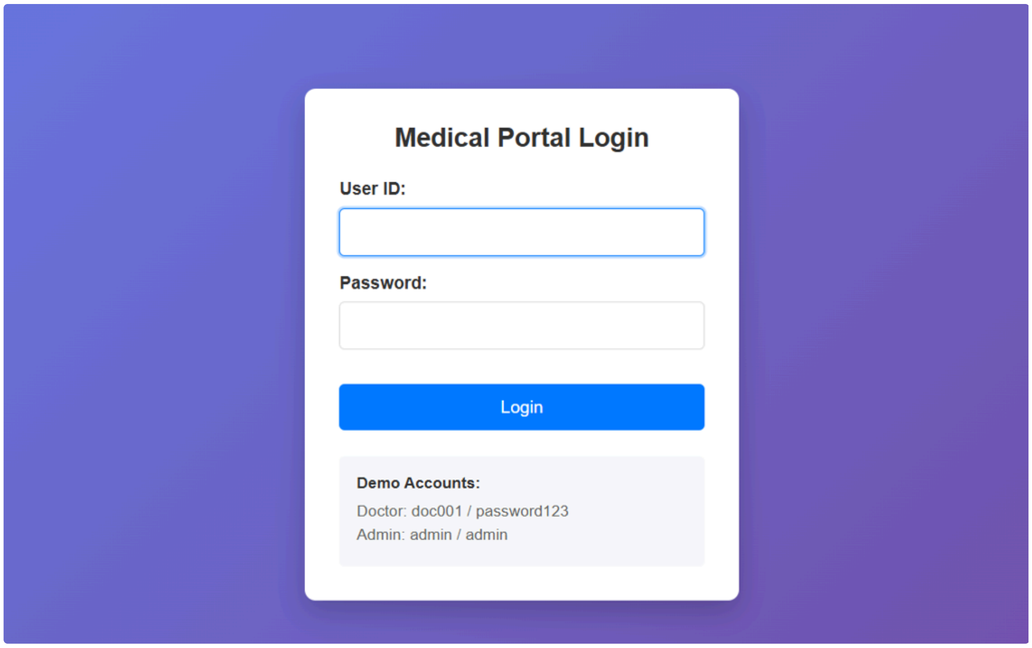
Documentation

All documentation here:

Emily: My role in the project is to develop the website component. We have chosen React as our front-end framework. The website is designed exclusively for use by medical professionals and administrators, allowing them to view their patients and access patient results. So far, I have successfully created the main layout and structure for the website. I have built the initial main page, which is a login portal. This is where a doctor or administrator will log in using their unique ID. Upon authentication, they are redirected to a dashboard. For a doctor, the dashboard presents a list of available actions, such as viewing all their assigned patients and accessing or comparing the quiz results for each patient. An administrator sees a similar dashboard but with an additional option to manage user accounts. This includes the ability to create, delete, or update doctor and patient profiles. When the "View Patients" option is selected, the user is navigated to a patients page that displays a list of all their patients. Clicking on an individual profile then directs them to that specific patient's detailed profile page. I have also developed a quiz results page, which is designed to display each patient's results from the mobile application quiz. Currently, this page is filled with temporary data to demonstrate its final look and functionality. Additionally, I have implemented a settings page, enabling the logged-in user to manage their own account details and customize their dashboard preferences. Finally, I have set up a configuration file to handle the connection between the website and our Firebase database. While the integration is not yet fully complete, finalizing this connection is one of the objectives for me for the next sprint.



