top:␣25%;margin-bottom:␣50%;␣
                       font-size:␣30px;"
                    data-toggle="button" aria-pressed="false"
                       autocomplete="off">
                    Deaths
                </button></a>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

3.2.2 Cases - Implemented by Talha Şahin

3.2.2.1 Setup Database Codes

Cases table is created at database and related data imported from csv table and inserted into cases table wit these codes.

Code 3.4: Setup file of Cases

```
import psycopg2
import pandas as pd

def insert_row(cols: list, conn, cursor):
    dataset_df = pd.read_csv("setup/dataset.csv")
    dataset_df = dataset_df.loc[:,cols].drop_duplicates()
    dataset_df = dataset_df[dataset_df.iso_code.apply(lambda row :
       row.split("_")[0]!="OWID")]
    # Sutun sayisi kadar %s ekle
    query = """INSERT INTO Locations VALUES(%(iso_code)s,%(
       continent)s,
                %(location)s,%(population)s,%(aged_65_older)s,
                 %(aged_70_older)s,%(median_age)s)"""
    for idx, row in dataset_df.iterrows():
        insert_dict = dict()
        for col in cols:
            if pd.isna(row[col]):
                insert_dict[col] = None
            else:
                insert_dict[col] = row[col]
        cursor.execute(query, insert_dict)
        conn.commit()

conn = psycopg2.connect(database="postgres",
                        host="localhost",
                        user="postgres",
                        password="1234",
                        port="5432")

query = """DROP TABLE IF EXISTS Locations;"""
cursor = conn.cursor()
cursor.execute(query)
conn.commit()


query = """CREATE TABLE Locations (
    location_id VARCHAR(80) PRIMARY KEY,
    continent VARCHAR(80),
    country VARCHAR(80),
    population BIGINT,
    rate_age_65_older NUMERIC,
    rate_age_70_older NUMERIC,
```

```
    median_age NUMERIC );"""

cursor = conn.cursor()
cursor.execute(query)
conn.commit()
insert_row(["iso_code","continent","location","population",
"aged_65_older","aged_70_older","median_age"], conn, cursor)
conn.close()
```

3.2.2.2 Model- Functions with Queries

Code 3.5: Function creating connection with DB

```
def connect():
        conn = psycopg2.connect(database="postgres",
                                host="localhost",
                                user="postgres",
                                password="1234",
                                port="5432")
        return conn
```

Code 3.6: Function getting data by primary key

```
def findById(id):
        query = """SELECT * FROM CASES C WHERE C.id = %s"""
        connection = cases.connect()
        try:
            cursor = connection.cursor()
            cursor.execute(query, (id,))
            return cursor.fetchone()
        except psycopg2.DatabaseError:
            connection.rollback()
        finally:
            cursor.close()
            connection.close()
```

Code 3.7: Function finding all rows primary key value

```
def findAll():
        query = """SELECT * FROM CASES"""
        connection = cases.connect()
        try:
            cursor = connection.cursor()
            cursor.execute(query)
            return cursor.fetchall()
        except psycopg2.DatabaseError:
            connection.rollback()
        finally:
            cursor.close()
            connection.close()
```

Code 3.8: Function deleting a row by id

```
def delete(id):
        query = """DELETE FROM CASES C WHERE C.id = %s"""
        connection = cases.connect()
        try:
            cursor = connection.cursor()
            cursor.execute(query, (id,))
            connection.commit()
        except psycopg2.DatabaseError:
            connection.rollback()
        finally:
            cursor.close()
            connection.close()
```

Code 3.9: Function inserting a new row to the table

```
def save(location_id, total_cases, new_cases,
   total_cases_per_million, new_cases_per_million,
   new_cases_smoothed_per_million, date_time):

        query = """INSERT INTO CASES(location_id,total_cases,
        new_cases,total_cases_per_million,
        new_cases_per_million,
        new_cases_smoothed_per_million,
        date_time)
        VALUES(%(location_id)s, %(total_cases)s, %(new_cases)s,
        %(total_cases_per_million)s,
        %(new_cases_per_million)s,
        %(new_cases_smoothed_per_million)s, %(date_time)s)"""
        connection = cases.connect()
        try:
            cursor = connection.cursor()
            cursor.execute(query, {
                'location_id': location_id,
                'total_cases': total_cases,
                'new_cases': new_cases,
                'total_cases_per_million': total_cases_per_million,
                'new_cases_per_million': new_cases_per_million,
                'new_cases_smoothed_per_million':
                   new_cases_smoothed_per_million,
                'date_time': date_time
            })
            connection.commit()
            return True
        except psycopg2.DatabaseError:
            connection.rollback()
            return False
        finally:
            cursor.close()
            connection.close()
```

Code 3.10: Function updating row by id

```
def update(id,location_id, total_cases, new_cases,
   total_cases_per_million,
new_cases_per_million, new_cases_smoothed_per_million, date_time):
        query = """UPDATE CASES SET (location_id,total_cases,
           new_cases,
        total_cases_per_million,
        new_cases_per_million, new_cases_smoothed_per_million,
        date_time)
        = (%(location_id)s,%(total_cases)s, %(new_cases)s,%(
           total_cases_per_million)s,
        %(new_cases_per_million)s, %(new_cases_smoothed_per_million
           )s, %(date_time)s)
        WHERE CASES.id = %(id)s"""
        connection = cases.connect()
        try:
            cursor = connection.cursor()
            cursor.execute(query, {
                'id': id,
                'location_id': location_id,
                'total_cases': total_cases,
                'new_cases': new_cases,
                'total_cases_per_million': total_cases_per_million,
                'new_cases_per_million': new_cases_per_million,
                'new_cases_smoothed_per_million':
                   new_cases_smoothed_per_million,
                'date_time': date_time
            })
            connection.commit()
            return True
        except psycopg2.DatabaseError:
            connection.rollback()
            return False
        finally:
            cursor.close()
            connection.close()
```

Code 3.11: Function finding datas by location id

```
def findByLocationId(location_id):
        query = """SELECT * FROM CASES C WHERE C.location_id = %s
           """
        connection = cases.connect()
        try:
            cursor = connection.cursor()
            cursor.execute(query, (location_id,))
            return cursor.fetchone()
        except psycopg2.DatabaseError:
            connection.rollback()
        finally:
            cursor.close()
```

```
            connection.close()
```

Code 3.12: Function getting 100 data by offset

```
def Get100ByOffset(offset):
        query = """SELECT * FROM CASES ORDER BY CASES.id OFFSET %s
           ROWS FETCH FIRST 100 ROW ONLY"""
        connection = cases.connect()
        try:
            cursor = connection.cursor()
            cursor.execute(query, (offset,))
            return cursor.fetchall()
        except psycopg2.DatabaseError:
            connection.rollback()
        finally:
            cursor.close()
            connection.close()
```

Code 3.13: Getting 100 datas by offset and location id

```
def Get100ByOffsetAndCountry(country, offset):
        query = """SELECT * FROM CASES WHERE CASES.location_id = %s
            ORDER BY CASES.id OFFSET %s ROWS FETCH FIRST 100 ROW
           ONLY"""
        connection = cases.connect()
        try:
            cursor = connection.cursor()
            cursor.execute(query, (country,offset,))
            return cursor.fetchall()
        except psycopg2.DatabaseError:
            connection.rollback()
        finally:
            cursor.close()
            connection.close()
```

3.2.2.3 View

Code 3.14: Flask Function of Cases Page

```
def cases_page(id = -1):
    user_id = str(session["id"])
    isAdmin = False
    if user_id is not None and user_id != "None":
         user = User()
         isAdmin = user.isAdmin(user_id)
    else:
        return redirect("/")

    if id != -1 and isAdmin:
        cases.delete(int(id))

    locations = Locations()
    pageNumber = int(request.args.get('page')) if request.args.get(
       'page') is not None else 1
    countryName = request.args.get('loc_name') if request.args.get(
       'loc_name') is not None else "?"

    pageNumber = int(pageNumber)
    offset = (pageNumber-1)*100

    countries = None
    casesData = None
    headings = ["ID", "Location␣Id", "Total␣Cases", "New␣Cases", "
       Total␣Cases␣PM",
                "New␣Cases␣PM", "New␣Cases␣SPM", "Date"]

    countries = locations.get_country_names()
    countriesData = []
    for row in countries:
        countriesData.append(row[0])


    location_id = locations.get_id_by_country_name(country=
       countryName)

    result = None
    if countryName != '?':
        result = cases.Get100ByOffsetAndCountry(country=location_id
           , offset=offset)
    else:
        result = cases.Get100ByOffset(offset=offset)

    casesData = np.zeros([1, 8], dtype='str')
    for row in result:
        newRow = np.array(row)
        casesData = np.vstack([casesData, newRow])
```

```
    casesData = np.delete(casesData, 0, 0)
    return render_template("cases/cases.html", table_headers=
       headings, locations=countriesData, table_rows=casesData,
       isAdmin=isAdmin)
```

Code 3.15: Flask Function of Cases Update Page

```
def update_cases_page(id = -1):
    user_id = str(session["id"])
    isAdmin = False
    if user_id is not None and user_id != "None":
         user = User()
         isAdmin = user.isAdmin(user_id)
    else:
        return redirect("/")

    if isAdmin is False:
        return redirect("/cases")

    message = "empty"
    updateData = ["","","","","","","","","]
    if id != -1:
        updateData = cases.findById(id=id)

    if request.method == "POST":
        if request.form["cases_id"] !="":
            updateData = cases.findById(id=request.form["cases_id"
               ])
        cases_id = request.form["cases_id"]
        location_id = request.form["location_id"] if request.form["
           location_id"] !="" else  updateData[1]
        total_cases = request.form["total_cases"] if request.form["
           total_cases"] !="" else  updateData[2]
        new_cases = request.form["new_cases"] if request.form["
           new_cases"] !="" else  updateData[3]
        total_cases_per_million = request.form["
           total_cases_per_million"] if request.form["
           total_cases_per_million"] !="" else  updateData[4]
        new_cases_per_million = request.form["new_cases_per_million
           "] if request.form["new_cases_per_million"] !="" else
           updateData[5]
        new_cases_smoothed_per_million = request.form["
           new_cases_smoothed_per_million"] if request.form["
           new_cases_smoothed_per_million"] !="" else  updateData
           [6]
        date_time = request.form["date_time"] if request.form["
           date_time"] !="" else  updateData[7]
        result = cases.update(cases_id, location_id, total_cases,
           new_cases, total_cases_per_million,
           new_cases_per_million, new_cases_smoothed_per_million,
           date_time)
        if result:
            message = "success"
        else:
            message = "failed"
```

```
    return render_template("cases/update-cases.html", data=
        updateData, message=message)
```

Code 3.16: Flask Function of Cases Add Page

```
def add_cases_page():
    user_id = str(session["id"])
    isAdmin = False
    if user_id is not None and user_id != "None":
         user = User()
         isAdmin = user.isAdmin(user_id)
    else:
        return redirect("/")

    if isAdmin is False:
        return redirect("/cases")

    message = "empty"

    if request.method == "POST":
        location_id = request.form["location_id"]
        total_cases = request.form["total_cases"]
        new_cases = request.form["new_cases"]
        total_cases_per_million = request.form["
           total_cases_per_million"] if request.form["
           total_cases_per_million"] !="" else "NULL"
        new_cases_per_million = request.form["new_cases_per_million
           "] if request.form["new_cases_per_million"] !="" else "
           NULL"
        new_cases_smoothed_per_million = request.form["
           new_cases_smoothed_per_million"] if request.form["
           new_cases_smoothed_per_million"] !="" else "NULL"
        date_time = request.form["date_time"]
        result = cases.save(location_id, total_cases, new_cases,
           total_cases_per_million, new_cases_per_million,
           new_cases_smoothed_per_million, date_time)
        if result:
            message = "success"
        else:
            message = "failed"

    return render_template("cases/add-cases.html", message=message)
```

3.2.2.4 Html Pages

Code 3.17: HTML form of Cases Page

```
{% extends "after_login.html" %}
{% block title %}Test{% endblock %}
{% block content %}
<script type="text/javascript">
    function goNewDirect(pageToggle, page, loc) {

        locStr = "";
        if (loc != null && loc != '')
            locStr = "&loc_name=" + loc;

        if (page == null) {
            window.location.href = "/cases?page=2" + locStr;
        }
        else {
            if (pageToggle == -1)
                window.location.href = "/cases?page=" + parseInt(
                   page).toString() + locStr;
            if (pageToggle == 0 && parseInt(page) - 1 >= 1)
                window.location.href = "/cases?page=" + (parseInt(
                   page) - 1).toString() + locStr;
            if (pageToggle == 1)
                window.location.href = "/cases?page=" + (parseInt(
                   page) + 1).toString() + locStr;
        }
    }

    function changePage(pageToggle) {
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);

        goNewDirect(pageToggle, urlPage.get('page'), urlPage.get("
           loc_name"));
    }

    function checkIfSelected(element) {
        if (element.value == 'Choose...') {
            alert("Choose␣!!");
            return false;
        }
        return true;
    }

    function filterCountry() {
        let location = document.getElementById("inputGroupSelect01"
           );
        if (checkIfSelected(location) == false)
            return;
```

```
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);

        goNewDirect(-1, 1, location.value);
    }

    function reset(type) {
        document.location.href = '/cases'
    }

    function deleteRow(row) {
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);
        if (row.value != "")
            window.location.href = "/cases/" + row.value + "?" +
               urlPage;
        else
            window.location.href = "/cases?" + urlPage;
    }

</script>
<div class="container">
    <div class="row␣d-flex␣mt-2">
        <div class="input-group␣col">
            <select class="form-select" id="inputGroupSelect01"
               aria-label="Example␣select␣with␣button␣addon">
                <option selected>Choose...</option>
                {% for loc in locations%}
                <option value={{loc}}>{{loc}}</option>
                {% endfor %}
            </select>
            <button class="btn␣btn-outline-secondary" type="button"
                onclick="filterCountry()">Filter</button>
            <button class="btn␣btn-outline-secondary" id="
               resetGroupButton02" onclick="reset('l')"
                type="button">Reset</button>
        </div>
        {% if isAdmin %}
        <div class="col">
            <button class="btn␣btn-outline-secondary" type="button"
                onclick="document.location.href␣=␣'/update-cases'␣"
                   >Update</button>
            <button class="btn␣btn-outline-secondary" type="button"
                onclick="document.location.href␣=␣'/add-cases'␣">
                   Add</button>
        </div>
        {% endif %}
        <div class="col">
            <nav aria-label="Page␣navigation␣example">
                <ul class="pagination␣justify-content-end␣mb-0">
```

```
                    <li class="page-item"><button class="btn␣btn-
                       outline-secondary" style="margin-right:␣5px;
                       "
                            onclick="changePage(0)" type="button">
                               PREV</button></li>
                    <li class="page-item"><button class="btn␣btn-
                       outline-secondary" onclick="changePage(1)"
                            type="button">NEXT</button></li>
                </ul>
            </nav>
        </div>
    </div>
    <div class="row␣my-3␣mx-0">
        <table class="table␣align-middle␣table-striped␣table-hover"
           >
            <thead>
                <tr class="align-bottom">

                    {% for head in table_headers %}
                    <th scope="col">"{{head}}"</th>
                    {% endfor %}
                    <th scope="col"> </th>
                </tr>
            </thead>
            <tbody>
                {% for row in table_rows%}
                <tr class="align-items-center">
                    {% for cell in row[0:] %}
                    <td>
                        <div class="data">
                            {{cell}}
                        </div>
                    </td>
                    {% endfor %}
                    <td>
                        {% if isAdmin %}
                        <div class="d-flex">
                            <div class="data" style="margin-right:␣
                               10px;">
                                <button class="btn␣btn-outline-
                                   secondary" title="DELETE" style=
                                   "background-color:␣red;" href="#
                                   "
                                    id="delete{{␣row[0]␣}}" value="
                                       {{␣row[0]␣}}"
                                    onclick="deleteRow(document.
                                       getElementById('delete{{␣row
                                       [0]␣}}'))" type="button"><i
                                        class="bi␣bi-trash" style="
                                           color:␣white;"></i>
```

```
                                    </button>
                                </div>
                                <div class="data">
                                    <button class="btn␣btn-outline-
                                       secondary" title="UPDATE" id="
                                       delete{{␣row[0]␣}}" value="{{␣
                                       row[0]␣}}"
                                        onclick="document.location.href
                                           ␣=␣'/update-cases/{{␣row[0]␣
                                           }}';" type="button">
                                        <i class="bi␣bi-arrow-clockwise
                                           "></i>
                                    </button>
                                </div>
                            </div>

                            {% endif %}
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
</div>
{% endblock %}
```

Code 3.18: HTML Form of Cases Update Page

```
{% extends "after_login.html" %}
{% block title %}Cases{% endblock %}
{% block content %}
<script type="text/javascript">
    const message = '{{message}}';
    function fetchData(id) {
        if (id != "") {
            window.location.href = "/update-cases/" + id;
            document.getElementById('cases_id').value = id;
        }
        else
            window.alert("Please␣enter␣an␣id␣and␣push␣to␣Fetch␣Data
               ␣buton␣after␣that.");
    }

    if (message != "empty") {
        if (message == "success") {
            window.alert("Data␣successfully␣updated.");
            window.location.href = "/cases";
        }
        else {
            window.alert("Data␣could␣not␣be␣updated.␣Please␣control
               ␣the␣values␣and␣try␣again!.");
            window.location.href = "/update-cases";
        }
    }

</script>
<h2 class="container␣my-4">Update Cases Data By Id</h2>
<div class="container␣my-4">
    <form action="/update-cases" method="POST">
        <div class="mb-3">
            <label for="cases_id" class="form-label">Id - You can
               see old values of data you want to update by
               entering
                id and press the Fetch button.</label>
            <div class="d-flex␣align-items-center">
                <input type="text" name="cases_id" class="form-
                   control" style="width:␣30%;" value="{{data[0]}}"
                    id="cases_id" aria-describedby="emailHelp">
                <button class="btn␣btn-blue␣btn-outline-secondary"
                   style="margin-left:␣10px;␣margin-right:␣10px;"
                    type="button" onclick="fetchData(document.
                       getElementById('cases_id').value)">Fetch</
                       button>
            </div>
        </div>
        <div class="mb-3">
            <label for="location_id" class="form-label">Location Id
```

```
            </label>
        <input type="text" name="location_id" placeholder="{{
            data[1]}}" class="form-control" style="width:␣30%;"
             id="location_id" aria-describedby="emailHelp">

    </div>
    <div class="mb-3">
        <label for="total_cases" class="form-label">Total Cases
            </label>
        <div class="d-flex"> <input type="text" name="
            total_cases" placeholder="{{data[2]}}" class="form-
            control"
                 style="width:␣30%;" id="total_cases" aria-
                    describedby="emailHelp">

        </div>
        <div class="mb-3">
            <label for="new_cases" class="form-label">New Cases
                </label>
            <input type="text" name="new_cases" placeholder="{{
                data[3]}}" class="form-control" style="width:␣
                30%;"
                 id="new_cases" aria-describedby="emailHelp">
        </div>
        <div class="mb-3">
            <label for="total_cases_per_million" class="form-
                label">Total Cases Per Million</label>
            <input type="text" name="total_cases_per_million"
                placeholder="{{data[4]}}" class="form-control"
                 style="width:␣30%;" id="total_cases_per_million
                    " aria-describedby="emailHelp">
        </div>
        <div class="mb-3">
            <label for="new_cases_per_million" class="form-
                label">New Cases Per Million</label>
            <input type="text" name="new_cases_per_million"
                placeholder="{{data[5]}}" class="form-control"
                 style="width:␣30%;" id="new_cases_per_million"
                    aria-describedby="emailHelp">
        </div>
        <div class="mb-3">
            <label for="new_cases_smoothed" class="form-label">
                New Cases Smoothed Per Million</label>
            <input type="text" name="
                new_cases_smoothed_per_million" placeholder="{{
                data[6]}}" class="form-control"
                 style="width:␣30%;" id="
                    new_cases_smoothed_per_million" aria-
                    describedby="emailHelp">
        </div>
```

```
        </div>
        <div class="mb-3">
            <label for="new_cases_smoothed_per_million" class="form
               -label">Date (yyyy-mm-dd)</label>
            <input type="text" name="date_time" placeholder="{{data
               [7]}}" class="form-control" style="width:␣30%;"
                id="date_time" aria-describedby="emailHelp">
        </div>
        <button type="submit" class="btn␣btn-primary">Update Data</
           button>
    </form>
</div>
{% endblock %}
```

Code 3.19: HTML Form of Cases Add Page

```
{% extends "after_login.html" %}
{% block title %}Cases{% endblock %}
{% block content %}
<script type="text/javascript">
    const message = '{{message}}';

    if (message != "empty") {
        if (message == "success") {
            window.alert("Data␣successfully␣added.");
            window.location.href = "/cases";
        }
        else {
            window.alert("Data␣could␣not␣be␣added.␣Please␣control␣
               the␣values␣and␣try␣again!.");
            window.location.href = "/add-cases";
        }
    }

</script>
<h2 class="container␣my-4">Add Cases Data</h2>
<div class="container␣my-4">
    <form action="/add-cases" method="POST">
        <div class="mb-3">
            <label for="location_id" class="form-label">Location Id
               </label>
            <input type="text" name="location_id" class="form-
               control" id="location_id" aria-describedby="
               emailHelp">
        </div>
        <div class="mb-3">
            <label for="total_cases" class="form-label">Total Cases
               </label>
            <input type="text" name="total_cases" class="form-
               control" id="total_cases" aria-describedby="
               emailHelp">
        </div>
        <div class="mb-3">
            <label for="new_cases" class="form-label">New Cases</
               label>
            <input type="text" name="new_cases" class="form-control
               " id="new_cases" aria-describedby="emailHelp">
        </div>
        <div class="mb-3">
            <label for="total_cases_per_million" class="form-label"
               >Total Cases Per Million</label>
            <input type="text" name="total_cases_per_million" class
               ="form-control" id="total_cases_per_million"
                aria-describedby="emailHelp">
        </div>
```

```
        <div class="mb-3">
            <label for="new_cases_per_million" class="form-label">
               New Cases Per Million</label>
            <input type="text" name="new_cases_per_million" class="
               form-control" id="new_cases_per_million"
                aria-describedby="emailHelp">
        </div>
        <div class="mb-3">
            <label for="new_cases_smoothed" class="form-label">New
               Cases Smoothed Per Million</label>
            <input type="text" name="new_cases_smoothed_per_million
               " class="form-control"
                id="new_cases_smoothed_per_million" aria-
                   describedby="emailHelp">
        </div>
        <div class="mb-3">
            <label for="new_cases_smoothed_per_million" class="form
               -label">Date (yyyy-mm-dd)</label>
            <input type="text" name="date_time" class="form-control
               " id="date_time" aria-describedby="emailHelp">
        </div>
        <button type="submit" class="btn␣btn-primary">Add Data</
           button>
    </form>
</div>
{% endblock %}
```

3.2.3 Deaths - Implemented by Zehra Asan

3.2.3.1 Setup Vaccinations Table

To create vaccinations table in database following code is used. This code reads data values according to the names of columns and add into the Vaccinations

Code 3.20: Setup Deaths in Database

```
import pandas as pd
import psycopg2

def insert_row(cols: list, conn, cursor):
    dataset_df = pd.read_csv("setup/dataset.csv")
    dataset_df = dataset_df.loc[:,cols].drop_duplicates()
    dataset_df = dataset_df[dataset_df.iso_code.apply(lambda row :
       row.split("_")[0]!="OWID")].dropna()
    query = """INSERT INTO DEATHS(location_id,total_deaths,
       new_deaths,new_deaths_smoothed,total_deaths_per_million,
       new_deaths_per_million,new_deaths_smoothed_per_million,
       date_time)
    VALUES(%(iso_code)s,%(total_deaths)s,%(new_deaths)s,%(
       new_deaths_smoothed)s,%(total_deaths_per_million)s,%(
       new_deaths_per_million)s,%(new_deaths_smoothed_per_million)s
       , %(date)s)"""
    for idx, row in dataset_df.iterrows():
        insert_dict = dict()
        for col in cols:
            if pd.isna(row[col]):
                insert_dict[col] = None
            else:
                insert_dict[col] = row[col]
        cursor.execute(query, insert_dict)
        conn.commit()

conn = psycopg2.connect(database="postgres",
                        host="localhost",
                        user="postgres",
                        password="1234",
                        port="5432")
cursor = conn.cursor()

queryTable = """CREATE TABLE DEATHS (
    id SERIAL PRIMARY KEY,
    location_id VARCHAR(80) REFERENCES locations(location_id),
    total_deaths INTEGER,
    new_deaths INTEGER,
    new_deaths_smoothed NUMERIC,
    total_deaths_per_million NUMERIC,
    new_deaths_per_million NUMERIC,
    new_deaths_smoothed_per_million NUMERIC,
```

```
    date_time DATE
);"""

cursor.execute(queryTable)
conn.commit()
insert_row(["iso_code","total_deaths","new_deaths","
   new_deaths_smoothed","total_deaths_per_million", "
   new_deaths_per_million", "new_deaths_smoothed_per_million","date
   "], conn, cursor)
conn.close()
```

3.2.3.2 Model Deaths

Code 3.21: Model for Deaths

```
import psycopg2 as ps
import numpy as np
from datetime import datetime
#Zehra's table --Deaths--
#Operation Functions

#Constructed --> connects to the db
#Destructed --> closes connection

class Deaths:
    #Constructor to connect -initalizer-
    def __init__(self):
        self.columns = "['location_id','total_deaths','new_deaths
           ','new_deaths_smoothed',\
␣␣␣␣␣␣␣␣'total_deaths_per_million','new_deaths_per_million','
   new_deaths_smoothed_per_million','date_time']"
        self.connection = None
        self.cursor = None
        self.connect()

    #Deconstructor to disconnect
    def __del__(self):
        try:
            self.connection.close()
        except:
            pass

    #Database connection
    def connect(self):
        self.connection = ps.connect(database="postgres",
                        host="localhost",
                        user="postgres",
                        password="1234",
                        port="5432")
```

```
    #Check connection
    def check_connection(self):
        try:
            self.connection.status
        except:
            self.connect()

    #Read by id(will be used if necessary)
    def read_with_id(self, id):
        query = """SELECT * FROM DEATHS AS d WHERE d.id = %s ORDER
           BY d.id;"""
        self.check_connection()
        try:
            self.cursor = self.connection.cursor()
            self.cursor.execute(query, (id,))
            return self.cursor.fetchone()
        except ps.DatabaseError:
            self.connection.rollback()
        finally:
            self.cursor.close()

    #Read all
    def readAll(self):
        query = """SELECT * FROM DEATHS"""
        self.check_connection()
        try:
            self.cursor = self.connection.cursor()
            self.cursor.execute(query, (id,))
            return self.cursor.fetchone()
        except ps.DatabaseError:
            self.connection.rollback()
        finally:
            self.cursor.close()
            self.connection.close()

    #Insert new row
    def insert_row(self, location_id,total_deaths,new_deaths,
       new_deaths_smoothed,total_deaths_per_million,\
                 new_deaths_per_million,
                    new_deaths_smoothed_per_million, date_time):

        query = """INSERT INTO DEATHS(location_id, total_deaths,
           new_deaths_smoothed,new_deaths,
        total_deaths_per_million,new_deaths_per_million,
           new_deaths_smoothed_per_million,date_time)
        VALUES(%(location_id)s,%(total_deaths)s,%(
           new_deaths_smoothed)s,%(new_deaths)s,%(
           total_deaths_per_million)s,
        %(new_deaths_per_million)s, %(
           new_deaths_smoothed_per_million)s, %(date_time)s)"""
```

```
        self.check_connection()
        try:
            self.cursor = self.connection.cursor()
            self.cursor.execute(query, {
                'location_id': location_id,
                'total_deaths': total_deaths,
                'new_deaths_smoothed': new_deaths_smoothed,
                'total_deaths_per_million':
                   total_deaths_per_million,
                'new_deaths_per_million': new_deaths_per_million,
                'new_deaths_smoothed_per_million':
                   new_deaths_smoothed_per_million,
                'new_deaths': new_deaths,
                'date_time': date_time
            })
            self.connection.commit()
            return True
        except ps.DatabaseError:
            self.connection.rollback()
            return False
        finally:
            self.cursor.close()
            self.connection.close()

    #Update a row by using id
    def update_row(self, id,location_id,total_deaths,new_deaths,
       new_deaths_smoothed,total_deaths_per_million,\
                 new_deaths_per_million,
                    new_deaths_smoothed_per_million, date_time):

        query = """UPDATE DEATHS SET(location_id, total_deaths,
           new_deaths_smoothed,new_deaths,
        total_deaths_per_million,new_deaths_per_million,
           new_deaths_smoothed_per_million,date_time)
        = (%(location_id)s,%(total_deaths)s,%(new_deaths_smoothed)s
           ,%(new_deaths)s,%(total_deaths_per_million)s,
        %(new_deaths_per_million)s, %(
           new_deaths_smoothed_per_million)s, %(date_time)s) WHERE
           DEATHS.id = %(id)s"""
        self.check_connection()
        try:
            self.cursor = self.connection.cursor()
            self.cursor.execute(query, {
                'id': id,
                'location_id': location_id,
                'total_deaths': total_deaths,
                'new_deaths_smoothed': new_deaths_smoothed,
                'total_deaths_per_million':
                   total_deaths_per_million,
                'new_deaths_per_million': new_deaths_per_million,
```

```
            'new_deaths_smoothed_per_million':
               new_deaths_smoothed_per_million,
            'new_deaths': new_deaths,
            'date_time': date_time
        })
        self.connection.commit()
        return True
    except ps.DatabaseError:
        self.connection.rollback()
        return False
    finally:
        self.cursor.close()
        self.connection.close()

#Delete row by id
def delete(self, id):
    query = """DELETE FROM DEATHS AS d WHERE d.id = %s"""
    self.check_connection()
    try:
        self.cursor = self.connection.cursor()
        self.cursor.execute(query, (id,))
        self.connection.commit()
        return True
    except ps.DatabaseError:
        self.connection.rollback()
        return False
    finally:
        self.cursor.close()


def get_location_names(self):
    loc_names = np.array(self.query_location_names())
    loc_count = loc_names.shape[0]
    return [name.replace("␣","_") for name in loc_names.reshape
       (-1,loc_count)[0] if name is not None]

def query_location_names(self):
    query = """SELECT DISTINCT country FROM DEATHS AS dt
            LEFT JOIN LOCATIONS AS L
            ON dt.location_id = L.location_id
            ORDER BY country ASC"""
    self.check_connection()
    try:
        self.cursor = self.connection.cursor()
        self.cursor.execute(query)
        return self.cursor.fetchall()
    except ps.DatabaseError:
        self.connection.rollback()
    finally:
        self.cursor.close()
```

```
def read_filter(self, limit=-1, offset=0, loc_name="?", date="?
   "):
    date_str = ""
    query = "SELECT␣*␣FROM␣DEATHS␣AS␣dt"
    if loc_name!="?":
        query += "␣LEFT␣JOIN␣LOCATIONS␣AS␣L␣ON␣dt.location_id␣=
           ␣L.location_id␣WHERE␣country=%(loc_name)s"
    if date!="?":
        date_str = "␣date_time␣>=␣TO_DATE(%(date)s,'YYYY-MM-DD
           ')"
    if loc_name=="?" and date!="?":
        query += "␣WHERE" + date_str
    elif loc_name!="?" and date!="?":
        query += "␣AND" + date_str

    query += "␣ORDER␣BY␣dt.id␣OFFSET␣%(offset)s"
    if limit != -1:
        query += "␣LIMIT␣" + str(limit)
    query += ";"
    self.check_connection()
    try:
        self.cursor = self.connection.cursor()
        self.cursor.execute(query, {"offset":str(offset),
                                    "loc_name":loc_name,
                                    "date":date})
        return self.cursor.fetchall()
    except ps.DatabaseError:
        self.connection.rollback()
    finally:
        self.cursor.close()



def get_dates(self,loc,start=-1):
    dates = np.array(self.query_dates(loc,start))
    date_count = dates.shape[0]
    dates_list = dates.reshape(-1,date_count)[0]
    return [date.strftime('%Y-%m-%d') for date in dates_list if
        date is not None]


def query_dates(self,loc,start):
    loc_str = ""
    start_str = ""
    query = "SELECT␣DISTINCT␣date_time␣FROM␣DEATHS␣AS␣dt"
    if loc != "?":
        loc_str = "␣LEFT␣JOIN␣LOCATIONS␣AS␣L␣ON␣dt.location_id␣
           =␣L.location_id␣WHERE␣country␣=␣%(loc_str)s"
```

```
        if start != -1:
            start = datetime.strptime(start,'%Y-%m-%d')
            start_str = "␣date_time␣>␣%(start)s"
        query += loc_str
        if loc == "?" and start != -1:
            query += "␣WHERE"
        elif loc != "?" and start != -1:
            query += "␣AND"
        query += start_str
        query += "␣ORDER␣BY␣date_time␣ASC;"
        self.check_connection()
        try:
            self.cursor = self.connection.cursor()
            self.cursor.execute(query,{'loc_str':loc, 'start':start
               })
            return self.cursor.fetchall()
        except ps.DatabaseError:
            self.connection.rollback()
        finally:
            self.cursor.close()
```

3.2.3.3 View Deaths

Code 3.22: View Functions for Pages of Deaths

```
from flask import render_template, request, session, redirect
import numpy as np

from model.deaths import *
from model.user import *

def deaths_page(id = -1):
    user_id = str(session["id"])
    is_admin = False
    if user_id is not None:
        user = User()
        is_admin = user.isAdmin(user_id)

    if user_id is not None and user_id != "None":
         user = User()
         isadmin = user.isAdmin(user_id)
    else:
        return redirect("/")

    page_id = request.args.get('page') if request.args.get('page')
       is not None else 1
    loc_name = request.args.get('loc_name') if request.args.get('
       loc_name') is not None else "?"
```

```
    date = request.args.get('date') if request.args.get('date') is
       not None else "?"
    table_size = 0

    page_id = int(page_id)

    deaths = Deaths()
    if id != -1:
        deaths.delete(int(id))

    loc_name = loc_name.replace("_","␣")

    loc_names = deaths.get_location_names()
    offset = (page_id-1)*50
    paginationValues = (page_id-1,page_id,page_id+1) if (page_id)>1
        else (1,2,3)


    try:
        covid_data = np.array(deaths.read_filter(50,offset,loc_name
           ,date))[:,0:9]
        table_size = covid_data.size
    except IndexError:
        covid_data = np.array([[]])
        table_size = 0

    start_dates = deaths.get_dates(loc_name)
    headers = ["Location␣Id", "Total␣Deaths", "New␣Deaths", "New␣
       Deaths␣Smoothed",
             "Total␣Deaths␣Per␣Million", "New␣Deaths␣Per␣Million
                ", "New␣Deaths␣Smoothed␣Per␣Million", "Date"]

    return render_template("deaths/deaths.html", table_headers=
       headers, table_rows = covid_data, \
        paginationValues=paginationValues, locations = loc_names,
           dates = start_dates, data_available=table_size, is_admin
           =is_admin)

def add_deaths_page():
    deaths = Deaths()
    message = "empty"

    if request.method == "POST":
        location_id = request.form["location_id"]
        total_deaths = request.form["total_deaths"]
        new_deaths_smoothed = request.form["new_deaths"]
        total_deaths_per_million = request.form["
           new_deaths_smoothed"] if request.form["
           new_deaths_smoothed"] !="" else None
```

```
        new_deaths_per_million = request.form["
           total_deaths_per_million"] if request.form["
           total_deaths_per_million"] !="" else None
        new_deaths_smoothed_per_million = request.form["
           new_deaths_per_million"] if request.form["
           new_deaths_per_million"] !="" else None
        new_deaths = request.form["new_deaths_smoothed_per_million"
           ] if request.form["new_deaths_smoothed_per_million"] !="
           " else None
        date_time = request.form["date_time"] if request.form["
           date_time"] !="" else None
        result = deaths.insert_row(location_id, total_deaths,
           new_deaths, new_deaths_smoothed,
           total_deaths_per_million, new_deaths_per_million,
           new_deaths_smoothed_per_million, date_time)
        if result:
            message = "success"
        else:
            message = "failed"
    return render_template("deaths/add-deaths.html", message=
       message)

def update_deaths_page():

    row_id = request.args.get('id')
    row_id = int(row_id)
    deaths = Deaths()
    row = np.array(deaths.read_with_id(row_id))
    message = "empty"
    if request.method == "POST":
        total_deaths = request.form["total_deaths"] if request.form
           ["total_deaths"] !="" else row[2]
        new_deaths_smoothed = request.form["new_deaths"] if request
           .form["new_deaths"] !="" else row[3]
        total_deaths_per_million = request.form["
           new_deaths_smoothed"] if request.form["
           new_deaths_smoothed"] !="" else row[4]
        new_deaths_per_million = request.form["
           total_deaths_per_million"] if request.form["
           total_deaths_per_million"] !="" else row[5]
        new_deaths_smoothed_per_million = request.form["
           new_deaths_per_million"] if request.form["
           new_deaths_per_million"] !="" else row[6]
        new_deaths = request.form["new_deaths_smoothed_per_million"
           ] if request.form["new_deaths_smoothed_per_million"] !="
           " else row[7]
        date_time = request.form["date_time"] if request.form["
           date_time"] !="" else row[8]
        result = deaths.update_row(row_id, row[1], total_deaths,
           new_deaths_smoothed, total_deaths_per_million,
```

```
            new_deaths_per_million, new_deaths_smoothed_per_million,
             new_deaths, date_time)
        if result:
            message = "success"
        else:
            message = "failed"

    return render_template("deaths/update-deaths.html", id = row_id
       , data=row, message=message)
```

3.2.3.4 HTMLs of Deaths pages

Code 3.23: Functions for Deaths Page

Deaths Page

```
    function goNewDirect(page, loc, date)
    {
        locStr = "";
        dateStr = "";
        if(loc != null && loc != '')
            locStr = "&loc_name="+loc;
        if(date != null && date != '')
            dateStr = "&date="+date;

        if(page == null)
        {
            window.location.href = "/deaths?page=1"+locStr+dateStr;
        }
        else
        {
            window.location.href = "/deaths?page="+page+locStr+
               dateStr;
        }
    }
    function changePage(pageToggle, data_available)
    {
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);
        page = urlPage.get("page");
        if(urlPage.get("page") == null || (urlPage.get("page") == '
           1' && pageToggle == 0))
            page = (1).toString();
        else
        {
            if(pageToggle == 0)
                page = (parseInt(urlPage.get("page")) - 1).toString
                   ();
            if(pageToggle == 1)
            {
                if(parseInt(data_available) < 50)
```

```
                window.alert("No␣more␣data.");
            else
                page = (parseInt(urlPage.get("page")) + 1).
                   toString();
        }
        if(pageToggle == -1)
            page = urlPage.get("page");
    }
    goNewDirect(page, urlPage.get("loc_name"), urlPage.get("
       date"));
}

function checkIfSelected(element)
{
    if(element.value == 'Choose...')
    {
        if(element.id == "inputGroupSelect01")
            alert("You␣didn't␣choose␣a␣country␣!!");
        if(element.id == "inputGroupSelect02")
            alert("You␣didn't␣choose␣a␣date␣!!");
        return false;
    }
    return true;
}
function filterCountry()
{
    let location = document.getElementById("inputGroupSelect01"
       );
    if(checkIfSelected(location) == false)
        return;
    const urlStr = window.location.search;
    let urlPage = new URLSearchParams(urlStr);
    goNewDirect(null, location.value, urlPage.get("date"));
}
function filterDates(button)
{
    if(checkIfSelected(button) == false)
        return;
    const urlStr = window.location.search;
    let urlPage = new URLSearchParams(urlStr);

    goNewDirect(null, urlPage.get("loc_name"), button.value);
}
function reset(type)
{
    const urlStr = window.location.search;
    let urlPage = new URLSearchParams(urlStr);
    if(type == 'l')
        goNewDirect(urlPage.get("page"), '', urlPage.get("date"
           ));
```

```
        if(type == 's')
            goNewDirect(urlPage.get("page"), urlPage.get("loc_name"
               ), '');
    }
    function deleteRow(row)
    {
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);
        if(row.value != "")
            window.location.href = "/deaths/"+row.value+"?"+urlPage
               ;
        else
        window.location.href = "/deaths?"+urlPage;
    }
    function updateRow(row)
    {
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);
        if(row.value == "")
            window.location.href = "/deaths/"+row.value+"?"+urlPage
               ;
        else
            window.location.href = "/update-deaths?id="+row.value;
```

Code 3.24: Deaths Page

```
<div class="container">
    <div class="row␣d-flex␣mt-2">
        <label for="inputGroupSelect01" class="form-country">
            <h6 style="display:␣inline">Choose a country: </h6>
            <p style="display:␣inline" id="form-country">
                <script>
                    const urlStrCt = window.location.search;
                    let urlPageCt = new URLSearchParams(urlStrCt);
                    if(urlPageCt.get("loc_name") == null)
                        document.getElementById("form-country").
                           textContent = "";
                    else
                        document.getElementById("form-country").
                           textContent = "Selected␣country␣>␣"+
                           urlPageCt.get("loc_name");
                </script>
            </p>
        </label>
        <div class="input-group␣col␣mt-3">
            <select class="form-select" id="inputGroupSelect01"
               aria-label="Example␣select␣with␣button␣addon">
              <option selected>Choose...</option>
            {% for loc in locations%}
               <option value={{loc}}>{{loc}}</option>
            {% endfor %}
```

```
            </select>
            <button style="width:␣70px;justify-content:␣center;"
               class="btn␣btn-outline-secondary␣" type="button"
               onclick="filterCountry()">Filter</button>
            <button style="width:␣70px;justify-content:␣center;"
               class="btn␣btn-outline-secondary" id="
               resetGroupButton02" onclick="reset('l')"
             type="button">Reset</button>
        </div>

        <div class="col">
            {% if is_admin %}
            <nav aria-label="Page␣navigation␣example">
                <ul class="pagination␣justify-content-end␣mb-0">
                    <li class="page-item"><button style="width:␣132
                       px;" class="btn␣btn-warning" type="button"
                        onclick="document.location.href␣=␣'/add-
                           deaths'␣">Add</button></li>
                </ul>
            </nav>
            {% endif %}
        </div>
    </div>

    <div class="row␣d-flex␣mt-3">
        <label for="inputGroupSelect02" class="form-label">
            <h6 style="display:␣inline;">Choose a date: </h6>
            <p style="display:␣inline;" id="form-date">
                <script>
                    const urlStrDate = window.location.search;
                    let urlPageDate = new URLSearchParams(
                       urlStrDate);
                    if(urlPageDate.get("date") == null)
                        document.getElementById("form-date").
                           textContent = "";
                    else
                        document.getElementById("form-date").
                           textContent = "Selected␣date␣>␣"+
                           urlPageDate.get("date");
                </script>
            </p>
        </label>
        <div class="input-group␣col␣mt-1">
            <select class="form-select" id="inputGroupSelect02"
               aria-label="Example␣select␣with␣button␣addon">
              <option selected>Choose...</option>
            {% for date in dates %}
               <option value={{date}}>{{date}}</option>
            {% endfor %}
            </select>
```

```
            <button style="width:␣70px;justify-content:␣center;"
               class="btn␣btn-outline-secondary" id="
               inputGroupButton02" onclick="filterDates(document.
               getElementById('inputGroupSelect02'))"
             type="button">Filter</button>
             <button style="width:␣70px;justify-content:␣center;"
                class="btn␣btn-outline-secondary" id="
                resetGroupButton02" onclick="reset('s')"
             type="button">Reset</button>
        </div>

        <div class="col">
            <nav aria-label="Page␣navigation␣example">
                <ul class="pagination␣justify-content-end␣mb-0">
                    <li class="page-item"><button class="btn␣btn-
                       outline-secondary" id="prev-button" value="
                       {{data_available}}" style="margin-right:␣5px
                       ;"
                            onclick="changePage(0,document.
                               getElementById('prev-button').value)
                               " type="button">PREV</button></li>
                    <li class="page-item"><button class="btn␣btn-
                       outline-secondary"  id="next-button" value="
                       {{data_available}}"
                        onclick="changePage(1,document.
                           getElementById('next-button').value)"
                           type="button">NEXT</button></li>
                </ul>
            </nav>
        </div>
    </div>
    <div class="row␣my-3␣mx-0">
        <table class="table␣align-middle␣table-striped␣table-hover␣
           text-center" id="data-table">
            <thead>
              <tr class="align-bottom">

                {% for head in table_headers %}
                <th scope="col">{{head}}</th>
                {% endfor %}
                {% if is_admin %}
                    <th scope="col" class="edit-buttons">Delete</th
                       >
                    <th scope="col" class="edit-buttons">Update</th
                       >
                {% endif %}
              </tr>
            </thead>
            <tbody>
                {% if data_available %}
```

```
                        {% for row in table_rows%}
                        <tr class="align-items-center">
                            {% for cell in row[1:] %}
                            <td class="align-items-center">
                                <div class="data">
                                    {{cell}}
                                </div>
                            </td>
                            {% endfor %}
                            {% if is_admin %}
                            <td class="edit-buttons">
                                <div class="data">
                                    <button class="btn␣btn-outline-
                                       secondary" style="background-
                                       color:␣red;" href="#"
                                            id="delete{{␣row[0]␣}}"
                                               value="{{␣row[0]␣}}"
                                            onclick="deleteRow(document
                                               .getElementById('delete
                                               {{␣row[0]␣}}'))" type="
                                               button"><i
                                            class="bi␣bi-trash" style="
                                               color:␣white;"></i>
                                    </button>
                                </div>
                            </td>
                            <td class="edit-buttons">
                                <div class="data">
                                    <button class="btn␣btn-outline-
                                       secondary" type="button"
                                    href="#" id="update{{␣row[0]␣}}"
                                       value="{{␣row[0]␣}}"
                                    onclick="updateRow(document.
                                       getElementById('update{{␣row[0]␣
                                       }}'))" "><i
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣class="bi-arrow-right"␣style="color
   : blue;"></i>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</button>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</div>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</td>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣{%␣endif␣%}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</tr>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣{%␣endfor␣%}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣{%␣endif␣%}
␣␣␣␣␣␣␣␣␣␣␣</tbody>
␣␣␣␣␣␣␣␣</table>
␣␣␣␣</div>
</div>
```

Update Deaths Page Functions

Code 3.25: Functions for Update Deaths

```
     const message = '{{message}}';
    if (message != "empty") {
        if (message == "success") {
            window.alert("Data␣successfully␣updated.");
            window.location.href = "/deaths";
        }
        else {
            window.alert("Data␣could␣not␣be␣updated.␣Please␣control
               ␣the␣values␣and␣try␣again!.");
            const urlStr = window.location.search;
            let urlPage = new URLSearchParams(urlStr);
            window.location.href = "/update-deaths?id="+urlPage.get
               ("id");
        }
    }
```

Code 3.26: Update Deaths Page

```
<h2 class="container␣my-4">Update Test Data</h2>
<div class="container␣my-4">
    <form action="/update-deaths?id={{id}}" method="POST">
        <div class="mb-3">
            <label for="total_deaths" class="form-label">Total
               deaths</label>
            <div class="d-flex">
                <input type="text" name="total_deaths" class="form-
                   control" style="width:␣30%;" id="total_deaths"
                    aria-describedby="emailHelp" placeholder="{{
                       data[2]}}">
                </div>
        </div>
        <div class="mb-3">
            <label for="new_deaths" class="form-label">New deaths</
               label>
            <div class="d-flex">
                <input type="text" name="new_deaths" class="form-
                   control" style="width:␣30%;" id="new_deaths"
                    aria-describedby="emailHelp" placeholder="{{
                       data[3]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="total_deaths_per_million" class="form-label
               ">Total deaths Per Million</label>
            <div class="d-flex">
                <input type="text" name="total_deaths_per_million"
                   class="form-control" style="width:␣30%;"
                    id="total_deaths_per_million" aria-describedby=
                       "emailHelp" placeholder="{{data[4]}}">
            </div>
```

```
        </div>
        <div class="mb-3">
            <label for="new_deaths_per_million" class="form-label">
               New deaths Per Million</label>
            <div class="d-flex">
                <input type="text" name="new_deaths_per_million"
                   class="form-control" style="width:␣30%;"
                    id="new_deaths_per_million" aria-describedby="
                       emailHelp" placeholder="{{data[5]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="new_deaths_smoothed" class="form-label">New
                deaths Smoothed</label>
            <div class="d-flex">
                <input type="text" name="new_deaths_smoothed" class
                   ="form-control" style="width:␣30%;"
                    id="new_deaths_smoothed" aria-describedby="
                       emailHelp" placeholder="{{data[6]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="new_deaths_smoothed_per_million" class="
               form-label">New Deaths Smoothed Per Million</label>
            <div class="d-flex">
                <input type="text" name="
                   new_deaths_smoothed_per_million" class="form-
                   control" style="width:␣30%;" id="
                   new_deaths_smoothed_per_million"
                    aria-describedby="emailHelp" placeholder="{{
                       data[7]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="date_time" class="form-label">Date (yyyy-mm
               -dd)</label>
            <div class="d-flex">
                <input type="text" name="date_time" class="form-
                   control" style="width:␣30%;" id="date_time"
                    aria-describedby="emailHelp" placeholder="{{
                       data[8]}}">
            </div>
        </div>
        <button type="submit" class="btn␣btn-primary">Update Data</
           button>
        <button type="button" href="#" onclick="document.location.
           href=␣'/deaths'" class="btn␣btn-primary">Back</button>
    </form>
</div>
```

Add Deaths Page Functions

Code 3.27: Functions for Add Deaths Page

```
    const message = '{{message}}';

    if (message != "empty") {
        if (message == "success") {
            window.alert("Data␣successfully␣added.");
            window.location.href = "/deaths";
        }
        else {
            window.alert("Data␣could␣not␣be␣added.␣Please␣control␣
               the␣values␣and␣try␣again!.");
            window.location.href = "/add-deaths";
        }
    }
```

Code 3.28: Add Deaths Page

```
<h2 class="container␣my-4">Add deaths Data</h2>
<div class="container␣my-4">
    <form action="/add-deaths" method="POST">
        <div class="mb-3">
            <label for="location_id" class="form-label">Location Id
               </label>
            <div class="d-flex">
                <input type="text" name="location_id" class="form-
                   control" style="width:␣30%;" id="location_id"
                    aria-describedby="emailHelp">
            </div>
        </div>
        <div class="mb-3">
            <label for="total_deaths" class="form-label">Total
               Deaths</label>
            <div class="d-flex">
                <input type="text" name="total_deaths" class="form-
                   control" style="width:␣30%;" id="total_deaths"
                    aria-describedby="emailHelp">
                </div>
        </div>
        <div class="mb-3">
            <label for="new_deaths" class="form-label">New Deaths</
               label>
            <div class="d-flex">
                <input type="text" name="new_deaths" class="form-
                   control" style="width:␣30%;" id="new_deaths"
                    aria-describedby="emailHelp">
            </div>
        </div>
        <div class="mb-3">
            <label for="new_deaths_smoothed_per_million" class="
               form-label">New Deaths Smoothed Per Million</label>
            <div class="d-flex">
```

```
            <input type="text" name="
               new_deaths_smoothed_per_million" class="form-
               control" style="width:␣30%;"
                id="new_deaths_smoothed_per_million" aria-
                   describedby="emailHelp">
        </div>
    </div>
    <div class="mb-3">
        <label for="new_deaths_per_million" class="form-label">
           New Deaths Per Million</label>
        <div class="d-flex">
            <input type="text" name="new_deaths_per_million"
               class="form-control" style="width:␣30%;"
                id="new_deaths_per_million" aria-describedby="
                   emailHelp">
        </div>
    </div>
    <div class="mb-3">
        <label for="total_deaths_per_million" class="form-label
           ">Total Deaths Per Million</label>
        <div class="d-flex">
            <input type="text" name="total_deaths_per_million"
               class="form-control" style="width:␣30%;"
                id="total_deaths_per_million" aria-describedby=
                   "emailHelp">
        </div>
    </div>
    <div class="mb-3">
        <label for="new_deaths_smoothed" class="form-label">New
            Deaths Smoothed</label>
        <div class="d-flex">
            <input type="text" name="new_deaths_smoothed" class
               ="form-control" style="width:␣30%;" id="
               new_deaths_smoothed"
                aria-describedby="emailHelp">
        </div>
    </div>
    <div class="mb-3">
        <label for="date_time" class="form-label">Date (yyyy-mm
           -dd)</label>
        <div class="d-flex">
            <input type="text" name="date_time" class="form-
               control" style="width:␣30%;" id="date_time"
                aria-describedby="emailHelp">
        </div>
    </div>
    <button type="submit" class="btn␣btn-primary">Update Data</
       button>
    <button type="button" href="#" onclick="document.location.
       href=␣'/deaths'" class="btn␣btn-primary">Back</button>
```

```
    </form>
</div>
```

3.2.4 Tests - Implemented by Mert Arabacı

3.2.4.1 Setup Covid Tests Codes

Query that creates Covid Tests table

Code 3.29: Create Query of Covid Tests table

```
conn = psycopg2.connect(database="postgres",
                        host="localhost",
                        user="postgres",
                        password="1234",
                        port="5432")
queryTable = """CREATE TABLE COVID_TESTS (
    id SERIAL PRIMARY KEY,
    location_id VARCHAR(80) REFERENCES locations(location_id),
    total_tests BIGINT,
    new_tests NUMERIC,
    total_tests_per_thousand NUMERIC,
    new_tests_per_thousand NUMERIC,
    new_tests_smoothed NUMERIC,
    positive_rate NUMERIC,
    date_time DATE
);"""
cursor = conn.cursor()
cursor.execute(queryTable)
conn.commit()
```

Covid tests part of our dataset is selected and inserted into Covid Tests table.

Code 3.30: Insert to Empty Table Function of Covid Tests table

```
def insert_row(cols: list, conn, cursor):
    dataset_df = pd.read_csv("setup/dataset.csv")
    dataset_df = dataset_df.loc[:,cols].drop_duplicates()
    dataset_df = dataset_df[dataset_df.iso_code.apply(lambda row :
       row.split("_")[0]!="OWID")].dropna()
    # S tun  s a y  s  kadar %s ekle
    query = """INSERT INTO COVID_TESTS(location_id, total_tests,
       new_tests,total_tests_per_thousand,new_tests_per_thousand,
       new_tests_smoothed,positive_rate,date_time)
                VALUES(%(iso_code)s,%(total_tests)s,
                                        %(new_tests)s,%(
                                           total_tests_per_thousand
                                           )s,%(
                                           new_tests_per_thousand)s
                                           , %(new_tests_smoothed)s
```

```
                                                , %(positive_rate)s, %(
                                                date)s)"""
    for idx, row in dataset_df.iterrows():
        insert_dict = dict()
        for col in cols:
            if pd.isna(row[col]):
                insert_dict[col] = None
            else:
                insert_dict[col] = row[col]
        cursor.execute(query, insert_dict)
        conn.commit()
```

3.2.4.2 Class Structure of Covid Tests Table

Code 3.31: CovidTests Class

```
class CovidTests:

    # Initialize object and connect to database
    def __init__(self):
        self.columns = ["location_id", "total_tests", "new_tests",
           "total_tests_per_thousand", "new_tests_per_thousand", "
           new_tests_smoothed", "positive_rate", "date_time"]
        self.conn = None
        self.connect()
        self.cusor = None

    # Close connection to the database and destruct
    def __del__(self):
        try:
            self.conn.close()
        except:
            pass
    # Connect to the database
    def connect(self)

    # Check connection and connect again if it is closed
    def check_conn(self):
        try:
            self.conn.status
        except:
            self.connect()

    def get_dates(self,loc,start=-1)

    def query_dates(self,loc,start)

    def get_location_names(self)
```

```
    def query_location_names(self)

    # Read a row by id
    def read_by_id(self, id)

    def read_filter(self, limit=-1, offset=0, loc_name="?", date="?
       ")

    # Insert a row into table
    def insert_row(self, location_id, total_tests, new_tests,
       total_tests_per_thousand, \
        new_tests_per_thousand,new_tests_smoothed,positive_rate,
           date_time)

    #Update a row by id
    def update(self, id, location_id, total_tests, new_tests,
       total_tests_per_thousand, \
        new_tests_per_thousand,new_tests_smoothed,positive_rate,
           date_time)

    # Delete a row by id
    def delete(self, id)
```

3.2.4.3 Read Functions of Covid Tests Table

Code 3.32: Read Dates Functions

```
def get_dates(self,loc,start=-1):
    dates = np.array(self.query_dates(loc,start))
    date_count = dates.shape[0]
    dates_list = dates.reshape(-1,date_count)[0]
    return [date.strftime('%Y-%m-%d') for date in dates_list if
       date is not None]

def query_dates(self,loc,start):
    loc_str = ""
    start_str = ""
    query = "SELECT␣DISTINCT␣date_time␣FROM␣COVID_TESTS␣AS␣CT"
    if loc != "?":
        loc_str = "␣LEFT␣JOIN␣LOCATIONS␣AS␣L␣ON␣CT.location_id␣=␣L.
           location_id␣WHERE␣country␣=␣%(loc_str)s"
    if start != -1:
        start = datetime.strptime(start,'%Y-%m-%d')
        start_str = "␣date_time␣>␣%(start)s"
    query += loc_str
    if loc == "?" and start != -1:
        query += "␣WHERE"
    elif loc != "?" and start != -1:
        query += "␣AND"
```

```
    query += start_str
    query += "␣ORDER␣BY␣date_time␣ASC;"
    self.check_conn()
    try:
        self.cursor = self.conn.cursor()
        self.cursor.execute(query,{'loc_str':loc, 'start':start})
        return self.cursor.fetchall()
    except ps.DatabaseError:
        self.conn.rollback()
    finally:
        self.cursor.close()
```

Code 3.33: Read Country Names Functions

```
def get_location_names(self):
    loc_names = np.array(self.query_location_names())
    loc_count = loc_names.shape[0]
    return [name.replace("␣","_") for name in loc_names.reshape(-1,
       loc_count)[0] if name is not None]

def query_location_names(self):
    query = """SELECT DISTINCT country FROM COVID_TESTS AS CT
            LEFT JOIN LOCATIONS AS L
            ON CT.location_id = L.location_id
            ORDER BY country ASC"""
    self.check_conn()
    try:
        self.cursor = self.conn.cursor()
        self.cursor.execute(query)
        return self.cursor.fetchall()
    except ps.DatabaseError:
        self.conn.rollback()
    finally:
        self.cursor.close()
```

Code 3.34: Read By id and Read by Location Names or Dates Functions

```
def read_by_id(self, id):
    query = """SELECT * FROM COVID_TESTS AS CT WHERE CT.id = %s
       ORDER BY CT.id;"""
    self.check_conn()
    try:
        self.cursor = self.conn.cursor()
        self.cursor.execute(query, (id,))
        return self.cursor.fetchone()
    except ps.DatabaseError:
        self.conn.rollback()
    finally:
        self.cursor.close()


def read_filter(self, limit=-1, offset=0, loc_name="?", date="?"):
```

```
    date_str = ""
    query = "SELECT␣*␣FROM␣COVID_TESTS␣AS␣CT"
    if loc_name!="?":
        query += "␣LEFT␣JOIN␣LOCATIONS␣AS␣L␣ON␣CT.location_id␣=␣L.
           location_id␣WHERE␣country=%(loc_name)s"
    if date!="?":
        date_str = "␣date_time␣>=␣TO_DATE(%(date)s,'YYYY-MM-DD')"
    if loc_name=="?" and date!="?":
        query += "␣WHERE" + date_str
    elif loc_name!="?" and date!="?":
        query += "␣AND" + date_str

    query += "␣ORDER␣BY␣CT.id␣OFFSET␣%(offset)s"
    if limit != -1:
        query += "␣LIMIT␣" + str(limit)
    query += ";"
    self.check_conn()
    try:
        self.cursor = self.conn.cursor()
        self.cursor.execute(query, {"offset":str(offset),
                                    "loc_name":loc_name,
                                    "date":date})
        return self.cursor.fetchall()
    except ps.DatabaseError:
        self.conn.rollback()
    finally:
        self.cursor.close()
```

3.2.4.4 Insert Function of Covid Tests Table

Code 3.35: Insert Function

```
def insert_row(self, location_id, total_tests, new_tests,
   total_tests_per_thousand, \
    new_tests_per_thousand,new_tests_smoothed,positive_rate,
       date_time):

    query = """INSERT INTO COVID_TESTS(location_id, total_tests,
       new_tests,total_tests_per_thousand,
    new_tests_per_thousand,new_tests_smoothed,positive_rate,
       date_time)
    VALUES(%(location_id)s,%(total_tests)s,%(new_tests)s,%(
       total_tests_per_thousand)s,%(new_tests_per_thousand)s,
    %(new_tests_smoothed)s, %(positive_rate)s, %(date_time)s)"""
    self.check_conn()
    try:
        self.cursor = self.conn.cursor()
        self.cursor.execute(query, {
            'location_id': location_id,
```

```
            'total_tests': total_tests,
            'new_tests': new_tests,
            'total_tests_per_thousand': total_tests_per_thousand,
            'new_tests_per_thousand': new_tests_per_thousand,
            'new_tests_smoothed': new_tests_smoothed,
            'positive_rate': positive_rate,
            'date_time': date_time
        })
        self.conn.commit()
        return True
    except ps.DatabaseError:
        self.conn.rollback()
        return False
    finally:
        self.cursor.close()
```

3.2.4.5 Update Function of Covid Tests Table

Code 3.36: Update Function

```
def update(self, id, location_id, total_tests, new_tests,
   total_tests_per_thousand, \
    new_tests_per_thousand,new_tests_smoothed,positive_rate,
       date_time):
    query = """UPDATE COVID_TESTS SET(location_id, total_tests,
       new_tests,total_tests_per_thousand,
    new_tests_per_thousand,new_tests_smoothed,positive_rate,
       date_time)
    = (%(location_id)s,%(total_tests)s,%(new_tests)s,%(
       total_tests_per_thousand)s,%(new_tests_per_thousand)s,
    %(new_tests_smoothed)s, %(positive_rate)s, %(date_time)s) WHERE
        COVID_TESTS.id = %(id)s"""
    self.check_conn()
    try:
        self.cursor = self.conn.cursor()
        self.cursor.execute(query, {
            'id': id,
            'location_id': location_id,
            'total_tests': total_tests,
            'new_tests': new_tests,
            'total_tests_per_thousand': total_tests_per_thousand,
            'new_tests_per_thousand': new_tests_per_thousand,
            'new_tests_smoothed': new_tests_smoothed,
            'positive_rate': positive_rate,
            'date_time': date_time
        })
        self.conn.commit()
        return True
    except ps.DatabaseError:
```

```
        self.conn.rollback()
        return False
    finally:
        self.cursor.close()
```

3.2.4.6 Delete Function of Covid Tests Table

Code 3.37: Delete Function

```
def delete(self, id):
    query = """DELETE FROM COVID_TESTS AS CT WHERE CT.id = %s"""
    self.check_conn()
    try:
        self.cursor = self.conn.cursor()
        self.cursor.execute(query, (id,))
        self.conn.commit()
        return True
    except ps.DatabaseError:
        self.conn.rollback()
        return False
    finally:
        self.cursor.close()
```

3.2.4.7 Tests Page Function and HTML Codes of Covid Tests Table

Python Code:

Code 3.38: Tests Page Function

```
def tests_page(id = -1):
    user_id = str(session["id"])
    is_admin = False
    if user_id is not None and user_id != "None":
        user = User()
        is_admin = user.isAdmin(user_id)
    else:
        return redirect("/")

    page_id = request.args.get('page') if request.args.get('page')
       is not None else 1
    loc_name = request.args.get('loc_name') if request.args.get('
       loc_name') is not None else "?"
    date = request.args.get('date') if request.args.get('date') is
       not None else "?"
    table_size = 0

    page_id = int(page_id)
```

```
    covid_tests = CovidTests()

    if id != -1 and is_admin:
        covid_tests.delete(int(id))

    loc_name = loc_name.replace("_","␣")

    loc_names = covid_tests.get_location_names()
    offset = (page_id-1)*50
    paginationValues = (page_id-1,page_id,page_id+1) if (page_id)>1
        else (1,2,3)


    try:
        covid_data = np.array(covid_tests.read_filter(50,offset,
           loc_name,date))[:,0:9]
        table_size = covid_data.size
    except IndexError:
        covid_data = np.array([[]])
        table_size = 0

    start_dates = covid_tests.get_dates(loc_name)
    headers = ["␣".join(head.split("_")).title() for head in
       covid_tests.columns]

    return render_template("tests/tests.html", table_headers=
       headers, table_rows = covid_data, \
        paginationValues=paginationValues, locations = loc_names,
           dates = start_dates, data_available=table_size, is_admin
           =is_admin)
```

HTML:

Code 3.39: Tests Page

```
{% extends "after_login.html" %}
{% block title %}Test{% endblock %}
{% block content %}

<script type="text/javascript">
    function goNewDirect(page, loc, date)
    {
        locStr = "";
        dateStr = "";
        if(loc != null && loc != '')
            locStr = "&loc_name="+loc;
        if(date != null && date != '')
            dateStr = "&date="+date;

        if(page == null)
        {
```

```
            window.location.href = "/tests?page=1"+locStr+dateStr;
        }
        else
        {
            window.location.href = "/tests?page="+page+locStr+
               dateStr;
        }
    }
    function changePage(pageToggle, data_available)
    {
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);
        page = urlPage.get("page");
        if(urlPage.get("page") == null || (urlPage.get("page") == '
           1' && pageToggle == 0))
            page = (1).toString();
        else
        {
            if(pageToggle == 0)
                page = (parseInt(urlPage.get("page")) - 1).toString
                   ();
            if(pageToggle == 1)
            {
                if(parseInt(data_available) < 50)
                    window.alert("No␣more␣data.");
                else
                    page = (parseInt(urlPage.get("page")) + 1).
                       toString();
            }
            if(pageToggle == -1)
                page = urlPage.get("page");
        }
        goNewDirect(page, urlPage.get("loc_name"), urlPage.get("
           date"));
    }

    function checkIfSelected(element)
    {
        if(element.value == 'Choose...')
        {
            if(element.id == "inputGroupSelect01")
                alert("You␣didn't␣choose␣a␣country␣!!");
            if(element.id == "inputGroupSelect02")
                alert("You␣didn't␣choose␣a␣date␣!!");
            return false;
        }
        return true;
    }
    function filterCountry()
    {
```

```
        let location = document.getElementById("inputGroupSelect01"
           );
        if(checkIfSelected(location) == false)
            return;
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);
        goNewDirect(null, location.value, urlPage.get("date"));
    }
    function filterDates(button)
    {
        if(checkIfSelected(button) == false)
            return;
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);

        goNewDirect(null, urlPage.get("loc_name"), button.value);
    }
    function reset(type)
    {
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);
        if(type == 'l')
            goNewDirect(urlPage.get("page"), '', urlPage.get("date"
               ));
        if(type == 's')
            goNewDirect(urlPage.get("page"), urlPage.get("loc_name"
               ), '');
    }
    function deleteRow(row)
    {
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);
        if(row.value != "")
            window.location.href = "/tests/"+row.value+"?"+urlPage;
        else
        window.location.href = "/tests?"+urlPage;
    }
    function updateRow(row)
    {
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);
        if(row.value == "")
            window.location.href = "/tests/"+row.value+"?"+urlPage;
        else
            window.location.href = "/update-tests?id="+row.value;
    }

</script>
<div class="container">
    <div class="row␣d-flex␣mt-2">
```

```
    <label for="inputGroupSelect01" class="form-country">
        <h6 style="display:␣inline">Choose a country: </h6>
        <p style="display:␣inline" id="form-country">
            <script>
                const urlStrCt = window.location.search;
                let urlPageCt = new URLSearchParams(urlStrCt);
                if(urlPageCt.get("loc_name") == null)
                    document.getElementById("form-country").
                       textContent = "";
                else
                    document.getElementById("form-country").
                       textContent = "Selected␣country␣>␣"+
                       urlPageCt.get("loc_name");
            </script>
        </p>
    </label>
    <div class="input-group␣col␣mt-3">
        <select class="form-select" id="inputGroupSelect01"
           aria-label="Example␣select␣with␣button␣addon">
          <option selected>Choose...</option>
        {% for loc in locations%}
           <option value={{loc}}>{{loc}}</option>
        {% endfor %}
        </select>
        <button style="width:␣70px;justify-content:␣center;"
           class="btn␣btn-outline-secondary␣" type="button"
           onclick="filterCountry()">Filter</button>
        <button style="width:␣70px;justify-content:␣center;"
           class="btn␣btn-outline-secondary" id="
           resetGroupButton02" onclick="reset('l')"
         type="button">Reset</button>
    </div>

    <div class="col">
        {% if is_admin %}
        <nav aria-label="Page␣navigation␣example">
            <ul class="pagination␣justify-content-end␣mb-0">
                <li class="page-item"><button style="width:␣132
                   px;" class="btn␣btn-warning" type="button"
                    onclick="document.location.href␣=␣'/add-
                       tests'␣">Add</button></li>
            </ul>
        </nav>
        {% endif %}
    </div>
</div>

<div class="row␣d-flex␣mt-3">
    <label for="inputGroupSelect02" class="form-label">
        <h6 style="display:␣inline;">Choose a date: </h6>
```

```
        <p style="display:␣inline;" id="form-date">
            <script>
                const urlStrDate = window.location.search;
                let urlPageDate = new URLSearchParams(
                   urlStrDate);
                if(urlPageDate.get("date") == null)
                    document.getElementById("form-date").
                       textContent = "";
                else
                    document.getElementById("form-date").
                       textContent = "Selected␣date␣>␣"+
                       urlPageDate.get("date");
            </script>
        </p>
    </label>
    <div class="input-group␣col␣mt-1">
        <select class="form-select" id="inputGroupSelect02"
           aria-label="Example␣select␣with␣button␣addon">
          <option selected>Choose...</option>
        {% for date in dates %}
           <option value={{date}}>{{date}}</option>
        {% endfor %}
        </select>
        <button style="width:␣70px;justify-content:␣center;"
           class="btn␣btn-outline-secondary" id="
           inputGroupButton02" onclick="filterDates(document.
           getElementById('inputGroupSelect02'))"
         type="button">Filter</button>
         <button style="width:␣70px;justify-content:␣center;"
            class="btn␣btn-outline-secondary" id="
            resetGroupButton02" onclick="reset('s')"
         type="button">Reset</button>
    </div>

    <div class="col">
        <nav aria-label="Page␣navigation␣example">
            <ul class="pagination␣justify-content-end␣mb-0">
                <li class="page-item"><button class="btn␣btn-
                   outline-secondary" id="prev-button" value="
                   {{data_available}}" style="margin-right:␣5px
                   ;"
                        onclick="changePage(0,document.
                           getElementById('prev-button').value)
                           " type="button">PREV</button></li>
                <li class="page-item"><button class="btn␣btn-
                   outline-secondary"  id="next-button" value="
                   {{data_available}}"
                    onclick="changePage(1,document.
                       getElementById('next-button').value)"
                       type="button">NEXT</button></li>
```

```
            </ul>
        </nav>
    </div>
</div>
<div class="row␣my-3␣mx-0">
    <table class="table␣align-middle␣table-striped␣table-hover␣
       text-center" id="data-table">
        <thead>
          <tr class="align-bottom">

            {% for head in table_headers %}
            <th scope="col">{{head}}</th>
            {% endfor %}
            {% if is_admin %}
                <th scope="col" class="edit-buttons">Delete</th
                   >
                <th scope="col" class="edit-buttons">Update</th
                   >
            {% endif %}
          </tr>
        </thead>
        <tbody>
            {% if data_available %}
                {% for row in table_rows%}
                <tr class="align-items-center">
                    {% for cell in row[1:] %}
                    <td class="align-items-center">
                        <div class="data">
                            {{cell}}
                        </div>
                    </td>
                    {% endfor %}
                    {% if is_admin %}
                    <td class="edit-buttons">
                        <div class="data">
                            <button class="btn␣btn-outline-
                               secondary" style="background-
                               color:␣red;" href="#"
                                    id="delete{{␣row[0]␣}}"
                                       value="{{␣row[0]␣}}"
                                    onclick="deleteRow(document
                                       .getElementById('delete
                                       {{␣row[0]␣}}'))" type="
                                       button"><i
                                    class="bi␣bi-trash" style="
                                       color:␣white;"></i>
                            </button>
                        </div>
                    </td>
                    <td class="edit-buttons">
```

```
                                <div class="data">
                                    <button class="btn␣btn-outline-
                                       secondary" type="button"
                                    href="#" id="update{{␣row[0]␣}}"
                                       value="{{␣row[0]␣}}"
                                    onclick="updateRow(document.
                                       getElementById('update{{␣row[0]␣
                                       }}'))" "><i
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣class="bi-arrow-right"␣style="color
   : blue;"></i>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</button>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</div>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</td>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣{%␣endif␣%}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</tr>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣{%␣endfor␣%}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣{%␣endif␣%}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣</tbody>
␣␣␣␣␣␣␣␣␣␣</table>
␣␣␣␣␣␣␣␣</div>
␣␣␣␣</div>
</div>
{%␣endblock␣%}
```

3.2.4.8 Add Tests Function and HTML Codes

Python Code:

Code 3.40: Add Tests Function

```
def add_tests_page():
    covid_test = CovidTests()
    message = "empty"
    user_id = str(session["id"])
    isAdmin = False
    if user_id is not None and user_id != "None":
         user = User()
         isAdmin = user.isAdmin(user_id)
    else:
        return redirect("/")

    if isAdmin is False:
        return redirect("/tests")

    if request.method == "POST":
        location_id = request.form["location_id"]
        total_tests = request.form["total_tests"]
        new_tests = request.form["new_tests"]
        total_tests_per_thousand = request.form["
           total_tests_per_thousand"] if request.form["
           total_tests_per_thousand"] !="" else None
```

```
        new_tests_per_thousand = request.form["
           new_tests_per_thousand"] if request.form["
           new_tests_per_thousand"] !="" else None
        new_tests_smoothed = request.form["new_tests_smoothed"] if
           request.form["new_tests_smoothed"] !="" else None
        positive_rate = request.form["positive_rate"] if request.
           form["positive_rate"] !="" else None
        date_time = request.form["date_time"] if request.form["
           date_time"] !="" else None
        result = covid_test.insert_row(location_id, total_tests,
           new_tests, total_tests_per_thousand,
           new_tests_per_thousand, new_tests_smoothed,
           positive_rate, date_time)
        if result:
            message = "success"
        else:
            message = "failed"
    return render_template("tests/add-tests.html", message=message)
```

HTML:

Code 3.41: Add Tests Page

```
{% extends "after_login.html" %}
{% block title %}Cases{% endblock %}
{% block content %}
<script type="text/javascript">
    const message = '{{message}}';

    if (message != "empty") {
        if (message == "success") {
            window.alert("Data␣successfully␣added.");
            window.location.href = "/tests";
        }
        else {
            window.alert("Data␣could␣not␣be␣added.␣Please␣control␣
               the␣values␣and␣try␣again!.");
            window.location.href = "/add-tests";
        }
    }

</script>
<h2 class="container␣my-4">Add Tests Data</h2>
<div class="container␣my-4">
    <form action="/add-tests" method="POST">
        <div class="mb-3">
            <label for="location_id" class="form-label">Location Id
               </label>
            <div class="d-flex">
                <input type="text" name="location_id" class="form-
                   control" style="width:␣30%;" id="location_id"
```

```
                aria-describedby="emailHelp">
            </div>
        </div>
        <div class="mb-3">
            <label for="total_tests" class="form-label">Total Tests
               </label>
            <div class="d-flex">
                <input type="text" name="total_tests" class="form-
                   control" style="width:␣30%;" id="total_tests"
                    aria-describedby="emailHelp">
                </div>
        </div>
        <div class="mb-3">
            <label for="new_tests" class="form-label">New Tests</
               label>
            <div class="d-flex">
                <input type="text" name="new_tests" class="form-
                   control" style="width:␣30%;" id="new_tests"
                    aria-describedby="emailHelp">
            </div>
        </div>
        <div class="mb-3">
            <label for="total_tests_per_thousand" class="form-label
               ">Total Tests Per Thousand</label>
            <div class="d-flex">
                <input type="text" name="total_tests_per_thousand"
                   class="form-control" style="width:␣30%;"
                    id="total_tests_per_thousand" aria-describedby=
                       "emailHelp">
            </div>
        </div>
        <div class="mb-3">
            <label for="new_tests_per_thousand" class="form-label">
               New Tests Per Thousand</label>
            <div class="d-flex">
                <input type="text" name="new_tests_per_thousand"
                   class="form-control" style="width:␣30%;"
                    id="new_tests_per_thousand" aria-describedby="
                       emailHelp">
            </div>
        </div>
        <div class="mb-3">
            <label for="new_tests_smoothed" class="form-label">New
               Tests Smoothed</label>
            <div class="d-flex">
                <input type="text" name="new_tests_smoothed" class=
                   "form-control" style="width:␣30%;"
                    id="new_tests_smoothed" aria-describedby="
                       emailHelp">
            </div>
```

```
        </div>
        <div class="mb-3">
            <label for="positive_rate" class="form-label">Positive
               Rate</label>
            <div class="d-flex">
                <input type="text" name="positive_rate" class="form
                   -control" style="width:␣30%;" id="positive_rate"
                    aria-describedby="emailHelp">
            </div>
        </div>
        <div class="mb-3">
            <label for="date_time" class="form-label">Date (yyyy-mm
               -dd)</label>
            <div class="d-flex">
                <input type="text" name="date_time" class="form-
                   control" style="width:␣30%;" id="date_time"
                    aria-describedby="emailHelp">
            </div>
        </div>
        <button type="submit" class="btn␣btn-primary">Update Data</
           button>
        <button type="button" href="#" onclick="document.location.
           href=␣'/tests'" class="btn␣btn-primary">Back</button>
    </form>
</div>
{% endblock %}
```

3.2.4.9 Update Tests Function and HTML Codes

Python Code:

Code 3.42: Update Tests Function

```
def update_tests_page():
    row_id = request.args.get('id')

    row_id = int(row_id)
    user_id = str(session["id"])
    isAdmin = False

    if user_id is not None and user_id != "None":
        user = User()
        isAdmin = user.isAdmin(user_id)
    else:
        return redirect("/")

    if isAdmin is False:
        return redirect("/tests")

    covid_test = CovidTests()
```

```
    row = np.array(covid_test.read_by_id(row_id))
    message = "empty"
    if request.method == "POST":
        total_tests = request.form["total_tests"] if request.form["
           total_tests"] !="" else row[2]
        new_tests = request.form["new_tests"] if request.form["
           new_tests"] !="" else row[3]
        total_tests_per_thousand = request.form["
           total_tests_per_thousand"] if request.form["
           total_tests_per_thousand"] !="" else row[4]
        new_tests_per_thousand = request.form["
           new_tests_per_thousand"] if request.form["
           new_tests_per_thousand"] !="" else row[5]
        new_tests_smoothed = request.form["new_tests_smoothed"] if
           request.form["new_tests_smoothed"] !="" else row[6]
        positive_rate = request.form["positive_rate"] if request.
           form["positive_rate"] !="" else row[7]
        date_time = request.form["date_time"] if request.form["
           date_time"] !="" else row[8]
        result = covid_test.update(row_id, row[1], total_tests,
           new_tests, total_tests_per_thousand,
           new_tests_per_thousand, new_tests_smoothed,
           positive_rate, date_time)
        if result:
            message = "success"
        else:
            message = "failed"

    return render_template("tests/update-tests.html", id = row_id,
       data=row, message=message)
```

HTML:

Code 3.43: Update Tests Page

```
{% extends "after_login.html" %}
{% block title %}Cases{% endblock %}
{% block content %}
<script type="text/javascript">
    const message = '{{message}}';
    if (message != "empty") {
        if (message == "success") {
            window.alert("Data␣successfully␣updated.");
            window.location.href = "/tests";
        }
        else {
            window.alert("Data␣could␣not␣be␣updated.␣Please␣control
               ␣the␣values␣and␣try␣again!.");
            const urlStr = window.location.search;
            let urlPage = new URLSearchParams(urlStr);
            window.location.href = "/update-tests?id="+urlPage.get(
```

```
                "id");
        }
    }

</script>
<h2 class="container␣my-4">Update Test Data</h2>
<div class="container␣my-4">
    <form action="/update-tests?id={{id}}" method="POST">
        <div class="mb-3">
            <label for="total_tests" class="form-label">Total Tests
               </label>
            <div class="d-flex">
                <input type="text" name="total_tests" class="form-
                   control" style="width:␣30%;" id="total_tests"
                    aria-describedby="emailHelp" placeholder="{{
                       data[2]}}">
                </div>
        </div>
        <div class="mb-3">
            <label for="new_tests" class="form-label">New Tests</
               label>
            <div class="d-flex">
                <input type="text" name="new_tests" class="form-
                   control" style="width:␣30%;" id="new_tests"
                    aria-describedby="emailHelp" placeholder="{{
                       data[3]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="total_tests_per_thousand" class="form-label
               ">Total Tests Per Thousand</label>
            <div class="d-flex">
                <input type="text" name="total_tests_per_thousand"
                   class="form-control" style="width:␣30%;"
                    id="total_tests_per_thousand" aria-describedby=
                       "emailHelp" placeholder="{{data[4]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="new_tests_per_thousand" class="form-label">
               New Tests Per Thousand</label>
            <div class="d-flex">
                <input type="text" name="new_tests_per_thousand"
                   class="form-control" style="width:␣30%;"
                    id="new_tests_per_thousand" aria-describedby="
                       emailHelp" placeholder="{{data[5]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="new_tests_smoothed" class="form-label">New
```

```
                Tests Smoothed</label>
            <div class="d-flex">
                <input type="text" name="new_tests_smoothed" class=
                   "form-control" style="width:␣30%;"
                    id="new_tests_smoothed" aria-describedby="
                       emailHelp" placeholder="{{data[6]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="positive_rate" class="form-label">Positive
               Rate</label>
            <div class="d-flex">
                <input type="text" name="positive_rate" class="form
                   -control" style="width:␣30%;" id="positive_rate"
                    aria-describedby="emailHelp" placeholder="{{
                       data[7]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="date_time" class="form-label">Date (yyyy-mm
               -dd)</label>
            <div class="d-flex">
                <input type="text" name="date_time" class="form-
                   control" style="width:␣30%;" id="date_time"
                    aria-describedby="emailHelp" placeholder="{{
                       data[8]}}">
            </div>
        </div>
        <button type="submit" class="btn␣btn-primary">Add Data</
           button>
        <button type="button" href="#" onclick="document.location.
           href=␣'/tests'" class="btn␣btn-primary">Back</button>
    </form>
</div>
{% endblock %}
```

Code 3.44: Search functions with Locations_id and date

```
    #Selecting by primary key value
    def selectFromLOCandDate(self, loc_id, date):
        query = f"""select location_id, date_time, icu_patients ,
        icu_patients_per_million,hosp_patients ,
           hosp_patients_per_million ,
        weekly_icu_admissions ,weekly_icu_admissions_per_million ,
           weekly_hosp_admissions ,
        weekly_hosp_admissions_per_million FROM HOSPITAL_AND_ICU
        WHERE (((icu_patients IS NOT NULL) OR (hosp_patients IS NOT
            NULL)) and
        location_id = '{loc_id}' and date_time = '{date}')"""
        self.con_control()
        try:
            cursor = self.connection.cursor()
            cursor.execute(query)
            result = cursor.fetchall()
        except psycopg2.DatabaseError:
            self.connection.rollback()
            result = None
        finally:
            cursor.close()
            return result
```

Code 3.45: Search functions that returns just id with Locations_id and date

```
    def selectFromLOCandDateReturnID(self, loc_id, date):
        query = f"""select id from HOSPITAL_AND_ICU
        WHERE (location_id = '{loc_id}' and date_time = '{date}')
           """
        self.con_control()
        try:
            cursor = self.connection.cursor()
            cursor.execute(query)
            result = cursor.fetchone()
        except psycopg2.DatabaseError:
            self.connection.rollback()
            result = None
        finally:
            cursor.close()
            return result
```

Code 3.46: Search functions with Locations_id

```
def selectFromLOC(self, location_id, offset):
    query = f"""SELECT location_id, date_time, icu_patients ,
    icu_patients_per_million,hosp_patients ,
       hosp_patients_per_million ,
    weekly_icu_admissions ,weekly_icu_admissions_per_million ,
       weekly_hosp_admissions ,
    weekly_hosp_admissions_per_million FROM HOSPITAL_AND_ICU
    WHERE (((icu_patients IS NOT NULL) OR (hosp_patients IS NOT
        NULL)) and (location_id = '{location_id}'))
    ORDER BY date_time desc OFFSET {offset} ROWS FETCH NEXT 50
       ROWS ONLY;"""
    self.con_control()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        result = cursor.fetchall()
    except psycopg2.DatabaseError:
        self.connection.rollback()
        result = None
    finally:
        cursor.close()
        return result
```

Code 3.47: Search function that returns all data

```
#Selecting all rows primary key value
def selectAll(self):
    query = """SELECT location_id, date_time, icu_patients ,
    icu_patients_per_million,hosp_patients ,
       hosp_patients_per_million ,
    weekly_icu_admissions ,weekly_icu_admissions_per_million ,
       weekly_hosp_admissions ,
    weekly_hosp_admissions_per_million FROM HOSPITAL_AND_ICU
    WHERE ((icu_patients IS NOT NULL) OR (hosp_patients IS NOT
       NULL))"""
    self.con_control()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        result = cursor.fetchall()
    except psycopg2.DatabaseError:
        self.connection.rollback()
        result = None
    finally:
        cursor.close()
        return result
```

Code 3.48: Search function that returns all data with offset

```
def selectAll(self, offset):
    query = f"""SELECT location_id, date_time, icu_patients ,
    icu_patients_per_million,hosp_patients ,
       hosp_patients_per_million ,
    weekly_icu_admissions ,weekly_icu_admissions_per_million ,
       weekly_hosp_admissions ,
    weekly_hosp_admissions_per_million FROM HOSPITAL_AND_ICU
    WHERE ((icu_patients IS NOT NULL) OR (hosp_patients IS NOT
       NULL))
    ORDER BY date_time desc OFFSET {offset} ROWS FETCH NEXT 50
       ROWS ONLY"""
    self.con_control()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        result = cursor.fetchall()
    except psycopg2.DatabaseError:
        self.connection.rollback()
        result = None
    finally:
        cursor.close()
        return result
```

Code 3.49: Delete functions with id

```
#Deleting a row by id
def delete(self, id):
    query = """DELETE FROM HOSPITAL_AND_ICU AS H WHERE H.id = %
       s"""
    self.con_control()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query, (id,))
        self.connection.commit()
    except psycopg2.DatabaseError:
        self.connection.rollback()
    finally:
        cursor.close()
```

Code 3.50: Insert functions

```
#Inserting a new row to the table
def insert(self, iso_code, icu_patients,
   icu_patients_per_million, hosp_patients,
   hosp_patients_per_million, weekly_icu_admissions,
   weekly_icu_admissions_per_million, weekly_hosp_admissions,
   weekly_hosp_admissions_per_million, date):

    query = f"""INSERT INTO HOSPITAL_AND_ICU(location_id,
       icu_patients ,
    icu_patients_per_million ,hosp_patients ,
       hosp_patients_per_million ,
    weekly_icu_admissions ,weekly_icu_admissions_per_million ,
       weekly_hosp_admissions ,
    weekly_hosp_admissions_per_million , date_time)
    VALUES('{iso_code}',{icu_patients},{
       icu_patients_per_million},{hosp_patients},
    {hosp_patients_per_million}, {weekly_icu_admissions}, {
       weekly_icu_admissions_per_million},
    {weekly_hosp_admissions}, {
       weekly_hosp_admissions_per_million}, '{date}');"""
    self.con_control()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        self.connection.commit()
    except psycopg2.DatabaseError:
        self.connection.rollback()
    finally:
        cursor.close()
```

Code 3.51: Update functions with date and location_id

```
#Updating a row by id
def update(self, iso_code, icu_patients,
   icu_patients_per_million, hosp_patients,
   hosp_patients_per_million, weekly_icu_admissions,
   weekly_icu_admissions_per_million, weekly_hosp_admissions,
   weekly_hosp_admissions_per_million, date):
    query = f"""UPDATE HOSPITAL_AND_ICU SET (location_id,
       icu_patients ,
    icu_patients_per_million,hosp_patients ,
       hosp_patients_per_million ,
    weekly_icu_admissions ,weekly_icu_admissions_per_million ,
       weekly_hosp_admissions ,
    weekly_hosp_admissions_per_million , date_time)
    = ('{iso_code}',{icu_patients},{icu_patients_per_million},{
       hosp_patients},
    {hosp_patients_per_million}, {weekly_icu_admissions}, {
       weekly_icu_admissions_per_million},
    {weekly_hosp_admissions}, {
```

```
        weekly_hosp_admissions_per_million}, '{date}') WHERE
        HOSPITAL_AND_ICU.date_time = '{date}' and
        HOSPITAL_AND_ICU.location_id = '{iso_code}'"""
    self.con_control()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        self.connection.commit()
    except psycopg2.DatabaseError:
        self.connection.rollback()
    finally:
        cursor.close()
```

Code 3.52: Function that checks if that data exists

```
def is_there(self, date, location):
    query = f"""select count(id) from hospital_and_icu
            where (date_time = '{date}' and location_id = '{
               location}')
            group by date_time, location_id"""

    self.con_control()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        result = cursor.fetchone()
    except psycopg2.DatabaseError:
        self.connection.rollback()
        result = None
    finally:
        cursor.close()
        return True if result is not None and result[0]>0 else
           False
```

Code 3.53: Function that returns all country names from Hospital_and_ICU table's iso_code

```
def get_country_names(self):
    query = """SELECT distinct L.country FROM hospital_and_icu
       as H left join locations as L on(h.location_id = L.
       location_id)
               order by 1;"""
    self.connect()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        return cursor.fetchall()
    except psycopg2.DatabaseError:
        self.connection.rollback()
    finally:
        cursor.close()
```

3.2.4.10 Patients Page

Code 3.54: HTML file of patients page

```
{% extends "after_login.html" %}
{% block title %}Patients{% endblock %}
{% block content %}
<div class="container-fluid">
    <div class="row␣my-2␣d-flex␣justify-content-between">
        <h3 class="fw-bold␣text-secondary␣col-4">Patients</h3>
        {% if isadmin == True %}
        <div class="col-2">
            <a href="/patients/edit" role="button" aria-pressed="
               true"
                class="btn␣btn-warning␣active␣w-100␣edit-btn">Edit<
                   /a>
        </div>
        {% endif %}
    </div>
    {% with messages = get_flashed_messages() %}
    {% if messages %}

    {% for message in messages %}
    <p class="text-danger␣text-center">{{message}}</p>
    {% endfor %}

    {% endif %}
    {% endwith %}
    <div class="row">
        <div class="col-6"style="width:␣100%;">
            <div class="row␣justify-content-start␣align-items-
               center">
                <div class="col-4" style="width:␣33%;">
                    <select id="sel_country" class="form-select"
                       name="country" aria-label=".form-select-sm␣
                       example">
                        {% if countries %}

                        {% for c in countries %}
                        <option>{{c[0]}}</option>
                        {% endfor %}

                        {% endif %}
                    </select>
                </div>
                <div class="col-4"style="width:␣34%;">
                    <input type="text" name="date" class="form-
                       control" id="date" placeholder="Please␣enter
                       ␣date␣in␣'YYYY-MM-DD'␣format">
                </div>
                <div class="col-4"style="width:␣33%;">
```

```
                    <button type="button" id="filter1" class="btn␣
                       btn-secondary␣w-100␣my-3"><a
                            class="link-light␣text-decoration-none"
                                id="Filter" href="#">Filter</a></
                               button>
                    <script>
                        document.querySelector("#filter1").
                           addEventListener("click", function () {
                            Filter = document.getElementById("
                               Filter")
                            country = document.getElementById("
                               sel_country").value
                            date = document.getElementById("date").
                               value
                            Filter.href = "/patients?"
                            Filter.href += "countryName=" + country
                                + "&dateVariable=" + date
                            window.location.replace(Filter.href)
                        });
                    </script>
                </div>
            </div>
        </div>
    </div>
    <div class="row␣my-3␣mx-0">
        <div class="col-12">
            <table class="table␣align-middle␣table-striped␣table-
               hover␣text-center" id="patients">
                <thead>
                    <tr>
                        <th scope="col">Country</th>
                        <th scope="col">Date</th>
                        {% for header in headings %}
                        <th scope="col">{{header}}</th>
                        {% endfor %}
                    </tr>
                </thead>
                <tbody>
                    {% for row in patients %}
                    <tr>
                        {% for cell in row %}
                        <td>
                            <div class="data">
                                {{cell}}
                            </div>
                        </td>
                        {% endfor %}
                    </tr>
                    {% endfor %}
                </tbody>
```

```
            </table>
        </div>
    </div>
    <div class="row">
        <div class="col-6">
            <nav aria-label="Page␣navigation" class="row␣h-100␣d-
               flex␣align-items-center">
                <ul class="pagination␣justify-content-end␣mb-0">
                    <li id="4" class="page-item"><a class="btn␣btn-
                       outline-secondary␣w-100">first</a></li>
                    <li id="1" class="page-item"><a class="btn␣btn-
                       outline-secondary␣w-100">{{paginationValues
                       [0]}}</a></li>
                    <li id="2" class="page-item"><a class="btn␣btn-
                       outline-secondary␣w-100">{{paginationValues
                       [1]}}</a></li>
                    <li id="3" class="page-item"><a class="btn␣btn-
                       outline-secondary␣w-100">{{paginationValues
                       [2]}}</a></li>
                    <script>
                        l1 = document.getElementById("1")
                        l2 = document.getElementById("2")
                        l3 = document.getElementById("3")
                        l4 = document.getElementById("4")

                        let params = (new URL(document.location)).
                           searchParams;
                        let name = params.get("countryName");
                        let url = "/patients?"

                        if (name != null) {
                            url = url + "countryName=" + name + "&"
                        }
                        l1.firstChild.href = url + "pageNumber=" +
                           l1.firstChild.innerText
                        l2.firstChild.href = url + "pageNumber=" +
                           l2.firstChild.innerText
                        l3.firstChild.href = url + "pageNumber=" +
                           l3.firstChild.innerText
                        l4.firstChild.href = url + "pageNumber=1"
                    </script>
                </ul>
            </nav>
        </div>
    </div>
</div>
{% endblock %}
```

Code 3.55: View Function of Patients page

```
def patients_page():
    connection = hospital_and_icu()
    #pagination
    countryName = request.args.get("countryName")
    dateFilter = request.args.get("dateVariable")
    pageNumber =request.args.get("pageNumber") if request.args.get(
       "pageNumber") is not None else "1"
    pageNumber = int(pageNumber)
    offset = (pageNumber-1)*50
    paginationValues = (pageNumber,pageNumber+1,pageNumber+2) if (
       pageNumber)>0 else (0,1,2)

    location = Locations()
    countries = connection.get_country_names()
    patients = None
    isadmin = False
    user_id = str(session["id"])
    if user_id is not None and user_id != "None":
        user = User()
        isadmin = user.isAdmin(user_id)
    else:
        return redirect("/")
    headings = ("icu_patients", "icu_patients_per_million","
       hosp_patients" ,"hosp_patients_per_million" , "
       weekly_icu_admissions" ,"weekly_icu_admissions_per_million"
       ,"weekly_hosp_admissions" ,"
       weekly_hosp_admissions_per_million")
    if(countryName is not None and dateFilter =='’):
        country_id = location.get_id_by_country_name(countryName)
        result = connection.selectFromLOC(country_id[0], offset)
    elif(countryName is not None and dateFilter is not None):
        country_id = location.get_id_by_country_name(countryName)
        result = connection.selectFromLOCandDate(country_id[0],
           dateFilter)
    else:
        result = connection.selectAll(offset)
    patients = np.zeros([1, 10], dtype='str')
    if(result is not None):
        for row in result:
            newRow = np.array(row)
            patients = np.vstack([patients, newRow])

    patients = np.delete(patients, 0, 0)
    return render_template("patients/patients.html",headings=
       headings, isadmin=isadmin,  patients=patients, countries=
       countries, paginationValues=paginationValues)
```

3.2.4.11 Edit Page

Code 3.56: HTML file of Patients Edit Page

```
{% extends "after_login.html" %}
{% block title %}Patients Edit{% endblock %}
{% block content %}
<div class="container-fluid">

    <h3 class="fw-bold␣text-secondary">Patients | Edit</h3>
    {% with messages = get_flashed_messages() %}
    {% if messages %}

    {% for message in messages %}
    <p class="text-danger␣text-center">{{message}}</p>
    {% endfor %}

    {% endif %}
    {% endwith %}
    <div class="row␣w-100␣mt-3␣add-del-btn-cont">
    </div>

    <div class="row">
        <div class="row␣justify-content-start␣align-items-center">
            <div class="col-2" style="width:␣20%;">
                <select id="sel_country" class="form-select" name="
                   country" aria-label=".form-select-sm␣example">
                    {% if countries %}

                    {% for c in countries %}
                    <option>{{c[0]}}</option>
                    {% endfor %}

                    {% endif %}
                </select>
            </div>
            <div class="col-2" style="width:␣20%;">
                <input type="text" name="date" class="form-control"
                    id="date" placeholder="Please␣enter␣date␣in␣'
                   YYYY-MM-DD'␣format">
            </div>
            <div class="col-2" style="width:␣20%;">
                <button type="button" id="filter1" class="btn␣btn-
                   secondary␣w-100"><a
                        class="link-light␣text-decoration-none" id=
                           "Filter" href="#">Filter</a></button>
                <script>
                    document.querySelector("#filter1").
                       addEventListener("click", function () {
                        Filter = document.getElementById("Filter")
```

```
                    country = document.getElementById("
                       sel_country").value
                    date = document.getElementById("date").
                       value
                    Filter.href = "/patients/edit?"
                    Filter.href += "countryName=" + country + "
                       &dateVariable=" + date
                    window.location.replace(Filter.href)
                });
            </script>
        </div>

        <div class="col-2" style="width:␣20%;">
            <a id="update_patients" href="/patients/update"
               class="btn␣btn-secondary␣w-100" role="button"
               aria-pressed="true">Update</a>
        </div>
        <div class="col-2" style="width:␣20%;">
            <a id="add_patients" href="/patients/add" class="
               btn␣btn-secondary␣w-100" role="button" aria-
               pressed="true">Add</a>
        </div>
    </div>
</div>
<div class="row␣my-3␣mx-0">
    <div class="col-12">
        <table class="table␣align-middle␣table-striped␣table-
           hover␣text-center" id="patients">
            <thead>
                <tr>
                    <th scope="col">Country</th>
                    <th scope="col">Date</th>
                    {% for header in headings %}
                    <th scope="col">{{header}}</th>
                    {% endfor %}
                    <th scope="col">Delete</th>
                </tr>
            </thead>
            <tbody>
                {% for row in patients %}
                <tr>
                    {% for cell in row %}
                    <td>
                        <div class="data">
                            {{cell}}
                        </div>
                    </td>
                    {% endfor %}
                    <td>
                        <div class="data">
```

```
                                <button class="btn␣btn-outline-
                                   secondary" href="#" id="delete{{
                                   ␣row[0]␣}}+{{␣row[1]␣}}"
                                value="countryName={{␣row[0]␣}}&
                                   dateVariable={{␣row[1]␣}}&
                                   deleteMode=on" onclick="
                                   deleteQuery(document.
                                   getElementById('delete{{␣row[0]␣
                                   }}+{{␣row[1]␣}}'))"
                                type="button">Delete
                                </button>
                            </div>
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
                <script type="text/javascript">
                    function deleteQuery(row) {
                        if (confirm('Are you sure you want to
                           delete this record?')){
                            const urlStr = window.location.search;
                            if (row.value != "")
                                window.location.href = "/patients/
                                   edit?" + row.value;
                            else
                                window.location.href = "/patients/
                                   edit";
                        }
                    }
                </script>
            </table>
        </div>
    </div>
    <div class="row">
        <nav aria-label="Page␣navigation" class="row␣h-100␣d-flex␣
           align-items-center">
            <ul class="pagination␣justify-content-center␣mb-0">
                <li id="4" class="page-item"><a class="btn␣btn-
                   outline-secondary␣w-100">first</a></li>
                <li id="1" class="page-item"><a class="btn␣btn-
                   outline-secondary␣w-100">{{paginationValues[0]}}
                   </a></li>
                <li id="2" class="page-item"><a class="btn␣btn-
                   outline-secondary␣w-100">{{paginationValues[1]}}
                   </a></li>
                <li id="3" class="page-item"><a class="btn␣btn-
                   outline-secondary␣w-100">{{paginationValues[2]}}
                   </a></li>
                <script>
                    l1 = document.getElementById("1")
```

```
                    l2 = document.getElementById("2")
                    l3 = document.getElementById("3")
                    l4 = document.getElementById("4")

                    let params = (new URL(document.location)).
                       searchParams;
                    let name = params.get("countryName");
                    let url = "/patients/edit?"

                    if (name != null) {
                        url = url + "countryName=" + name + "&"
                    }
                    l1.firstChild.href = url + "pageNumber=" + l1.
                       firstChild.innerText
                    l2.firstChild.href = url + "pageNumber=" + l2.
                       firstChild.innerText
                    l3.firstChild.href = url + "pageNumber=" + l3.
                       firstChild.innerText
                    l4.firstChild.href = url + "pageNumber=1"
                </script>
            </ul>
        </nav>
    </div>
</div>
{% endblock %}
```

Code 3.57: View Function of Edit Page

```
def edit_patients_page():
    connection = hospital_and_icu()
    #pagination
    countryName = request.args.get("countryName")
    dateFilter = request.args.get("dateVariable")
    pageNumber =request.args.get("pageNumber") if request.args.get(
       "pageNumber") is not None else "1"
    pageNumber = int(pageNumber)
    offset = (pageNumber-1)*50
    paginationValues = (pageNumber,pageNumber+1,pageNumber+2) if (
       pageNumber)>0 else (0,1,2)


    isadmin = False
    user_id = str(session["id"])
    if user_id is not None and user_id != "None":
        user = User()
        isadmin = user.isAdmin(user_id)
    else:
        return redirect("/")

    if isadmin == False:
        return redirect("/patients")
```

```
    if(request.args.get("deleteMode") == "on"):
        if(countryName != '' and dateFilter != ''):
            delete_patients(countryName, dateFilter)
        else:
            flash("For␣delete␣operation,␣Date␣or␣Country␣field␣
               cannot␣be␣blank!")
        return redirect("/patients/edit")
    else:
        location = Locations()
        countries = connection.get_country_names()
        patients = None
        headings = ("icu_patients", "icu_patients_per_million","
           hosp_patients" ,"hosp_patients_per_million" , "
           weekly_icu_admissions" ,"
           weekly_icu_admissions_per_million" ,"
           weekly_hosp_admissions" ,"
           weekly_hosp_admissions_per_million")
        if(countryName is not None and dateFilter =='''):
            country_id = location.get_id_by_country_name(
               countryName)
            result = connection.selectFromLOC(country_id[0], offset
               )
        elif(countryName is not None and dateFilter is not None):
            country_id = location.get_id_by_country_name(
               countryName)
            result = connection.selectFromLOCandDate(country_id[0],
                dateFilter)
        else:
            result = connection.selectAll(offset)
        patients = np.zeros([1, 10], dtype='str')
        if(result is not None):
            for row in result:
                newRow = np.array(row)
                patients = np.vstack([patients, newRow])

        patients = np.delete(patients, 0, 0)
        return render_template("patients/edit-patients.html",
           headings=headings,  patients=patients, countries=
           countries, paginationValues=paginationValues)
```

Code 3.58: Delete function

```
def delete_patients(country_id,date):
    connection = hospital_and_icu()
    loc_con = Locations()
    check = connection.selectFromLOCandDateReturnID(country_id,
       date)
    if(check is not None):
        connection.delete(check)
    else:
```

```
        flash("There␣is␣no␣this␣record␣in␣database")
        return redirect("/patients/edit")
```

3.2.4.12 Add Page

Code 3.59: HTML file of Patients Add Page

```
{% extends "after_login.html" %}
{% block title %}Patients Add{% endblock %}
{% block content %}

<div class="container">
    <h3 class="mx-auto">Add New Patients Data</h3>
    {% with messages = get_flashed_messages() %}
    {% if messages %}

    {% for message in messages %}
    <p class="text-danger␣text-start␣fs-3␣fw-bold">{{message}}</p>
    {% endfor %}

    {% endif %}
    {% endwith %}
    <form action="/patients/add" method="POST">
        <div class="mb-3">
            <label for="icu_patients" class="form-label">icu
               patients</label>
            <input type="text" name="icu_patients" class="form-
               control">
        </div>
        <div class="mb-3">
            <label for="icu_patients_per_million" class="form-label
               ">icu patients per million</label>
            <input type="text" name="icu_patients_per_million"
               class="form-control">
        </div>
        <div class="mb-3">
            <label for="hosp_patients" class="form-label">hospital
               patients</label>
            <input type="text" name="hosp_patients" class="form-
               control">
        </div>
        <div class="mb-3">
            <label for="hosp_patients_per_million" class="form-
               label">hospital patients per million</label>
            <input type="text" name="hosp_patients_per_million"
               class="form-control">
        </div>
        <div class="mb-3">
```

```
            <label for="weekly_icu_admissions" class="form-label">
               weekly icu admissions</label>
            <input type="text" name="weekly_icu_admissions" class="
               form-control">
        </div>
        <div class="mb-3">
            <label for="weekly_icu_admissions_per_million" class="
               form-label">weekly icu admissions per million</label
               >
            <input type="text" name="
               weekly_icu_admissions_per_million" class="form-
               control">
        </div>
        <div class="␣mb-3">
            <label for="weekly_hosp_admissions" class="form-label">
               weekly hospital admissions</label>
            <input type="text" name="weekly_hosp_admissions" class=
               "form-control">
        </div>
        <div class="mb-3">
            <label for="weekly_hosp_admissions_per_million" class="
               form-label">weekly hospital admissions per million</
               label>
            <input type="text" name="
               weekly_hosp_admissions_per_million" class="form-
               control">
        </div>
        <p>Country</p>
        <select class="␣form-select␣mb-4" name="country" aria-label
           =".form-select-sm␣example">
            {% if countries %}

            {% for c in countries %}
            <option>{{c[0]}}</option>
            {% endfor %}

            {% endif %}
        </select>
        <div class="mb-3">
            <label for="date" class="form-label">Date</label>
            <p>Please enter date in 'YYYY-MM-DD' format</p>
            <input type="text" name="date" class="form-control" id=
               "date">
        </div>
        <button type="submit" class="btn␣btn-primary">Submit</
           button>
    </form>

</div>
{% endblock %}
```

Code 3.60: View Function of Add Page

```
def add_patients_data():
    connection = hospital_and_icu()
    loc_con = Locations()
    countries = loc_con.get_country_names()
    headings = ("icu_patients", "icu_patients_per_million","
       hosp_patients" ,"hosp_patients_per_million" , "
       weekly_icu_admissions" ,"weekly_icu_admissions_per_million"
       ,"weekly_hosp_admissions" ,"
       weekly_hosp_admissions_per_million")

    isadmin = False
    user_id = str(session["id"])
    if user_id is not None and user_id != "None":
        user = User()
        isadmin = user.isAdmin(user_id)
    else:
        return redirect("/")

    if isadmin == False:
        return redirect("/patients")

    if request.method == "POST":
        if(request.form["country"] and request.form["date"]):
            location_id = request.form["country"]
            location_id = loc_con.get_id_by_country_name(
               location_id)
            date_time = request.form["date"]
        else:
            flash("Both␣country␣and␣date␣fields␣cannot␣be␣blank")
            return render_template("patients/add.html", headings =
               headings, countries = countries)
        if(request.form["icu_patients"] or request.form["
           hosp_patients"]):
            icu_patients = request.form["icu_patients"] if request.
               form["icu_patients"] !="" else "NULL"
            hosp_patients = request.form["hosp_patients"] if
               request.form["hosp_patients"] !="" else "NULL"
        else:
            flash("Either␣icu_patients␣or␣hospital_patients␣cannot␣
               be␣blank!")
            return render_template("patients/add.html", headings =
               headings, countries = countries)
        icu_patients_per_million = request.form["
           icu_patients_per_million"] if request.form["
           icu_patients_per_million"] !="" else "NULL"
        hosp_patients_per_million = request.form["
           hosp_patients_per_million"] if request.form["
           hosp_patients_per_million"] !="" else "NULL"
        weekly_hosp_admissions = request.form["
```

```
        weekly_hosp_admissions"] if request.form["
        weekly_hosp_admissions"] !="" else "NULL"
    weekly_hosp_admissions_per_million = request.form["
        weekly_hosp_admissions_per_million"] if request.form["
        weekly_hosp_admissions_per_million"] !="" else "NULL"
    weekly_icu_admissions = request.form["weekly_icu_admissions
        "] if request.form["weekly_icu_admissions"] !="" else "
        NULL"
    weekly_icu_admissions_per_million = request.form["
        weekly_icu_admissions_per_million"] if request.form["
        weekly_icu_admissions_per_million"] !="" else "NULL"

    country_id_fetched  = loc_con.is_there(location_id[0])

    if country_id_fetched is None:
        flash("Please␣enter␣a␣valid␣country")
        return render_template("patients/add.html", headings=
            headings, countries=countries)

    (country_id,) = country_id_fetched

    format = "%Y-%m-%d"

    try:
        datetime.strptime(date_time, format)
    except ValueError:
        flash("Please␣enter␣a␣valid␣date␣in␣the␣format␣YYYY-MM-
            DD")
        return render_template("patients/add.html", headings=
            headings,countries=countries)

    check_q = connection.is_there(date_time, country_id)
    if check_q:
        flash("You␣can␣not␣add␣a␣new␣record␣into␣an␣already␣
            existing␣record")
        return render_template("patients/add.html",headings=
            headings,  countries=countries)

    connection.insert(country_id, icu_patients,
        icu_patients_per_million, hosp_patients,
        hosp_patients_per_million, weekly_icu_admissions,
        weekly_icu_admissions_per_million,
        weekly_hosp_admissions,
        weekly_hosp_admissions_per_million, date_time)

    flash("Successfully␣created")
    return render_template("patients/add.html", headings=
        headings, countries=countries)
else:
    return render_template("patients/add.html", headings=
```

```
            headings, countries=countries)
```

3.2.4.13 Update Page

Code 3.61: HTML file of Patients Update Page

```
{% extends "after_login.html" %}
{% block title %}Patients Update{% endblock %}
{% block content %}

<div class="container">
    <h3 class="mx-auto">Update Patients Data</h3>
    {% with messages = get_flashed_messages() %}
    {% if messages %}

    {% for message in messages %}
    <p class="text-danger␣text-start␣fs-3␣fw-bold">{{message}}</p>
    {% endfor %}

    {% endif %}
    {% endwith %}
    <form action="/patients/update" method="POST">
        <p>Country</p>
        <select class="␣form-select␣mb-4" name="country" aria-label
           =".form-select-sm␣example">
            {% if countries %}

            {% for c in countries %}
            <option>{{c[0]}}</option>
            {% endfor %}

            {% endif %}
        </select>
        <div class="mb-3">
            <label for="date" class="form-label">Date</label>
            <p>Please enter date in 'YYYY-MM-DD' format</p>
            <input type="text" name="date" class="form-control" id=
               "total_cases">
        </div>
        <div class="mb-3">
            <label for="icu_patients" class="form-label">icu
               patients</label>
            <input type="text" name="icu_patients" class="form-
               control">
        </div>
        <div class="mb-3">
            <label for="icu_patients_per_million" class="form-label
               ">icu patients per million</label>
```

```
            <input type="text" name="icu_patients_per_million"
               class="form-control">
        </div>
        <div class="mb-3">
            <label for="hosp_patients" class="form-label">hospital
               patients</label>
            <input type="text" name="hosp_patients" class="form-
               control">
        </div>
        <div class="mb-3">
            <label for="hosp_patients_per_million" class="form-
               label">hospital patients per million</label>
            <input type="text" name="hosp_patients_per_million"
               class="form-control">
        </div>
        <div class="mb-3">
            <label for="weekly_icu_admissions" class="form-label">
               weekly icu admissions</label>
            <input type="text" name="weekly_icu_admissions" class="
               form-control">
        </div>
        <div class="mb-3">
            <label for="weekly_icu_admissions_per_million" class="
               form-label">weekly icu admissions per million</label
               >
            <input type="text" name="
               weekly_icu_admissions_per_million" class="form-
               control">
        </div>
        <div class="␣mb-3">
            <label for="weekly_hosp_admissions" class="form-label">
               weekly hospital admissions</label>
            <input type="text" name="weekly_hosp_admissions" class=
               "form-control">
        </div>
        <div class="mb-3">
            <label for="weekly_hosp_admissions_per_million" class="
               form-label">weekly hospital admissions per million</
               label>
            <input type="text" name="
               weekly_hosp_admissions_per_million" class="form-
               control">
        </div>
        <button type="submit" class="btn␣btn-primary">Submit</
           button>
    </form>

</div>
{% endblock %}
```

Code 3.62: View Function of Update Page

```
def update_patients_data():
    connection = hospital_and_icu()
    loc_con = Locations()
    countries = loc_con.get_country_names()
    headings = ("icu_patients", "icu_patients_per_million","
       hosp_patients" ,"hosp_patients_per_million" , "
       weekly_icu_admissions" ,"weekly_icu_admissions_per_million"
       ,"weekly_hosp_admissions" ,"
       weekly_hosp_admissions_per_million")

    isadmin = False
    user_id = str(session["id"])
    if user_id is not None and user_id != "None":
        user = User()
        isadmin = user.isAdmin(user_id)
    else:
        return redirect("/")

    if isadmin == False:
        return redirect("/patients")

    if request.method == "POST":
        if(request.form["country"] and request.form["date"]):
            location_id = request.form["country"]
            location_id = loc_con.get_id_by_country_name(
               location_id)
            date_time = request.form["date"]
        else:
            flash("Both␣country␣and␣date␣fields␣cannot␣be␣blank")
            return render_template("patients/update.html", headings
                = headings, countries = countries)
        if(request.form["icu_patients"] or request.form["
           hosp_patients"]):
            icu_patients = request.form["icu_patients"] if request.
               form["icu_patients"] !="" else "NULL"
            hosp_patients = request.form["hosp_patients"] if
               request.form["hosp_patients"] !="" else "NULL"
        else:
            flash("Either␣icu_patients␣or␣hospital_patients␣cannot␣
               be␣blank!")
            return render_template("patients/update.html", headings
                = headings, countries = countries)
        icu_patients_per_million = request.form["
           icu_patients_per_million"] if request.form["
           icu_patients_per_million"] !="" else "NULL"
        hosp_patients_per_million = request.form["
           hosp_patients_per_million"] if request.form["
           hosp_patients_per_million"] !="" else "NULL"
        weekly_hosp_admissions = request.form["
```

```
        weekly_hosp_admissions"] if request.form["
        weekly_hosp_admissions"] !="" else "NULL"
    weekly_hosp_admissions_per_million = request.form["
        weekly_hosp_admissions_per_million"] if request.form["
        weekly_hosp_admissions_per_million"] !="" else "NULL"
    weekly_icu_admissions = request.form["weekly_icu_admissions
        "] if request.form["weekly_icu_admissions"] !="" else "
        NULL"
    weekly_icu_admissions_per_million = request.form["
        weekly_icu_admissions_per_million"] if request.form["
        weekly_icu_admissions_per_million"] !="" else "NULL"

    country_id_fetched  = loc_con.is_there(location_id[0])

    if country_id_fetched is None:
        flash("Please␣enter␣a␣valid␣country")
        return render_template("patients/update.html", headings
            =headings, countries=countries)

    (country_id,) = country_id_fetched

    format = "%Y-%m-%d"

    try:
        datetime.strptime(date_time, format)
    except ValueError:
        flash("Please␣enter␣a␣valid␣date␣in␣the␣format␣YYYY-MM-
            DD")
        return render_template("patients/update.html", headings
            =headings,countries=countries)

    check_q = connection.is_there(date_time, country_id)
    if check_q is False:
        flash("You␣can␣not␣update␣non-exist␣record")
        return render_template("patients/update.html",headings=
            headings,  countries=countries)

    connection.update(country_id, icu_patients,
        icu_patients_per_million, hosp_patients,
        hosp_patients_per_million, weekly_icu_admissions,
        weekly_icu_admissions_per_million,
        weekly_hosp_admissions,
        weekly_hosp_admissions_per_million, date_time)

    flash("Successfully␣updated")
    return render_template("patients/update.html", headings=
        headings, countries=countries)
else:
    return render_template("patients/update.html", headings=
        headings, countries=countries)
```

3.2.5 Patients - Implemented by Cemalettin Celal Toy

3.2.5.1 Setup Database Codes

Hospital_and_ICU table is created at database and related data imported from csv table and inserted into cases table wit these codes

Code 3.63: Setup file of Hospital_and_ICU table

```
import pandas as pd
import psycopg2

def count_nans(row, df, nan_count):
    count = 0
    for col in df.columns:
        if pd.isna(row[col]):
            if col == "icu_patients" or col == "hosp_patients":
                return False
            count += 1
    return count <= nan_count

def insert_row(cols: list, conn, cursor):
    dataset_df = pd.read_csv("setup/dataset.csv")
    dataset_df = dataset_df.loc[:,cols].drop_duplicates()
    dataset_df = dataset_df[dataset_df.iso_code.apply(lambda row :
       row.split("_")[0]!="OWID")]
    dataset_df = dataset_df[dataset_df.apply(lambda row: count_nans
       (row,dataset_df,2), axis=1)]

    query = """INSERT INTO Hospital_AND_ICU(location_id,
       icu_patients ,icu_patients_per_million,hosp_patients ,
       hosp_patients_per_million ,weekly_icu_admissions ,
       weekly_icu_admissions_per_million ,weekly_hosp_admissions ,
       weekly_hosp_admissions_per_million , date_time) VALUES(%(
       iso_code)s,%(icu_patients)s,
       %(icu_patients_per_million)s,%(hosp_patients)s,%(
       hosp_patients_per_million)s, %(weekly_icu_admissions)s, %(
       weekly_icu_admissions_per_million)s,(weekly_hosp_admissions)
       s, %(weekly_hosp_admissions_per_million)s, %(date)s);"""
    for idx, row in dataset_df.iterrows():
        insert_dict = dict()
        for col in cols:
            if pd.isna(row[col]):
                insert_dict[col] = None
            else:
                insert_dict[col] = row[col]
        cursor.execute(query, insert_dict)
        conn.commit()
```

```
conn = psycopg2.connect(database="postgres",
                        host="localhost",
                        user="postgres",
                        password="1234",
                        port="5432")

queryTable = """DROP TABLE IF EXISTS Hospital_AND_ICU;"""
cursor = conn.cursor()
cursor.execute(queryTable)
conn.commit()

cursor = conn.cursor()

queryTable = """CREATE TABLE Hospital_AND_ICU (
    ID SERIAL primary key,
    location_id varchar(80) references locations (location_id),
    icu_patients integer,
    icu_patients_per_million numeric,
    hosp_patients integer,
    hosp_patients_per_million numeric,
    weekly_icu_admissions integer,
    weekly_icu_admissions_per_million numeric,
    weekly_hosp_admissions integer,
    weekly_hosp_admissions_per_million numeric,
    date_time date
);"""

cursor.execute(queryTable)
conn.commit()
insert_row(["iso_code","icu_patients","icu_patients_per_million","
   hosp_patients","hosp_patients_per_million", "
   weekly_icu_admissions", "weekly_icu_admissions_per_million", "
   weekly_hosp_admissions", "weekly_hosp_admissions_per_million", "
   date"], conn, cursor)
conn.close()
```

3.2.5.2 Model of Hospital_and_ICU table and functions of this model

Code 3.64: Hospital_and_ICU class

```
    import psycopg2

#Celal's table(HOSPITAL_AND_ICU) operation functions
#Connects to the database when it is created
#Closes the connection with the database when it is destructed
class hospital_and_icu:
    #Constructer
    def __init__(self):
        self.connection = None
        self.connect()

    #Destructure
    def __del__(self):
        try:
            self.connection.close()
        except:
            pass

    #connection to the database
    def connect(self):
        self.connection = psycopg2.connect(database="postgres",
                        host="localhost",
                        user="postgres",
                        password="1234",
                        port="5432")

    #Connection control
    def con_control(self):
        try:
            self.connection.status
        except:
            self.connect()
```

Code 3.65: Search functions with Locations_id and date

```
#Selecting by primary key value
def selectFromLOCandDate(self, loc_id, date):
    query = f"""select location_id, date_time, icu_patients ,
    icu_patients_per_million,hosp_patients ,
       hosp_patients_per_million ,
    weekly_icu_admissions ,weekly_icu_admissions_per_million ,
       weekly_hosp_admissions ,
    weekly_hosp_admissions_per_million FROM HOSPITAL_AND_ICU
    WHERE (((icu_patients IS NOT NULL) OR (hosp_patients IS NOT
        NULL)) and
    location_id = '{loc_id}' and date_time = '{date}')"""
    self.con_control()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        result = cursor.fetchall()
    except psycopg2.DatabaseError:
        self.connection.rollback()
        result = None
    finally:
        cursor.close()
        return result
```

Code 3.66: Search functions that returns just id with Locations_id and date

```
def selectFromLOCandDateReturnID(self, loc_id, date):
    query = f"""select id from HOSPITAL_AND_ICU
    WHERE (location_id = '{loc_id}' and date_time = '{date}')
       """
    self.con_control()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        result = cursor.fetchone()
    except psycopg2.DatabaseError:
        self.connection.rollback()
        result = None
    finally:
        cursor.close()
        return result
```

Code 3.67: Search functions with Locations_id

```
    def selectFromLOC(self, location_id, offset):
        query = f"""SELECT location_id, date_time, icu_patients ,
        icu_patients_per_million,hosp_patients ,
           hosp_patients_per_million ,
        weekly_icu_admissions ,weekly_icu_admissions_per_million ,
           weekly_hosp_admissions ,
        weekly_hosp_admissions_per_million FROM HOSPITAL_AND_ICU
        WHERE (((icu_patients IS NOT NULL) OR (hosp_patients IS NOT
            NULL)) and (location_id = '{location_id}'))
        ORDER BY date_time desc OFFSET {offset} ROWS FETCH NEXT 50
           ROWS ONLY;"""
        self.con_control()
        try:
            cursor = self.connection.cursor()
            cursor.execute(query)
            result = cursor.fetchall()
        except psycopg2.DatabaseError:
            self.connection.rollback()
            result = None
        finally:
            cursor.close()
            return result
```

Code 3.68: Search function that returns all data

```
    #Selecting all rows primary key value
    def selectAll(self):
        query = """SELECT location_id, date_time, icu_patients ,
        icu_patients_per_million,hosp_patients ,
           hosp_patients_per_million ,
        weekly_icu_admissions ,weekly_icu_admissions_per_million ,
           weekly_hosp_admissions ,
        weekly_hosp_admissions_per_million FROM HOSPITAL_AND_ICU
        WHERE ((icu_patients IS NOT NULL) OR (hosp_patients IS NOT
           NULL))"""
        self.con_control()
        try:
            cursor = self.connection.cursor()
            cursor.execute(query)
            result = cursor.fetchall()
        except psycopg2.DatabaseError:
            self.connection.rollback()
            result = None
        finally:
            cursor.close()
            return result
```

Code 3.69: Search function that returns all data with offset

```
    def selectAll(self, offset):
        query = f"""SELECT location_id, date_time, icu_patients ,
        icu_patients_per_million,hosp_patients ,
           hosp_patients_per_million ,
        weekly_icu_admissions ,weekly_icu_admissions_per_million ,
           weekly_hosp_admissions ,
        weekly_hosp_admissions_per_million FROM HOSPITAL_AND_ICU
        WHERE ((icu_patients IS NOT NULL) OR (hosp_patients IS NOT
           NULL))
        ORDER BY date_time desc OFFSET {offset} ROWS FETCH NEXT 50
           ROWS ONLY"""
        self.con_control()
        try:
            cursor = self.connection.cursor()
            cursor.execute(query)
            result = cursor.fetchall()
        except psycopg2.DatabaseError:
            self.connection.rollback()
            result = None
        finally:
            cursor.close()
            return result
```

Code 3.70: Delete functions with id

```
    #Deleting a row by id
    def delete(self, id):
        query = """DELETE FROM HOSPITAL_AND_ICU AS H WHERE H.id = %
           s"""
        self.con_control()
        try:
            cursor = self.connection.cursor()
            cursor.execute(query, (id,))
            self.connection.commit()
        except psycopg2.DatabaseError:
            self.connection.rollback()
        finally:
            cursor.close()
```

Code 3.71: Insert functions

```
#Inserting a new row to the table
def insert(self, iso_code, icu_patients,
   icu_patients_per_million, hosp_patients,
   hosp_patients_per_million, weekly_icu_admissions,
   weekly_icu_admissions_per_million, weekly_hosp_admissions,
   weekly_hosp_admissions_per_million, date):

    query = f"""INSERT INTO HOSPITAL_AND_ICU(location_id,
       icu_patients ,
    icu_patients_per_million ,hosp_patients ,
       hosp_patients_per_million ,
    weekly_icu_admissions ,weekly_icu_admissions_per_million ,
       weekly_hosp_admissions ,
    weekly_hosp_admissions_per_million , date_time)
    VALUES('{iso_code}',{icu_patients},{
       icu_patients_per_million},{hosp_patients},
    {hosp_patients_per_million}, {weekly_icu_admissions}, {
       weekly_icu_admissions_per_million},
    {weekly_hosp_admissions}, {
       weekly_hosp_admissions_per_million}, '{date}');"""
    self.con_control()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        self.connection.commit()
    except psycopg2.DatabaseError:
        self.connection.rollback()
    finally:
        cursor.close()
```

Code 3.72: Update functions with date and location_id

```
#Updating a row by id
def update(self, iso_code, icu_patients,
   icu_patients_per_million, hosp_patients,
   hosp_patients_per_million, weekly_icu_admissions,
   weekly_icu_admissions_per_million, weekly_hosp_admissions,
   weekly_hosp_admissions_per_million, date):
    query = f"""UPDATE HOSPITAL_AND_ICU SET (location_id,
       icu_patients ,
    icu_patients_per_million,hosp_patients ,
       hosp_patients_per_million ,
    weekly_icu_admissions ,weekly_icu_admissions_per_million ,
       weekly_hosp_admissions ,
    weekly_hosp_admissions_per_million , date_time)
    = ('{iso_code}',{icu_patients},{icu_patients_per_million},{
       hosp_patients},
    {hosp_patients_per_million}, {weekly_icu_admissions}, {
       weekly_icu_admissions_per_million},
    {weekly_hosp_admissions}, {
```

```
        weekly_hosp_admissions_per_million}, '{date}') WHERE
        HOSPITAL_AND_ICU.date_time = '{date}' and
        HOSPITAL_AND_ICU.location_id = '{iso_code}'"""
    self.con_control()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        self.connection.commit()
    except psycopg2.DatabaseError:
        self.connection.rollback()
    finally:
        cursor.close()
```

Code 3.73: Function that checks if that data exists

```
def is_there(self, date, location):
    query = f"""select count(id) from hospital_and_icu
            where (date_time = '{date}' and location_id = '{
               location}')
            group by date_time, location_id"""

    self.con_control()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        result = cursor.fetchone()
    except psycopg2.DatabaseError:
        self.connection.rollback()
        result = None
    finally:
        cursor.close()
        return True if result is not None and result[0]>0 else
           False
```

Code 3.74: Function that returns all country names from Hospital_and_ICU table's iso_code

```
def get_country_names(self):
    query = """SELECT distinct L.country FROM hospital_and_icu
       as H left join locations as L on(h.location_id = L.
       location_id)
               order by 1;"""
    self.connect()
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        return cursor.fetchall()
    except psycopg2.DatabaseError:
        self.connection.rollback()
    finally:
        cursor.close()
```

3.2.5.3 Patients Page

Code 3.75: HTML file of patients page

```
{% extends "after_login.html" %}
{% block title %}Patients{% endblock %}
{% block content %}
<div class="container-fluid">
    <div class="row␣my-2␣d-flex␣justify-content-between">
        <h3 class="fw-bold␣text-secondary␣col-4">Patients</h3>
        {% if isadmin == True %}
        <div class="col-2">
            <a href="/patients/edit" role="button" aria-pressed="
               true"
                class="btn␣btn-warning␣active␣w-100␣edit-btn">Edit<
                   /a>
        </div>
        {% endif %}
    </div>
    {% with messages = get_flashed_messages() %}
    {% if messages %}

    {% for message in messages %}
    <p class="text-danger␣text-center">{{message}}</p>
    {% endfor %}

    {% endif %}
    {% endwith %}
    <div class="row">
        <div class="col-6"style="width:␣100%;">
            <div class="row␣justify-content-start␣align-items-
               center">
                <div class="col-4" style="width:␣33%;">
                    <select id="sel_country" class="form-select"
                       name="country" aria-label=".form-select-sm␣
                       example">
                        {% if countries %}

                        {% for c in countries %}
                        <option>{{c[0]}}</option>
                        {% endfor %}

                        {% endif %}
                    </select>
                </div>
                <div class="col-4"style="width:␣34%;">
                    <input type="text" name="date" class="form-
                       control" id="date" placeholder="Please␣enter
                       ␣date␣in␣'YYYY-MM-DD'␣format">
                </div>
                <div class="col-4"style="width:␣33%;">
```

```
                    <button type="button" id="filter1" class="btn␣
                       btn-secondary␣w-100␣my-3"><a
                            class="link-light␣text-decoration-none"
                                id="Filter" href="#">Filter</a></
                               button>
                    <script>
                        document.querySelector("#filter1").
                           addEventListener("click", function () {
                            Filter = document.getElementById("
                               Filter")
                            country = document.getElementById("
                               sel_country").value
                            date = document.getElementById("date").
                               value
                            Filter.href = "/patients?"
                            Filter.href += "countryName=" + country
                                + "&dateVariable=" + date
                            window.location.replace(Filter.href)
                        });
                    </script>
                </div>
            </div>
        </div>
    </div>
<div class="row␣my-3␣mx-0">
    <div class="col-12">
        <table class="table␣align-middle␣table-striped␣table-
           hover␣text-center" id="patients">
            <thead>
                <tr>
                    <th scope="col">Country</th>
                    <th scope="col">Date</th>
                    {% for header in headings %}
                    <th scope="col">{{header}}</th>
                    {% endfor %}
                </tr>
            </thead>
            <tbody>
                {% for row in patients %}
                <tr>
                    {% for cell in row %}
                    <td>
                        <div class="data">
                            {{cell}}
                        </div>
                    </td>
                    {% endfor %}
                </tr>
                {% endfor %}
            </tbody>
```

```
            </table>
        </div>
    </div>
    <div class="row">
        <div class="col-6">
            <nav aria-label="Page␣navigation" class="row␣h-100␣d-
               flex␣align-items-center">
                <ul class="pagination␣justify-content-end␣mb-0">
                    <li id="4" class="page-item"><a class="btn␣btn-
                       outline-secondary␣w-100">first</a></li>
                    <li id="1" class="page-item"><a class="btn␣btn-
                       outline-secondary␣w-100">{{paginationValues
                       [0]}}</a></li>
                    <li id="2" class="page-item"><a class="btn␣btn-
                       outline-secondary␣w-100">{{paginationValues
                       [1]}}</a></li>
                    <li id="3" class="page-item"><a class="btn␣btn-
                       outline-secondary␣w-100">{{paginationValues
                       [2]}}</a></li>
                    <script>
                        l1 = document.getElementById("1")
                        l2 = document.getElementById("2")
                        l3 = document.getElementById("3")
                        l4 = document.getElementById("4")

                        let params = (new URL(document.location)).
                           searchParams;
                        let name = params.get("countryName");
                        let url = "/patients?"

                        if (name != null) {
                            url = url + "countryName=" + name + "&"
                        }
                        l1.firstChild.href = url + "pageNumber=" +
                           l1.firstChild.innerText
                        l2.firstChild.href = url + "pageNumber=" +
                           l2.firstChild.innerText
                        l3.firstChild.href = url + "pageNumber=" +
                           l3.firstChild.innerText
                        l4.firstChild.href = url + "pageNumber=1"
                    </script>
                </ul>
            </nav>
        </div>
    </div>
</div>
{% endblock %}
```

Code 3.76: View Function of Patients page

```
def patients_page():
    connection = hospital_and_icu()
    #pagination
    countryName = request.args.get("countryName")
    dateFilter = request.args.get("dateVariable")
    pageNumber =request.args.get("pageNumber") if request.args.get(
       "pageNumber") is not None else "1"
    pageNumber = int(pageNumber)
    offset = (pageNumber-1)*50
    paginationValues = (pageNumber,pageNumber+1,pageNumber+2) if (
       pageNumber)>0 else (0,1,2)

    location = Locations()
    countries = connection.get_country_names()
    patients = None
    isadmin = False
    user_id = str(session["id"])
    if user_id is not None and user_id != "None":
        user = User()
        isadmin = user.isAdmin(user_id)
    else:
        return redirect("/")
    headings = ("icu_patients", "icu_patients_per_million","
       hosp_patients" ,"hosp_patients_per_million" , "
       weekly_icu_admissions" ,"weekly_icu_admissions_per_million"
       ,"weekly_hosp_admissions" ,"
       weekly_hosp_admissions_per_million")
    if(countryName is not None and dateFilter =='’):
        country_id = location.get_id_by_country_name(countryName)
        result = connection.selectFromLOC(country_id[0], offset)
    elif(countryName is not None and dateFilter is not None):
        country_id = location.get_id_by_country_name(countryName)
        result = connection.selectFromLOCandDate(country_id[0],
           dateFilter)
    else:
        result = connection.selectAll(offset)
    patients = np.zeros([1, 10], dtype='str')
    if(result is not None):
        for row in result:
            newRow = np.array(row)
            patients = np.vstack([patients, newRow])

    patients = np.delete(patients, 0, 0)
    return render_template("patients/patients.html",headings=
       headings, isadmin=isadmin,  patients=patients, countries=
       countries, paginationValues=paginationValues)
```

3.2.5.4 Edit Page

Code 3.77: HTML file of Patients Edit Page

```
{% extends "after_login.html" %}
{% block title %}Patients Edit{% endblock %}
{% block content %}
<div class="container-fluid">

    <h3 class="fw-bold␣text-secondary">Patients | Edit</h3>
    {% with messages = get_flashed_messages() %}
    {% if messages %}

    {% for message in messages %}
    <p class="text-danger␣text-center">{{message}}</p>
    {% endfor %}

    {% endif %}
    {% endwith %}
    <div class="row␣w-100␣mt-3␣add-del-btn-cont">
    </div>

    <div class="row">
        <div class="row␣justify-content-start␣align-items-center">
            <div class="col-2" style="width:␣20%;">
                <select id="sel_country" class="form-select" name="
                   country" aria-label=".form-select-sm␣example">
                    {% if countries %}

                    {% for c in countries %}
                    <option>{{c[0]}}</option>
                    {% endfor %}

                    {% endif %}
                </select>
            </div>
            <div class="col-2" style="width:␣20%;">
                <input type="text" name="date" class="form-control"
                    id="date" placeholder="Please␣enter␣date␣in␣'
                   YYYY-MM-DD'␣format">
            </div>
            <div class="col-2" style="width:␣20%;">
                <button type="button" id="filter1" class="btn␣btn-
                   secondary␣w-100"><a
                        class="link-light␣text-decoration-none" id=
                           "Filter" href="#">Filter</a></button>
                <script>
                    document.querySelector("#filter1").
                       addEventListener("click", function () {
                        Filter = document.getElementById("Filter")
```

```
                    country = document.getElementById("
                       sel_country").value
                    date = document.getElementById("date").
                       value
                    Filter.href = "/patients/edit?"
                    Filter.href += "countryName=" + country + "
                       &dateVariable=" + date
                    window.location.replace(Filter.href)
                });
            </script>
        </div>

        <div class="col-2" style="width:␣20%;">
            <a id="update_patients" href="/patients/update"
               class="btn␣btn-secondary␣w-100" role="button"
               aria-pressed="true">Update</a>
        </div>
        <div class="col-2" style="width:␣20%;">
            <a id="add_patients" href="/patients/add" class="
               btn␣btn-secondary␣w-100" role="button" aria-
               pressed="true">Add</a>
        </div>
    </div>
</div>
<div class="row␣my-3␣mx-0">
    <div class="col-12">
        <table class="table␣align-middle␣table-striped␣table-
           hover␣text-center" id="patients">
            <thead>
                <tr>
                    <th scope="col">Country</th>
                    <th scope="col">Date</th>
                    {% for header in headings %}
                    <th scope="col">{{header}}</th>
                    {% endfor %}
                    <th scope="col">Delete</th>
                </tr>
            </thead>
            <tbody>
                {% for row in patients %}
                <tr>
                    {% for cell in row %}
                    <td>
                        <div class="data">
                            {{cell}}
                        </div>
                    </td>
                    {% endfor %}
                    <td>
                        <div class="data">
```

```
                                <button class="btn␣btn-outline-
                                   secondary" href="#" id="delete{{
                                   ␣row[0]␣}}+{{␣row[1]␣}}"
                                value="countryName={{␣row[0]␣}}&
                                   dateVariable={{␣row[1]␣}}&
                                   deleteMode=on" onclick="
                                   deleteQuery(document.
                                   getElementById('delete{{␣row[0]␣
                                   }}+{{␣row[1]␣}}'))"
                                type="button">Delete
                                </button>
                            </div>
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
                <script type="text/javascript">
                    function deleteQuery(row) {
                        if (confirm('Are you sure you want to
                           delete this record?')){
                            const urlStr = window.location.search;
                            if (row.value != "")
                                window.location.href = "/patients/
                                   edit?" + row.value;
                            else
                                window.location.href = "/patients/
                                   edit";
                        }
                    }
                </script>
            </table>
        </div>
    </div>
    <div class="row">
        <nav aria-label="Page␣navigation" class="row␣h-100␣d-flex␣
           align-items-center">
            <ul class="pagination␣justify-content-center␣mb-0">
                <li id="4" class="page-item"><a class="btn␣btn-
                   outline-secondary␣w-100">first</a></li>
                <li id="1" class="page-item"><a class="btn␣btn-
                   outline-secondary␣w-100">{{paginationValues[0]}}
                   </a></li>
                <li id="2" class="page-item"><a class="btn␣btn-
                   outline-secondary␣w-100">{{paginationValues[1]}}
                   </a></li>
                <li id="3" class="page-item"><a class="btn␣btn-
                   outline-secondary␣w-100">{{paginationValues[2]}}
                   </a></li>
                <script>
                    l1 = document.getElementById("1")
```

```
                    l2 = document.getElementById("2")
                    l3 = document.getElementById("3")
                    l4 = document.getElementById("4")

                    let params = (new URL(document.location)).
                       searchParams;
                    let name = params.get("countryName");
                    let url = "/patients/edit?"

                    if (name != null) {
                        url = url + "countryName=" + name + "&"
                    }
                    l1.firstChild.href = url + "pageNumber=" + l1.
                       firstChild.innerText
                    l2.firstChild.href = url + "pageNumber=" + l2.
                       firstChild.innerText
                    l3.firstChild.href = url + "pageNumber=" + l3.
                       firstChild.innerText
                    l4.firstChild.href = url + "pageNumber=1"
                </script>
            </ul>
        </nav>
    </div>
</div>
{% endblock %}
```

Code 3.78: View Function of Edit Page

```
def edit_patients_page():
    connection = hospital_and_icu()
    #pagination
    countryName = request.args.get("countryName")
    dateFilter = request.args.get("dateVariable")
    pageNumber =request.args.get("pageNumber") if request.args.get(
       "pageNumber") is not None else "1"
    pageNumber = int(pageNumber)
    offset = (pageNumber-1)*50
    paginationValues = (pageNumber,pageNumber+1,pageNumber+2) if (
       pageNumber)>0 else (0,1,2)


    isadmin = False
    user_id = str(session["id"])
    if user_id is not None and user_id != "None":
        user = User()
        isadmin = user.isAdmin(user_id)
    else:
        return redirect("/")

    if isadmin == False:
        return redirect("/patients")
```

```
    if(request.args.get("deleteMode") == "on"):
        if(countryName != '' and dateFilter != ''):
            delete_patients(countryName, dateFilter)
        else:
            flash("For␣delete␣operation,␣Date␣or␣Country␣field␣
               cannot␣be␣blank!")
        return redirect("/patients/edit")
    else:
        location = Locations()
        countries = connection.get_country_names()
        patients = None
        headings = ("icu_patients", "icu_patients_per_million","
           hosp_patients" ,"hosp_patients_per_million" , "
           weekly_icu_admissions" ,"
           weekly_icu_admissions_per_million" ,"
           weekly_hosp_admissions" ,"
           weekly_hosp_admissions_per_million")
        if(countryName is not None and dateFilter =='''):
            country_id = location.get_id_by_country_name(
               countryName)
            result = connection.selectFromLOC(country_id[0], offset
               )
        elif(countryName is not None and dateFilter is not None):
            country_id = location.get_id_by_country_name(
               countryName)
            result = connection.selectFromLOCandDate(country_id[0],
                dateFilter)
        else:
            result = connection.selectAll(offset)
        patients = np.zeros([1, 10], dtype='str')
        if(result is not None):
            for row in result:
                newRow = np.array(row)
                patients = np.vstack([patients, newRow])

        patients = np.delete(patients, 0, 0)
        return render_template("patients/edit-patients.html",
           headings=headings,  patients=patients, countries=
           countries, paginationValues=paginationValues)
```

Code 3.79: Delete function

```
def delete_patients(country_id,date):
    connection = hospital_and_icu()
    loc_con = Locations()
    check = connection.selectFromLOCandDateReturnID(country_id,
       date)
    if(check is not None):
        connection.delete(check)
    else:
```

```
        flash("There␣is␣no␣this␣record␣in␣database")
        return redirect("/patients/edit")
```

3.2.5.5 Add Page

Code 3.80: HTML file of Patients Add Page

```
{% extends "after_login.html" %}
{% block title %}Patients Add{% endblock %}
{% block content %}

<div class="container">
    <h3 class="mx-auto">Add New Patients Data</h3>
    {% with messages = get_flashed_messages() %}
    {% if messages %}

    {% for message in messages %}
    <p class="text-danger␣text-start␣fs-3␣fw-bold">{{message}}</p>
    {% endfor %}

    {% endif %}
    {% endwith %}
    <form action="/patients/add" method="POST">
        <div class="mb-3">
            <label for="icu_patients" class="form-label">icu
               patients</label>
            <input type="text" name="icu_patients" class="form-
               control">
        </div>
        <div class="mb-3">
            <label for="icu_patients_per_million" class="form-label
               ">icu patients per million</label>
            <input type="text" name="icu_patients_per_million"
               class="form-control">
        </div>
        <div class="mb-3">
            <label for="hosp_patients" class="form-label">hospital
               patients</label>
            <input type="text" name="hosp_patients" class="form-
               control">
        </div>
        <div class="mb-3">
            <label for="hosp_patients_per_million" class="form-
               label">hospital patients per million</label>
            <input type="text" name="hosp_patients_per_million"
               class="form-control">
        </div>
        <div class="mb-3">
```

```
            <label for="weekly_icu_admissions" class="form-label">
               weekly icu admissions</label>
            <input type="text" name="weekly_icu_admissions" class="
               form-control">
        </div>
        <div class="mb-3">
            <label for="weekly_icu_admissions_per_million" class="
               form-label">weekly icu admissions per million</label
               >
            <input type="text" name="
               weekly_icu_admissions_per_million" class="form-
               control">
        </div>
        <div class="␣mb-3">
            <label for="weekly_hosp_admissions" class="form-label">
               weekly hospital admissions</label>
            <input type="text" name="weekly_hosp_admissions" class=
               "form-control">
        </div>
        <div class="mb-3">
            <label for="weekly_hosp_admissions_per_million" class="
               form-label">weekly hospital admissions per million</
               label>
            <input type="text" name="
               weekly_hosp_admissions_per_million" class="form-
               control">
        </div>
        <p>Country</p>
        <select class="␣form-select␣mb-4" name="country" aria-label
           =".form-select-sm␣example">
            {% if countries %}

            {% for c in countries %}
            <option>{{c[0]}}</option>
            {% endfor %}

            {% endif %}
        </select>
        <div class="mb-3">
            <label for="date" class="form-label">Date</label>
            <p>Please enter date in 'YYYY-MM-DD' format</p>
            <input type="text" name="date" class="form-control" id=
               "date">
        </div>
        <button type="submit" class="btn␣btn-primary">Submit</
           button>
    </form>

</div>
{% endblock %}
```

Code 3.81: View Function of Add Page

```
def add_patients_data():
    connection = hospital_and_icu()
    loc_con = Locations()
    countries = loc_con.get_country_names()
    headings = ("icu_patients", "icu_patients_per_million","
       hosp_patients" ,"hosp_patients_per_million" , "
       weekly_icu_admissions" ,"weekly_icu_admissions_per_million"
       ,"weekly_hosp_admissions" ,"
       weekly_hosp_admissions_per_million")

    isadmin = False
    user_id = str(session["id"])
    if user_id is not None and user_id != "None":
        user = User()
        isadmin = user.isAdmin(user_id)
    else:
        return redirect("/")

    if isadmin == False:
        return redirect("/patients")

    if request.method == "POST":
        if(request.form["country"] and request.form["date"]):
            location_id = request.form["country"]
            location_id = loc_con.get_id_by_country_name(
               location_id)
            date_time = request.form["date"]
        else:
            flash("Both␣country␣and␣date␣fields␣cannot␣be␣blank")
            return render_template("patients/add.html", headings =
               headings, countries = countries)
        if(request.form["icu_patients"] or request.form["
           hosp_patients"]):
            icu_patients = request.form["icu_patients"] if request.
               form["icu_patients"] !="" else "NULL"
            hosp_patients = request.form["hosp_patients"] if
               request.form["hosp_patients"] !="" else "NULL"
        else:
            flash("Either␣icu_patients␣or␣hospital_patients␣cannot␣
               be␣blank!")
            return render_template("patients/add.html", headings =
               headings, countries = countries)
        icu_patients_per_million = request.form["
           icu_patients_per_million"] if request.form["
           icu_patients_per_million"] !="" else "NULL"
        hosp_patients_per_million = request.form["
           hosp_patients_per_million"] if request.form["
           hosp_patients_per_million"] !="" else "NULL"
        weekly_hosp_admissions = request.form["
```

```
        weekly_hosp_admissions"] if request.form["
        weekly_hosp_admissions"] !="" else "NULL"
    weekly_hosp_admissions_per_million = request.form["
        weekly_hosp_admissions_per_million"] if request.form["
        weekly_hosp_admissions_per_million"] !="" else "NULL"
    weekly_icu_admissions = request.form["weekly_icu_admissions
        "] if request.form["weekly_icu_admissions"] !="" else "
        NULL"
    weekly_icu_admissions_per_million = request.form["
        weekly_icu_admissions_per_million"] if request.form["
        weekly_icu_admissions_per_million"] !="" else "NULL"

    country_id_fetched  = loc_con.is_there(location_id[0])

    if country_id_fetched is None:
        flash("Please␣enter␣a␣valid␣country")
        return render_template("patients/add.html", headings=
            headings, countries=countries)

    (country_id,) = country_id_fetched

    format = "%Y-%m-%d"

    try:
        datetime.strptime(date_time, format)
    except ValueError:
        flash("Please␣enter␣a␣valid␣date␣in␣the␣format␣YYYY-MM-
            DD")
        return render_template("patients/add.html", headings=
            headings,countries=countries)

    check_q = connection.is_there(date_time, country_id)
    if check_q:
        flash("You␣can␣not␣add␣a␣new␣record␣into␣an␣already␣
            existing␣record")
        return render_template("patients/add.html",headings=
            headings,  countries=countries)

    connection.insert(country_id, icu_patients,
        icu_patients_per_million, hosp_patients,
        hosp_patients_per_million, weekly_icu_admissions,
        weekly_icu_admissions_per_million,
        weekly_hosp_admissions,
        weekly_hosp_admissions_per_million, date_time)

    flash("Successfully␣created")
    return render_template("patients/add.html", headings=
        headings, countries=countries)
else:
    return render_template("patients/add.html", headings=
```

```
            headings, countries=countries)
```

3.2.5.6 Update Page

Code 3.82: HTML file of Patients Update Page

```
{% extends "after_login.html" %}
{% block title %}Patients Update{% endblock %}
{% block content %}

<div class="container">
    <h3 class="mx-auto">Update Patients Data</h3>
    {% with messages = get_flashed_messages() %}
    {% if messages %}

    {% for message in messages %}
    <p class="text-danger␣text-start␣fs-3␣fw-bold">{{message}}</p>
    {% endfor %}

    {% endif %}
    {% endwith %}
    <form action="/patients/update" method="POST">
        <p>Country</p>
        <select class="␣form-select␣mb-4" name="country" aria-label
           =".form-select-sm␣example">
            {% if countries %}

            {% for c in countries %}
            <option>{{c[0]}}</option>
            {% endfor %}

            {% endif %}
        </select>
        <div class="mb-3">
            <label for="date" class="form-label">Date</label>
            <p>Please enter date in 'YYYY-MM-DD' format</p>
            <input type="text" name="date" class="form-control" id=
               "total_cases">
        </div>
        <div class="mb-3">
            <label for="icu_patients" class="form-label">icu
               patients</label>
            <input type="text" name="icu_patients" class="form-
               control">
        </div>
        <div class="mb-3">
            <label for="icu_patients_per_million" class="form-label
               ">icu patients per million</label>
```

```
            <input type="text" name="icu_patients_per_million"
               class="form-control">
        </div>
        <div class="mb-3">
            <label for="hosp_patients" class="form-label">hospital
               patients</label>
            <input type="text" name="hosp_patients" class="form-
               control">
        </div>
        <div class="mb-3">
            <label for="hosp_patients_per_million" class="form-
               label">hospital patients per million</label>
            <input type="text" name="hosp_patients_per_million"
               class="form-control">
        </div>
        <div class="mb-3">
            <label for="weekly_icu_admissions" class="form-label">
               weekly icu admissions</label>
            <input type="text" name="weekly_icu_admissions" class="
               form-control">
        </div>
        <div class="mb-3">
            <label for="weekly_icu_admissions_per_million" class="
               form-label">weekly icu admissions per million</label
               >
            <input type="text" name="
               weekly_icu_admissions_per_million" class="form-
               control">
        </div>
        <div class="␣mb-3">
            <label for="weekly_hosp_admissions" class="form-label">
               weekly hospital admissions</label>
            <input type="text" name="weekly_hosp_admissions" class=
               "form-control">
        </div>
        <div class="mb-3">
            <label for="weekly_hosp_admissions_per_million" class="
               form-label">weekly hospital admissions per million</
               label>
            <input type="text" name="
               weekly_hosp_admissions_per_million" class="form-
               control">
        </div>
        <button type="submit" class="btn␣btn-primary">Submit</
           button>
    </form>

</div>
{% endblock %}
```

Code 3.83: View Function of Update Page

```
def update_patients_data():
    connection = hospital_and_icu()
    loc_con = Locations()
    countries = loc_con.get_country_names()
    headings = ("icu_patients", "icu_patients_per_million","
       hosp_patients" ,"hosp_patients_per_million" , "
       weekly_icu_admissions" ,"weekly_icu_admissions_per_million"
       ,"weekly_hosp_admissions" ,"
       weekly_hosp_admissions_per_million")

    isadmin = False
    user_id = str(session["id"])
    if user_id is not None and user_id != "None":
        user = User()
        isadmin = user.isAdmin(user_id)
    else:
        return redirect("/")

    if isadmin == False:
        return redirect("/patients")

    if request.method == "POST":
        if(request.form["country"] and request.form["date"]):
            location_id = request.form["country"]
            location_id = loc_con.get_id_by_country_name(
               location_id)
            date_time = request.form["date"]
        else:
            flash("Both␣country␣and␣date␣fields␣cannot␣be␣blank")
            return render_template("patients/update.html", headings
                = headings, countries = countries)
        if(request.form["icu_patients"] or request.form["
           hosp_patients"]):
            icu_patients = request.form["icu_patients"] if request.
               form["icu_patients"] !="" else "NULL"
            hosp_patients = request.form["hosp_patients"] if
               request.form["hosp_patients"] !="" else "NULL"
        else:
            flash("Either␣icu_patients␣or␣hospital_patients␣cannot␣
               be␣blank!")
            return render_template("patients/update.html", headings
                = headings, countries = countries)
        icu_patients_per_million = request.form["
           icu_patients_per_million"] if request.form["
           icu_patients_per_million"] !="" else "NULL"
        hosp_patients_per_million = request.form["
           hosp_patients_per_million"] if request.form["
           hosp_patients_per_million"] !="" else "NULL"
        weekly_hosp_admissions = request.form["
```

```
        weekly_hosp_admissions"] if request.form["
        weekly_hosp_admissions"] !="" else "NULL"
    weekly_hosp_admissions_per_million = request.form["
        weekly_hosp_admissions_per_million"] if request.form["
        weekly_hosp_admissions_per_million"] !="" else "NULL"
    weekly_icu_admissions = request.form["weekly_icu_admissions
        "] if request.form["weekly_icu_admissions"] !="" else "
        NULL"
    weekly_icu_admissions_per_million = request.form["
        weekly_icu_admissions_per_million"] if request.form["
        weekly_icu_admissions_per_million"] !="" else "NULL"

    country_id_fetched  = loc_con.is_there(location_id[0])

    if country_id_fetched is None:
        flash("Please␣enter␣a␣valid␣country")
        return render_template("patients/update.html", headings
            =headings, countries=countries)

    (country_id,) = country_id_fetched

    format = "%Y-%m-%d"

    try:
        datetime.strptime(date_time, format)
    except ValueError:
        flash("Please␣enter␣a␣valid␣date␣in␣the␣format␣YYYY-MM-
            DD")
        return render_template("patients/update.html", headings
            =headings,countries=countries)

    check_q = connection.is_there(date_time, country_id)
    if check_q is False:
        flash("You␣can␣not␣update␣non-exist␣record")
        return render_template("patients/update.html",headings=
            headings,  countries=countries)

    connection.update(country_id, icu_patients,
        icu_patients_per_million, hosp_patients,
        hosp_patients_per_million, weekly_icu_admissions,
        weekly_icu_admissions_per_million,
        weekly_hosp_admissions,
        weekly_hosp_admissions_per_million, date_time)

    flash("Successfully␣updated")
    return render_template("patients/update.html", headings=
        headings, countries=countries)
else:
    return render_template("patients/update.html", headings=
        headings, countries=countries)
```

3.2.6 Vaccinations - Implemented by Hilal Erdoğan

3.2.6.1 Setup Vaccinations Table

To create vaccinations table in database following code is used. This code reads data values according to the names of columns and add into the Vaccinations

Code 3.84: Setup Vaccinations in Database

```
def insert_row(cols: list, conn, cursor):
    dataset_df = pd.read_csv("setup/dataset.csv")
    dataset_df = dataset_df.loc[:,cols].drop_duplicates()
    dataset_df = dataset_df[dataset_df.iso_code.apply(lambda row :
       row.split("_")[0]!="OWID")].dropna()
    # Sutun sayisi kadar %s ekle
    query = """INSERT INTO VACCINATIONS(location_id,
       total_vaccinations, people_vaccinated,
       people_fully_vaccinated, total_boosters, new_vaccinations,
       new_vaccinations_smoothed, date_time)
VALUES(%(iso_code)s,%(total_vaccinations)s, %(people_vaccinated)s
   ,%(people_fully_vaccinated)s,%(total_boosters)s, %(
   new_vaccinations)s, %(new_vaccinations_smoothed)s, %(date)s)"""
    for idx, row in dataset_df.iterrows():
        insert_dict = dict()
        for col in cols:
            if pd.isna(row[col]):
                insert_dict[col] = None
            else:
                insert_dict[col] = row[col]
        cursor.execute(query, insert_dict)
        conn.commit()


conn = psycopg2.connect(database="postgres",
                        host="localhost",
                        user="postgres",
                        password="1234",
                        port="5432")
queryTable = """DROP TABLE IF EXISTS COVID_TESTS;"""
cursor = conn.cursor()
cursor.execute(queryTable)
conn.commit()


queryTable = """CREATE TABLE VACCINATIONS(
    id SERIAL PRIMARY KEY,
    location_id VARCHAR(80) REFERENCES locations(location_id),
    total_vaccinations NUMERIC,
    people_vaccinated NUMERIC,
    people_fully_vaccinated NUMERIC,
    total_boosters NUMERIC,
```

```
    new_vaccinations NUMERIC,
    new_vaccinations_smoothed NUMERIC,
    date_time DATE
);"""
cursor = conn.cursor()
cursor.execute(queryTable)
conn.commit()
insert_row(["iso_code","total_vaccinations","people_vaccinated","
   people_fully_vaccinated","total_boosters", "new_vaccinations","
   new_vaccinations_smoothed", "date"], conn, cursor)
conn.close()
```

3.2.6.2 Model Vaccinations

Code 3.85: Model for Vaccinations

```
import psycopg2 as psg
import numpy as np
from datetime import datetime

# Class for Vaccinations table
# Connects to the db when it is constructed
# Checks db connection before each operation
# Closes connection when it is destructed
class Vaccinations:

    # Initialize object and connect to database
    def __init__(self):
        self.columns = ['location_id', 'total_vaccinations', '
           people_vaccinated', 'people_fully_vaccinated', '
           total_boosters', 'new_vaccinations', '
           new_vaccinations_smoothed', 'date_time']
        self.conn = None
        self.connect()

    # Close connection to the database and destruct
    def __del__(self):
        try:
            self.conn.close()
        except:
            pass

    # Connect to the database
    def connect(self):
        self.conn = psg.connect(database="postgres",
                        host="localhost",
                        user="postgres",
                        password="1234",
                        port="5432")
```

```
    # Check connection and connect again if it is closed
    def check_conn(self):
        try:
            self.conn.status
        except:
            self.connect()

    def get_dates(self,loc,start=-1):
        dates = np.array(self.query_dates(loc,start))
        date_count = dates.shape[0]
        dates_list = dates.reshape(-1,date_count)[0]
        return [date.strftime('%Y-%m-%d') for date in dates_list if
            date is not None]

    def query_dates(self,loc,start):
        loc_str = ""
        start_str = ""
        query = "SELECT␣DISTINCT␣date_time␣FROM␣VACCINATIONS␣AS␣V"
        if loc != "?":
            loc_str = "␣LEFT␣JOIN␣LOCATIONS␣AS␣L␣ON␣V.location_id␣=
               ␣L.location_id␣WHERE␣country␣=␣%(loc_str)s"
        if start != -1:
            start = datetime.strptime(start,'%Y-%m-%d')
            start_str = "␣date_time␣>␣%(start)s"
        query += loc_str
        if loc == "?" and start != -1:
            query += "␣WHERE"
        elif loc != "?" and start != -1:
            query += "␣AND"
        query += start_str
        query += "␣ORDER␣BY␣date_time␣ASC;"
        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query,{'loc_str':loc, 'start':start
               })
            return self.cursor.fetchall()
        except psg.DatabaseError:
            self.conn.rollback()
        finally:
            self.cursor.close()


    def get_location_names(self):
        loc_names = np.array(self.query_location_names())
        loc_count = loc_names.shape[0]
        return [name.replace("␣","_") for name in loc_names.reshape
           (-1,loc_count)[0] if name is not None]
```

```
    def query_location_names(self):
        query = """SELECT DISTINCT country FROM VACCINATIONS AS V
                LEFT JOIN LOCATIONS AS L
                ON V.location_id = L.location_id
                ORDER BY country ASC"""
        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query)
            return self.cursor.fetchall()
        except psg.DatabaseError:
            self.conn.rollback()
        finally:
            self.cursor.close()

    # Read a row by id
    def read_with_id(self, id):
        query = """SELECT * FROM VACCINATIONS AS V WHERE V.id = %s
           ORDER BY V.id;"""
        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query, (id,))
            return self.cursor.fetchone()
        except psg.DatabaseError:
            self.conn.rollback()
        finally:
            self.cursor.close()

    # Read all data
    def read(self):
        query = """SELECT * VACCINATIONS CASES"""
        self.check_conn()
        try:
            cursor = self.conn.cursor()
            cursor.execute(query)
            return cursor.fetchall()
        except psg.DatabaseError:
            self.conn.rollback()
        finally:
            cursor.close()


    def read_filter(self, limit=-1, offset=0, loc_name="?", date="?
       "):
        date_str = ""
        query = "SELECT␣*␣FROM␣VACCINATIONS␣AS␣V"
        if loc_name!="?":
            query += "␣LEFT␣JOIN␣LOCATIONS␣AS␣L␣ON␣V.location_id␣=␣
               L.location_id␣WHERE␣country=%(loc_name)s"
```

```
        if date!="?":
            date_str = "␣date_time␣>=␣TO_DATE(%(date)s,'YYYY-MM-DD
               ')"
        if loc_name=="?" and date!="?":
            query += "␣WHERE" + date_str
        elif loc_name!="?" and date!="?":
            query += "␣AND" + date_str

        query += "␣ORDER␣BY␣V.id␣OFFSET␣%(offset)s"
        if limit != -1:
            query += "␣LIMIT␣" + str(limit)
        query += ";"
        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query, {"offset":str(offset),
                                        "loc_name":loc_name,
                                        "date":date})
            return self.cursor.fetchall()
        except psg.DatabaseError:
            self.conn.rollback()
        finally:
            self.cursor.close()


    # Insert a row into table
    def insert_row(self, location_id, total_vaccinations,
       people_vaccinated, \
         people_fully_vaccinated, total_boosters, new_vaccinations,
             new_vaccinations_smoothed, date_time):
        query = """INSERT INTO VACCINATIONS(location_id,
           total_vaccinations, people_vaccinated,
        people_fully_vaccinated, total_boosters, new_vaccinations,
           new_vaccinations_smoothed, date_time)
        VALUES(%(location_id)s,%(total_vaccinations)s, %(
           people_vaccinated)s,%(people_fully_vaccinated)s,
        %(total_boosters)s, %(new_vaccinations)s, %(
           new_vaccinations_smoothed)s, %(date_time)s)"""

        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query, {
                'location_id': location_id,
                'total_vaccinations': total_vaccinations,
                'people_vaccinated': people_vaccinated,
                'people_fully_vaccinated': people_fully_vaccinated,
                'total_boosters': total_boosters,
                'new_vaccinations': new_vaccinations,
```

```
            'new_vaccinations_smoothed':
               new_vaccinations_smoothed,
            'date_time': date_time
        })
        self.conn.commit()
        return True
    except psg.DatabaseError:
        self.conn.rollback()
        return False
    finally:
        self.cursor.close()


#Update a row by id
def update(self, id, location_id, total_vaccinations,
   people_vaccinated, \
     people_fully_vaccinated, total_boosters, new_vaccinations,
         new_vaccinations_smoothed, date_time):
    query = """UPDATE VACCINATIONS SET(location_id,
       total_vaccinations, people_vaccinated,
    people_fully_vaccinated, total_boosters, new_vaccinations,
       new_vaccinations_smoothed, date_time)
    = (%(location_id)s,%(total_vaccinations)s, %(
       people_vaccinated)s,%(people_fully_vaccinated)s,
    %(total_boosters)s, %(new_vaccinations)s, %(
       new_vaccinations_smoothed)s, %(date_time)s) WHERE
       VACCINATIONS.id = %(id)s"""
    self.check_conn()
    try:
        self.cursor = self.conn.cursor()
        self.cursor.execute(query, {
            'id' : id,
            'location_id': location_id,
            'total_vaccinations': total_vaccinations,
            'people_vaccinated': people_vaccinated,
            'people_fully_vaccinated': people_fully_vaccinated,
            'total_boosters': total_boosters,
            'new_vaccinations': new_vaccinations,
            'new_vaccinations_smoothed':
               new_vaccinations_smoothed,
            'date_time': date_time
        })
        self.conn.commit()
        return True
    except psg.DatabaseError:
        self.conn.rollback()
        return False
    finally:
        self.cursor.close()
```

```
    # Delete a row by id
    def delete(self, id):
        query = """DELETE FROM VACCINATIONS AS V WHERE V.id = %s"""
        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query, (id,))
            self.conn.commit()
            return True
        except psg.DatabaseError:
            self.conn.rollback()
            return False
        finally:
            self.cursor.close()
```

3.2.6.3 View Vaccinations

Code 3.86: View Functions for Pages of Vaccinations

```
from flask import render_template, request, session, redirect
import numpy as np

from model.vaccinations import *
from model.user import *

def vaccinations_page(id = -1):
    user_id = str(session["id"])
    is_admin = False
    if user_id is not None:
        user = User()
        is_admin = user.isAdmin(user_id)

    page_id = request.args.get('page') if request.args.get('page')
       is not None else 1
    loc_name = request.args.get('loc_name') if request.args.get('
       loc_name') is not None else "?"
    date = request.args.get('date') if request.args.get('date') is
       not None else "?"
    table_size = 0

    page_id = int(page_id)

    vaccinations = Vaccinations()
    if id != -1:
        vaccinations.delete(int(id))

    loc_name = loc_name.replace("_"," ")
```

```
    loc_names = vaccinations.get_location_names()
    offset = (page_id-1)*50
    paginationValues = (page_id-1,page_id,page_id+1) if (page_id)>1
        else (1,2,3)


    try:
        covid_data = np.array(vaccinations.read_filter(50,offset,
           loc_name,date))[:,0:9]
        table_size = covid_data.size
    except IndexError:
        covid_data = np.array([[]])
        table_size = 0

    start_dates = vaccinations.get_dates(loc_name)
    headers = [" ".join(head.split("_")).title() for head in
       vaccinations.columns]

    return render_template("vaccinations/vaccinations.html",
       table_headers=headers, table_rows = covid_data, \
        paginationValues=paginationValues, locations = loc_names,
           dates = start_dates, data_available=table_size, is_admin
           =is_admin)

def add_vaccinations_page():
    vaccinations = Vaccinations()
    message = "empty"

    if request.method == "POST":
        location_id = request.form["location_id"]
        total_vaccinations = request.form["total_vaccinations"]
        people_vaccinated = request.form["people_vaccinated"]
        people_fully_vaccinated = request.form["
           people_fully_vaccinated"] if request.form["
           people_fully_vaccinated"] !="" else None
        total_boosters = request.form["total_boosters"] if request.
           form["total_boosters"] !="" else None
        new_vaccinations = request.form["new_vaccinations"] if
           request.form["new_vaccinations"] !="" else None
        new_vaccinations_smoothed = request.form["
           new_vaccinations_smoothed"] if request.form["
           new_vaccinations_smoothed"] !="" else None
        date_time = request.form["date_time"] if request.form["
           date_time"] !="" else None
        result = vaccinations.insert_row(location_id,
           total_vaccinations, people_vaccinated,
           people_fully_vaccinated, total_boosters,
           new_vaccinations, new_vaccinations_smoothed, date_time)
        if result:
            message = "success"
```

```
        else:
            message = "failed"
    return render_template("vaccinations/add-vaccinations.html",
       message=message)

def update_vaccinations_page():

    row_id = request.args.get('id')
    row_id = int(row_id)
    vaccinations = Vaccinations()
    row = np.array(vaccinations.read_with_id(row_id))
    message = "empty"
    if request.method == "POST":
        total_vaccinations = request.form["total_vaccinations"] if
           request.form["total_vaccinations"] !="" else row[2]
        people_vaccinated = request.form["people_vaccinated"] if
           request.form["people_vaccinated"] !="" else row[3]
        people_fully_vaccinated = request.form["
           people_fully_vaccinated"] if request.form["
           people_fully_vaccinated"] !="" else row[4]
        total_boosters = request.form["total_boosters"] if request.
           form["total_boosters"] !="" else row[5]
        new_vaccinations = request.form["new_vaccinations"] if
           request.form["new_vaccinations"] !="" else row[6]
        new_vaccinations_smoothed = request.form["
           new_vaccinations_smoothed"] if request.form["
           new_vaccinations_smoothed"] !="" else row[7]
        date_time = request.form["date_time"] if request.form["
           date_time"] !="" else row[8]
        result = vaccinations.update(row_id, row[1],
           total_vaccinations, people_vaccinated,
           people_fully_vaccinated, total_boosters,
           new_vaccinations, new_vaccinations_smoothed, date_time)
        if result:
            message = "success"
        else:
            message = "failed"

    return render_template("vaccinations/update-vaccinations.html",
        id = row_id, data=row, message=message)
```

3.2.6.4 HTMLs of Vaccinations pages

Vaccinations Page

Code 3.87: Functions for Vacctinations Page

```
    function goNewDirect(page, loc, date) {

        locStr = "";
        dateStr = "";
```

```
        if (loc != null && loc != '')
            locStr = "&loc_name=" + loc;
        if(date != null && date != '')
            dateStr = "&date="+date;
        if (page == null) {
            window.location.href = "/vaccinations?page=1" + locStr+
               dateStr;
        }
        else
        {
            window.location.href = "/vaccinations?page="+page+
               locStr+dateStr;
        }
    }

    function changePage(pageToggle, data_available) {
        const urlStr = window.location.search;
        let urlPage = new URLSearchParams(urlStr);
        page = urlPage.get("page");
        if(urlPage.get("page") == null ||
        (urlPage.get("page") == '1' && pageToggle == 0))
            page = (1).toString();
        else
        {
            if(pageToggle == 0)
             page = (parseInt(urlPage.get("page")) - 1).toString();
            if(pageToggle == 1)
            {
                if(parseInt(data_available) < 50)
                    window.alert("No␣more␣data.");
                else
                    page = (parseInt(urlPage.get("page")) + 1).
                       toString();
            }
            if(pageToggle == -1)
                page = urlPage.get("page");
        }

        goNewDirect(page, urlPage.get('loc_name'),
        urlPage.get("data"));
    }

    function checkIfSelected(element)
    {
        if(element.value == 'Choose...')
        {
            if(element.id == "inputGroupSelect01")
                alert("You␣didn't␣choose␣a␣country␣!!");
            if(element.id == "inputGroupSelect02")
                alert("You␣didn't␣choose␣a␣date␣!!");
```

```
            return false;
        }
        return true;
    }

function filterCountry() {
    let location = document.getElementById("inputGroupSelect01");
    if (checkIfSelected(location) == false)
            return;
    const urlStr = window.location.search;
    let urlPage = new URLSearchParams(urlStr);
    goNewDirect(null, location.value, urlPage.get("date"));
}

function filterDates(button)
 {
      if(checkIfSelected(button) == false)
          return;
      const urlStr = window.location.search;
      let urlPage = new URLSearchParams(urlStr);

      goNewDirect(null, urlPage.get("loc_name"), button.value);
}

function reset(type)
{
   const urlStr = window.location.search;
   let urlPage = new URLSearchParams(urlStr);
   if(type == 'l')
       goNewDirect(urlPage.get("page"), '', urlPage.get("date"));
   if(type == 's')
        goNewDirect(urlPage.get("page"), urlPage.get("loc_name"), '
           ');
}

function deleteRow(row) {
   const urlStr = window.location.search;
   let urlPage = new URLSearchParams(urlStr);
   if (row.value != "")
      window.location.href = "/vaccinations/" + row.value + "?" +
         urlPage;
   else
      window.location.href = "/vaccinations?" + urlPage;
}

function updateRow(row)
{
   const urlStr = window.location.search;
   let urlPage = new URLSearchParams(urlStr);
     if(row.value == "")
```

```
        window.location.href = "/vaccinations/"+row.value+"?"+
           urlPage;
     else
        window.location.href = "/update-vaccinations?id="+row.value
           ;
}
```

Code 3.88: Vacctinations Page

```
<div class="container">
    <div class="row␣d-flex␣mt-2">
        <label for="inputGroupSelect01" class="form-country">
            <h6 style="display:␣inline">Choose a country: </h6>
            <p style="display:␣inline" id="form-country">
                <script>
                    const urlStrCt = window.location.search;
                    let urlPageCt = new URLSearchParams(urlStrCt);
                    if(urlPageCt.get("loc_name") == null)
                        document.getElementById("form-country").
                           textContent = "";
                    else
                        document.getElementById("form-country").
                           textContent = "Selected␣country␣>␣"+
                           urlPageCt.get("loc_name");
                </script>
            </p>
        </label>
<div class="input-group␣col␣mt-3">
   <select class="form-select" id="inputGroupSelect01" aria-label="
      Example␣select␣with␣button␣addon">
  <option selected>Choose...</option>
    {% for loc in locations%}
   <option value={{loc}}>{{loc}}</option>
    {% endfor %}
     </select>
     <button style="width:␣70px;justify-content:␣center;" class="
        btn␣btn-outline-secondary␣" type="button" onclick="
        filterCountry()">Filter</button>
    <button style="width:␣70px;justify-content:␣center;" class="btn
       ␣btn-outline-secondary" id="resetGroupButton02" onclick="
       reset('l')"
             type="button">Reset</button>
        </div>

        <div class="col">
            {% if is_admin %}
            <nav aria-label="Page␣navigation␣example">
                <ul class="pagination␣justify-content-end␣mb-0">
                    <li class="page-item"><button style="width:␣132
                       px;" class="btn␣btn-warning" type="button"
                        onclick="document.location.href␣=␣'/add-
```

```
                        vaccinations'␣">Add</button></li>
            </ul>
        </nav>
        {% endif %}
    </div>
</div>

<div class="row␣d-flex␣mt-3">
    <label for="inputGroupSelect02" class="form-label">
        <h6 style="display:␣inline;">Choose a date: </h6>
        <p style="display:␣inline;" id="form-date">
            <script>
                const urlStrDate = window.location.search;
                let urlPageDate = new URLSearchParams(
                   urlStrDate);
                if(urlPageDate.get("date") == null)
                    document.getElementById("form-date").
                       textContent = "";
                else
                    document.getElementById("form-date").
                       textContent = "Selected␣date␣>␣"+
                       urlPageDate.get("date");
            </script>
        </p>
    </label>
    <div class="input-group␣col␣mt-1">
        <select class="form-select" id="inputGroupSelect02"
           aria-label="Example␣select␣with␣button␣addon">
          <option selected>Choose...</option>
        {% for date in dates %}
           <option value={{date}}>{{date}}</option>
        {% endfor %}
        </select>
        <button style="width:␣70px;justify-content:␣center;"
           class="btn␣btn-outline-secondary" id="
           inputGroupButton02" onclick="filterDates(document.
           getElementById('inputGroupSelect02'))"
         type="button">Filter</button>
         <button style="width:␣70px;justify-content:␣center;"
            class="btn␣btn-outline-secondary" id="
            resetGroupButton02" onclick="reset('s')"
         type="button">Reset</button>
    </div>

    <div class="col">
        <nav aria-label="Page␣navigation␣example">
            <ul class="pagination␣justify-content-end␣mb-0">
                <li class="page-item"><button class="btn␣btn-
                   outline-secondary" id="prev-button" value="
                   {{data_available}}" style="margin-right:␣5px
```

```
                    ;"
                     onclick="changePage(0,document.
                        getElementById('prev-button').value)"
                        type="button">PREV</button></li>
                <li class="page-item"><button class="btn␣btn-
                   outline-secondary"  id="next-button" value="
                   {{data_available}}"
                     onclick="changePage(1,document.
                        getElementById('next-button').value)"
                        type="button">NEXT</button></li>
            </ul>
        </nav>
    </div>
</div>
<div class="row␣my-3␣mx-0">
    <table class="table␣align-middle␣table-striped␣table-hover␣
       text-center" id="data-table">
        <thead>
          <tr class="align-bottom">

            {% for head in table_headers %}
            <th scope="col">{{head}}</th>
            {% endfor %}
            {% if is_admin %}
                <th scope="col" class="edit-buttons">Delete</th
                   >
                <th scope="col" class="edit-buttons">Update</th
                   >
            {% endif %}
          </tr>
        </thead>
        <tbody>
            {% if data_available %}
                {% for row in table_rows%}
                <tr class="align-items-center">
                    {% for cell in row[1:] %}
                    <td class="align-items-center">
                        <div class="data">
                            {{cell}}
                        </div>
                    </td>
                    {% endfor %}
                    {% if is_admin %}
                    <td class="edit-buttons">
                        <div class="data">
                            <button class="btn␣btn-outline-
                               secondary" style="background-
                               color:␣red;" href="#"
                                    id="delete{{␣row[0]␣}}"
                                       value="{{␣row[0]␣}}"
```

```
                                                onclick="deleteRow(document
                                                    .getElementById('delete
                                                    {{␣row[0]␣}}'))" type="
                                                    button"><i
                                                class="bi␣bi-trash" style="
                                                    color:␣white;"></i>
                                            </button>
                                        </div>
                                    </td>
                                    <td class="edit-buttons">
                                        <div class="data">
                                            <button class="btn␣btn-outline-
                                                secondary" type="button"
                                            href="#" id="update{{␣row[0]␣}}"
                                                value="{{␣row[0]␣}}"
                                            onclick="updateRow(document.
                                                getElementById('update{{␣row[0]␣
                                                }}'))" "><i
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣class="bi-arrow-right"␣style="color: blue;"></i>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</button>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</div>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</td>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣{%␣endif␣%}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</tr>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣{%␣endfor␣%}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣{%␣endif␣%}
␣␣␣␣␣␣␣␣␣␣␣</tbody>
␣␣␣␣␣␣␣␣</table>
␣␣␣␣</div>
</div>
```

Code 3.89: Functions for Update Vaccinations

Update Vaccinations Page

```
    const message = '{{message}}';
    if (message != "empty") {
        if (message == "success") {
            window.alert("Data␣successfully␣updated.");
            window.location.href = "/vaccinations";
        }
        else {
            window.alert("Data␣could␣not␣be␣updated.␣Please␣control
                ␣the␣values␣and␣try␣again!.");
            const urlStr = window.location.search;
            let urlPage = new URLSearchParams(urlStr);
            window.location.href = "/update-vaccinations?id="+
                urlPage.get("id");
        }
    }
```

Code 3.90: Update Vaccinations Page

```
<h2 class="container␣my-4">Update Vaccinations</h2>
<div class="container␣my-4">
    <form action="/update-vaccinations?id={{id}}" method="POST">
        <div class="mb-3">
            <label for="total_vaccinations" class="form-label">
               Total Vaccinations</label>
            <div class="d-flex">
                <input type="text" name="total_vaccinations" class=
                   "form-control" style="width:␣30%;" id="
                   total_vaccinations"
                    aria-describedby="emailHelp" placeholder="{{
                       data[2]}}">
                </div>
        </div>
        <div class="mb-3">
            <label for="people_vaccinated" class="form-label">
               People Vaccinated</label>
            <div class="d-flex">
                <input type="text" name="people_vaccinated" class="
                   form-control" style="width:␣30%;" id="
                   people_vaccinated"
                    aria-describedby="emailHelp" placeholder="{{
                       data[3]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="people_fully_vaccinated" class="form-label"
               >People Fully Vaccinated</label>
            <div class="d-flex">
                <input type="text" name="people_fully_vaccinated"
                   class="form-control" style="width:␣30%;"
                    id="people_fully_vaccinated" aria-describedby="
                       emailHelp" placeholder="{{data[4]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="total_boosters" class="form-label">Total
               Boosters</label>
            <div class="d-flex">
                <input type="text" name="total_boosters" class="
                   form-control" style="width:␣30%;"
                    id="total_boosters" aria-describedby="emailHelp
                       " placeholder="{{data[5]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="new_vaccinations" class="form-label">New
               Vaccinations</label>
            <div class="d-flex">
```

```
                <input type="text" name="new_vaccinations" class="
                   form-control" style="width:␣30%;"
                    id="new_vaccinations" aria-describedby="
                       emailHelp" placeholder="{{data[6]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="new_vaccinations_smoothed" class="form-
               label">New Vaccinations Smoothed</label>
            <div class="d-flex">
                <input type="text" name="new_vaccinations_smoothed"
                    class="form-control" style="width:␣30%;" id="
                   new_vaccinations_smoothed"
                    aria-describedby="emailHelp" placeholder="{{
                       data[7]}}">
            </div>
        </div>
        <div class="mb-3">
            <label for="date_time" class="form-label">Date (yyyy-mm
               -dd)</label>
            <div class="d-flex">
                <input type="text" name="date_time" class="form-
                   control" style="width:␣30%;" id="date_time"
                    aria-describedby="emailHelp" placeholder="{{
                       data[8]}}">
            </div>
        </div>
        <button type="submit" class="btn␣btn-primary">Update Data</
           button>
        <button type="button" href="#" onclick="document.location.
           href=␣'/vaccinations'" class="btn␣btn-primary">Back</
           button>
    </form>
</div>
```

Code 3.91: Functions for Add Vaccinations Page

Add Vaccinations Page

```
    const message = '{{message}}';

    if (message != "empty") {
        if (message == "success") {
            window.alert("Data␣successfully␣added.");
            window.location.href = "/vaccinations";
        }
        else {
            window.alert("Data␣could␣not␣be␣added.␣Please␣control␣
               the␣values␣and␣try␣again!.");
            window.location.href = "/add-vaccinations";
        }
    }
```

Code 3.92: Add Vaccinations Page

```
<h2 class="container␣my-4">Add Vaccinations Data</h2>
<div class="container␣my-4">
    <form action="/add-vaccinations" method="POST">
        <div class="mb-3">
            <label for="location_id" class="form-label">Location Id
               </label>
            <div class="d-flex">
                <input type="text" name="location_id" class="form-
                   control" style="width:␣30%;" id="location_id"
                    aria-describedby="emailHelp">
            </div>
        </div>
        <div class="mb-3">
            <label for="total_vaccinations" class="form-label">
               Total Vaccinations</label>
            <div class="d-flex">
                <input type="text" name="total_vaccinations" class=
                   "form-control" style="width:␣30%;" id="
                   total_vaccinations"
                    aria-describedby="emailHelp">
                </div>
        </div>
        <div class="mb-3">
            <label for="people_vaccinated" class="form-label">
               People Vaccinated</label>
            <div class="d-flex">
                <input type="text" name="people_vaccinated" class="
                   form-control" style="width:␣30%;" id="
                   people_vaccinated"
                    aria-describedby="emailHelp">
            </div>
        </div>
        <div class="mb-3">
            <label for="people_fully_vaccinated" class="form-label"
               >People Fully Vaccinated</label>
            <div class="d-flex">
                <input type="text" name="people_fully_vaccinated"
                   class="form-control" style="width:␣30%;"
                    id="people_fully_vaccinated" aria-describedby="
                       emailHelp">
            </div>
        </div>
        <div class="mb-3">
            <label for="total_boosters" class="form-label">Total
               Boosters</label>
            <div class="d-flex">
                <input type="text" name="total_boosters" class="
```

```
                    form-control" style="width:␣30%;"
                     id="total_boosters" aria-describedby="emailHelp
                        ">
            </div>
        </div>
        <div class="mb-3">
            <label for="new_vaccinations" class="form-label">New
               Vaccinations</label>
            <div class="d-flex">
                <input type="text" name="new_vaccinations" class="
                   form-control" style="width:␣30%;"
                    id="new_vaccinations" aria-describedby="
                       emailHelp">
            </div>
        </div>
        <div class="mb-3">
            <label for="new_vaccinations_smoothed" class="form-
               label">New Vaccinations Smoothed</label>
            <div class="d-flex">
                <input type="text" name="new_vaccinations_smoothed"
                    class="form-control" style="width:␣30%;" id="
                   new_vaccinations_smoothed"
                    aria-describedby="emailHelp">
            </div>
        </div>
        <div class="mb-3">
            <label for="date_time" class="form-label">Date (yyyy-mm
               -dd)</label>
            <div class="d-flex">
                <input type="text" name="date_time" class="form-
                   control" style="width:␣30%;" id="date_time"
                    aria-describedby="emailHelp">
            </div>
        </div>
        <button type="submit" class="btn␣btn-primary">Update Data</
           button>
        <button type="button" href="#" onclick="document.location.
           href=␣'/vaccinations'" class="btn␣btn-primary">Back</
           button>
    </form>
</div>
```

3.2.7 Locations - Implemented by all team members together

3.2.7.1 Setup Locations Table

c

Code 3.93: Setup Locations table in DB

```
import psycopg2
import pandas as pd

def insert_row(cols: list, conn, cursor):
    dataset_df = pd.read_csv("setup/dataset.csv")
    dataset_df = dataset_df.loc[:,cols].drop_duplicates()
    dataset_df = dataset_df[dataset_df.iso_code.apply(lambda row :
       row.split("_")[0]!="OWID")]
    # S tun  say s  kadar %s ekle
    query = """INSERT INTO Locations VALUES(%(iso_code)s,%(
       continent)s,%(location)s,%(population)s,%(aged_65_older)s,
                                        %(aged_70_older)s,%(
                                           median_age)s)"""
    for idx, row in dataset_df.iterrows():
        insert_dict = dict()
        for col in cols:
            if pd.isna(row[col]):
                insert_dict[col] = None
            else:
                insert_dict[col] = row[col]
        cursor.execute(query, insert_dict)
        conn.commit()

conn = psycopg2.connect(database="postgres",
                        host="localhost",
                        user="postgres",
                        password="1234",
                        port="5432")

query = """DROP TABLE IF EXISTS Locations;"""
cursor = conn.cursor()
cursor.execute(query)
conn.commit()


query = """CREATE TABLE Locations (
    location_id VARCHAR(80) PRIMARY KEY,
    continent VARCHAR(80),
    country VARCHAR(80),
    population BIGINT,
    rate_age_65_older NUMERIC,
    rate_age_70_older NUMERIC,
    median_age NUMERIC );"""

cursor = conn.cursor()
cursor.execute(query)
conn.commit()
insert_row(["iso_code","continent","location","population","
   aged_65_older","aged_70_older","median_age"], conn, cursor)
```

```
conn.close()
```

3.2.7.2 Model - Functions with Queries

Code 3.94: Function constructor of Locations class

```
def __init__(self):
        self.columns = ["location_id","continent","location","
           population","aged_65_older","aged_70_older","median_age"
           ]
        self.conn = None
        self.connect()
        self.cursor = None
```

Code 3.95: Function to create connection with DB

```
def connect(self):
        self.conn = psycopg2.connect(database="postgres",
                        host="localhost",
                        user="postgres",
                        password="1234",
                        port="5432")
```

Code 3.96: Function controlling status of connection

```
def check_conn(self):
        try:
            self.conn.status
        except:
            self.connect()
```

Code 3.97: Function closing connection with DB

```
def __del__(self):
        try:
            self.conn.close()
        except:
            pass
```

Code 3.98: Function finding data by primary key

```
def find_by_id(self, location_id):
        query = """SELECT * FROM LOCATIONS L WHERE L.location_id =
           %s"""
        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query, (location_id,))
            return self.cursor.fetchone()
        except psycopg2.DatabaseError:
            self.conn.rollback()
```

```
        finally:
            self.cursor.close()
```

Code 3.99: Function finding all rows

```
def find_all(self):
        query = """SELECT * FROM LOCATIONS"""
        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query)
            return self.cursor.fetchall()
        except psycopg2.DatabaseError:
            self.conn.rollback()
        finally:
            self.cursor.close()
```

Code 3.100: Function deleting a row by id

```
def delete(self, location_id):
        query = """DELETE FROM LOCATIONS L WHERE L.location_id = %s
           """
        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query, (location_id,))
            self.conn.commit()
        except psycopg2.DatabaseError:
            self.conn.rollback()
        finally:
            self.cursor.close()
```

Code 3.101: Function inserting a new row to the table

```
def save(self, location_id, country, population, aged_65_older,
   aged_70_older, median_age, handwashing_facilities):

        query = """INSERT INTO LOCATIONS(location_id, country,
           population, aged_65_older,
            aged_70_older, median_age, handwashing_facilities)
        VALUES(%(location_id)s,%(country)s,%(population)s,%(
           aged_65_older)s,
            %(aged_70_older)s,%(median_age)s,%(
               handwashing_facilities)s)"""
        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query, {
                'location_id': location_id,
                'country': country,
                'population': population,
                'aged_65_older': aged_65_older,
```

```
                'age_70_older': aged_70_older,
                'median_age': median_age,
                'handwashing_facilities': handwashing_facilities
            })
            self.conn.commit()
        except psycopg2.DatabaseError:
            self.conn.rollback()
        finally:
            self.cursor.close()
```

Code 3.102: Function updating a row by id

```
def update(self, location_id, country, population, aged_65_older,
   aged_70_older, median_age, handwashing_facilities):
        query = """UPDATE LOCATIONS SET (location_id, country,
           population, aged_65_older,
            aged_70_older, median_age, handwashing_facilities) =
            (%(location_id)s,%(country)s,%(population)s,%(
               aged_65_older)s,
            %(aged_70_older)s,%(median_age)s,%(
               handwashing_facilities)s) WHERE LOCATIONS.
               location_id = %(location_id)s"""
        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query, {
                'location_id': location_id,
                'country': country,
                'population': population,
                'aged_65_older': aged_65_older,
                'age_70_older': aged_70_older,
                'median_age': median_age,
                'handwashing_facilities': handwashing_facilities
            })
            self.conn.commit()
        except psycopg2.DatabaseError:
            self.conn.rollback()
        finally:
            self.cursor.close()
```

Code 3.103: Function getting country names

```
def get_country_names(self):
        query = """SELECT L.country FROM LOCATIONS L"""
        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query)
            return self.cursor.fetchall()
        except psycopg2.DatabaseError:
            self.conn.rollback()
        finally:
```

```
            self.cursor.close()
```

Code 3.104: Function getting id by country name

```
def get_id_by_country_name(self, country):
        query = """SELECT L.location_id FROM LOCATIONS L WHERE L.
           country = %s"""
        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query, (country,))
            return self.cursor.fetchone()
        except psycopg2.DatabaseError:
            self.conn.rollback()
        finally:
            self.cursor.close()
```

Code 3.105: Function controlling data availability

```
def is_there(self, l_id):
        query = f"""SELECT LOCATION_ID FROM LOCATIONS
                WHERE (LOCATION_ID = '{l_id}')"""

        self.check_conn()
        try:
            self.cursor = self.conn.cursor()
            self.cursor.execute(query)
            return self.cursor.fetchone()
        except psycopg2.DatabaseError:
            self.conn.rollback()
        finally:
            self.cursor.close()
```

3.2.7.3 View

Code 3.106: Flask function of Locations Page

```
def locations_page():
    chart_paths = os.path.join('static', 'charts')
    locations_table = Locations()
    loc_count = np.array(locations_table.get_country_names()).shape
       [0]
    loc_list = np.array(locations_table.get_country_names()).
       reshape(-1,loc_count)[0]
    return render_template("locations/locations.html", locations =
       loc_list, charts=[chart_paths+i for i in ["\\aged_65_older.
       png",\
                                                                  "
```

Code 3.107: Flask function of Locatio Information Page

```
def location_page(loc_name):
    locations_table = Locations()
    loc_id = locations_table.get_id_by_country_name(loc_name)
    if loc_id == None:
        return render_template("locations/location.html",
           location_info = {"location":-1})
    loc_info = locations_table.find_by_id(loc_id[0])
    print(dict(zip(locations_table.columns,loc_info)))
    return render_template("locations/location.html", location_info
        = dict(zip(locations_table.columns,loc_info)))
```

3.2.7.4 HTML Pages

Code 3.108: HTML Form of Locatios Page

```
{% extends "after_login.html" %}
{% block title %}Country{% endblock %}
{% block content %}

<script>
  function goToLocation(selection) {
    let location_name = inputLocSelect.value;
    if(location_name == 'Choose...')
      alert("Select␣a␣country␣!!!");
    else
      window.location.href = "/locations/" + location_name;
```

```
  }
</script>


<div class="row">
  <div class="d-flex␣justify-content-center">
    <div class="w-75␣p-3">
      <div class="col-55">
        <div class="row␣justify-content-start␣align-items-center">
          <div class="input-group">
            <select class="form-select" id="inputLocSelect" aria-
               label="Example␣select␣with␣button␣addon">
              <option selected value='Choose...'>Choose...</option>
            {% for loc in locations %}
                <option value={{loc}}>{{loc}}</option>
            {% endfor %}
            </select>
            <button style="width:␣70px;justify-content:␣center;"
               class="btn␣btn-outline-secondary" id="inputLocButton
               " onclick="goToLocation(document.getElementById('
               inputLocSelect'))"
              type="button">Filter</button>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

<div class="d-flex␣justify-content-center␣mb-4␣my-4">
  <button class="selfButton␣position-relative␣btn␣btn-dark␣" style=
     "margin-right:␣10px;" type="button"
    data-bs-target="#carouselExampleControls" data-bs-slide="prev">
    <span class="carousel-control-prev-icon"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="selfButton␣position-relative␣btn␣btn-dark" type="
     button" data-bs-target="#carouselExampleControls"
    data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></
       span>
    <span class="visually-hidden">Next</span>
  </button>
</div>

<div id="carouselExampleControls" class="carousel␣slide" data-bs-
   ride="carousel">
  <div style="margin-bottom:␣10%;" class="carousel-inner">
    <div class="carousel-item␣active">
      <div class="container">
```

```
      <div class="row">
        <div class="col-lg-6␣col-md-12">
          <div class="card">
            <img src="{{charts[0]}}" class="card-img-top" alt="
               Fissure␣in␣Sandstone" />
            <div class="card-body">
              <h5 class="card-title">Percentage of People Older
                 Than 65 By Continent</h5>
            </div>
          </div>
        </div>

        <div class="col-lg-6␣d-none␣d-lg-block">
          <div class="card">
            <img src="{{charts[1]}}" class="card-img-top" alt="
               Storm␣Clouds" />
            <div class="card-body">
              <h5 class="card-title">Percentage of People Older
                 Than 70 By Continent</h5>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
  <div class="carousel-item">
    <div class="container">
      <div class="row">
        <div class="col-lg-6␣col-md-12">
          <div class="card">
            <img src="{{charts[2]}}" class="card-img-top" alt="
               Fissure␣in␣Sandstone" />
            <div class="card-body">
              <h5 class="card-title">Population Means By
                 Continent</h5>
            </div>
          </div>
        </div>

        <div class="col-lg-6␣d-none␣d-lg-block">
          <div class="card">
            <img src="{{charts[3]}}" class="card-img-top" alt="
               Storm␣Clouds" />
            <div class="card-body">
              <h5 class="card-title">Median Ages By Continent</h5
                 >
              </p>
            </div>
          </div>
        </div>
```

```
        </div>
      </div>
    </div>
  </div>
</div>
</div>
{% endblock %}
```

Code 3.109: HTML Form of Location Information Page

```
{% extends "after_login.html" %}
{% block title %}Country{% endblock %}
{% block content %}
<div class="container">
    {% if location_info['location'] == -1 %}
    <h1>Data could not be found.</h1>
    {% else %}
    <div class="card␣text-center">
        <div class="mx-0␣card-header␣fw-bold␣row␣d-flex␣justify-
           content-center">
            <h3 class="col-12">{{ location_info['location'] }}</h3>
        </div>
        <div class="card-body">
            <div class="country-info-container␣mt-3␣mb-5">
                <h5 class="fs-6␣card-title"><span class="fw-bold␣me
                   -2">Population:</span> {{ location_info['
                   population']
                    }}
                </h5>
                <h5 class="fs-6␣card-title"><span style="display:␣
                   inline;" class="fw-bold␣me-2">age_65_older: </
                   span>
                    {% if location_info['age_65_older'] == null %}
                        <p style="display:␣inline;" >None</p>
                    {% else %}
                        <p style="display:␣inline;" >{{
                           location_info['age_65_older'] }}</p>
                    {% endif %}

                </h5>
                <h5 class="fs-6␣card-title"><span style="display:␣
                   inline;" class="fw-bold␣me-2">age_70_older: </
                   span>
                    {% if location_info['age_70_older'] == null %}
                        <p style="display:␣inline;" >None</p>
                    {% else %}
                        <p style="display:␣inline;" >{{
                           location_info['age_70_older'] }}</p>
                    {% endif %}

                </h5>
```

```
                <h5 class="fs-6␣card-title"><span style="display:␣
                   inline;" class="fw-bold␣me-2">median_age: </span
                   >
                    {% if location_info['median_age'] == null %}
                        <p style="display:␣inline;" >None</p>
                    {% else %}
                        <p style="display:␣inline;" >{{
                           location_info['median_age'] }}</p>
                    {% endif %}

                </h5>
                <h5 class="fs-6␣card-title"><span style="display:␣
                   inline;" class="fw-bold␣me-2">continent: </span>
                    {% if location_info['continent'] == null %}
                        <p style="display:␣inline;" >None</p>
                    {% else %}
                        <p style="display:␣inline;" >{{
                           location_info['continent'] }}</p>
                    {% endif %}

                </h5>
            </div>
        </div>
    </div>
    {% endif %}

    <div class="col-md-12␣text-center">
        <div class="w-100␣p-3">
            <div>
                <a href="{{␣url_for('locations_page')␣}}" class="
                   btn␣btn-warning␣active␣col-2␣me-2" role="button"
                    aria-pressed="true">Go
                    back</a>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

3.2.8 HTML Templates - Implemented by all team members together

3.2.8.1 Before Login HTML template

Code 3.110: HTML Template File of Before Login Pages

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="utf-8" />
  <title>Covid19|{% block title %}{% endblock %}</title>
  <!-- Title of the web page changes dynamically  "Covid19|Home"
     or   "Covid19|Global" -->
  <!-- Import the Bootstrap stylesheet -->
  <link rel="stylesheet"
    href="{{␣url_for('static',␣filename='extentions/bootstrap
       -5.2.3-dist/css/bootstrap.min.css')␣}}">
  <!-- Import our custom stylesheet -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/
     libs/font-awesome/6.0.0-beta3/css/all.min.css"
    integrity="sha512-Fo3rlrZj/
       k7ujTnHg4CGR2D7kSs0v4LLanw2qksYuRlEzO+tcaEPQogQ0KaoGN26/
       zrn20ImR1DfuLWnOo7aBA=="
    crossorigin="anonymous" referrerpolicy="no-referrer" />
  <link rel="stylesheet" href="../static/customs/tests.css" />
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css
     /bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js
     /bootstrap.bundle.min.js"></script>
</head>

<body>
  <header>
    <nav class="navbar␣navbar-expand-lg␣navbar-light␣bg-light">
      <div class="container">
        <div class="row␣w-100␣d-flex␣justify-between">
          <div class="col-12">
            <div class="row␣d-flex">
              {% if isHome %}
              <div class="col-2␣d-flex␣justify-content-center">
                <a class="navbar-brand␣me-0" href="{{␣url_for('
                   home_page')}}"> COVID19</a>
              </div>
              <div class="col-8"></div>
              <div class="col-2␣d-flex␣justify-content-center">
                <a class="btn␣btn-danger␣me-0␣justify-end␣mb-0"
                   href="{{␣url_for('login_page')}}">Log out</a>
              </div>
              {% else %}
              <a class="navbar-brand␣col-2␣me-0" href="{{␣url_for('
                 login_page')}}"> COVID19</a>
              {% endif %}
            </div>
          </div>
        </div>
      </div>
    </nav>
  </header>
```

```
  <div class="content-container">
    {% block content %}{% endblock %}
    <!-- This content part is changing according to Results -->
    <!-- Jinja template engine renders it -->
  </div>


  {% block script %}

  <!-- Import Jquery -->
  {% endblock %}
  <footer style="background-color:␣#1f76b4;" class="py-3␣my-4">
    <p style="color:␣white;" class="text-center">   2022 ITU-DB2218
      </p>
  </footer>

</html>
```

3.2.8.2 After Login HTML template

Code 3.111: HTML Template File of After Login Pages

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <title>Covid19|{% block title %}{% endblock %}</title>
  <!-- Title of the web page changes dynamically  "Covid19|Home"
     or   "Covid19|Global" -->
  <!-- Import the Bootstrap stylesheet -->
  <link rel="stylesheet"
    href="{{␣url_for('static',␣filename='extentions/bootstrap
      -5.2.3-dist/css/bootstrap.min.css')␣}}">
  <!-- Import our custom stylesheet -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/
     libs/font-awesome/6.0.0-beta3/css/all.min.css"
    integrity="sha512-Fo3rlrZj/
      k7ujTnHg4CGR2D7kSs0v4LLanw2qksYuRlEzO+tcaEPQogQ0KaoGN26/
      zrn20ImR1DfuLWnOo7aBA=="
    crossorigin="anonymous" referrerpolicy="no-referrer" />
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css
     /bootstrap.min.css" rel="stylesheet"
    integrity="sha384-EVSTQN3/
      azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
      crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js
     /bootstrap.bundle.min.js"
```

```
    integrity="sha384-MrcW6ZMFYlzcLA8Nl+
       NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
    crossorigin="anonymous"></script>
  <link rel="stylesheet" href="../static/customs/tests.css" />
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/
     bootstrap-icons@1.3.0/font/bootstrap-icons.css">
  <!--Importing bootstrap icons-->
</head>

<body>
  <header>
    <nav class="navbar␣navbar-expand-lg␣navbar-light␣bg-light␣d-
       flex␣align-items-center">
      <a style="margin-left:␣5px;" class="navbar-brand␣col-1␣me-0"
         href="{{␣url_for('home_page')}}"> COVID19</a>
      <div class="container">
        <div class="row␣w-100␣d-flex␣justify-between">
          <div class="col-12">
            <div class="row␣d-flex">
              <a class="navbar-brand␣col-2␣me-0" href="{{␣url_for('
                 locations_page')}}">Countries</a>
              <a class="navbar-brand␣col-2␣me-0" href="{{␣url_for('
                 patients_page')}}">Patients</a>
              <a class="navbar-brand␣col-2␣me-0" href="{{␣url_for('
                 cases_page')}}">Cases</a>
              <a class="navbar-brand␣col-2␣me-0" href="{{␣url_for('
                 tests_page')}}">Tests</a>
              <a class="navbar-brand␣col-2␣me-0" href="{{␣url_for('
                 deaths_page')}}">Deaths</a>
              <a class="navbar-brand␣col-2␣me-0" href="{{␣url_for('
                 vaccinations_page')}}">Vaccinations</a>
            </div>
          </div>
        </div>
      </div>
      <a style="margin-right:␣5px;" class="btn␣btn-danger␣col-2␣me
         -0␣justify-content-end␣mb-0" href="{{␣url_for('login_page
         ')}}">Log out</a>
    </nav>
  </header>

  <div class="content-container">
    {% block content %}{% endblock %}
    <!-- This content part is changing according to Results -->
    <!-- Jinja template engine renders it -->
  </div>

  <footer style="background-color:␣#1f76b4;" class="py-3␣my-4">
    <p style="color:␣white;" class="text-center">   2022 ITU-DB2218
       </p>
```

```
  </footer>

  {% block script %}

  <!-- Import Jquery -->
{% endblock %}
</body>

</html>
```