Medical Portal Login

User ID:

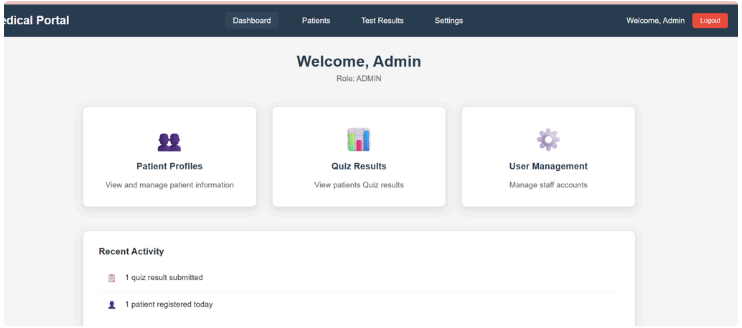
Password:

Demo Accounts:

Doctor: doc001 / password123

Admin: admin / admin

Login Page



Welcome, Admin

Role: ADMIN

Patient Profiles

View and manage patient information

Quiz Results

View patients Quiz results

User Management

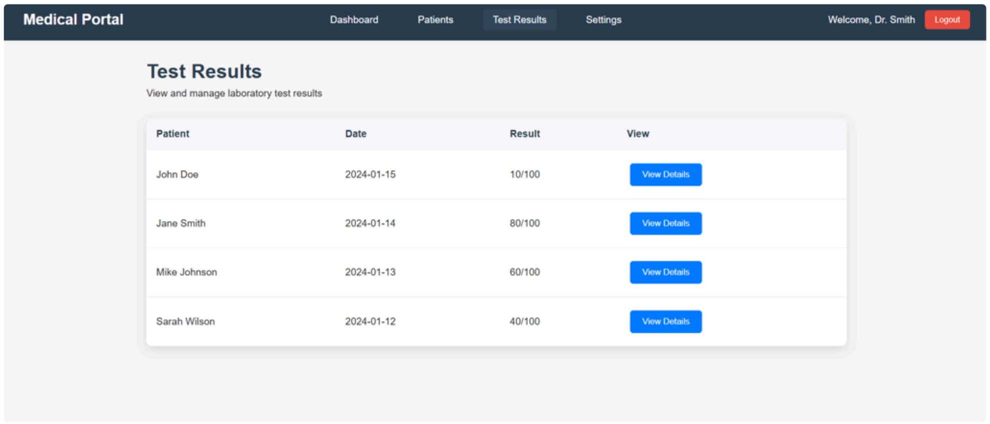
Manage staff accounts

Recent Activity

1 quiz result submitted

1 patient registered today

Dashboard for Admin

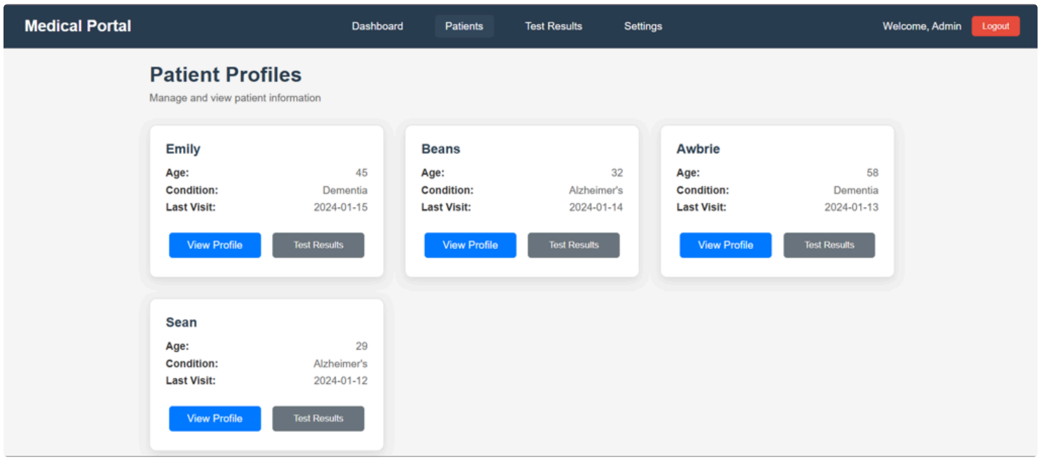


Test Results

View and manage laboratory test results

Patient	Date	Result	View
John Doe	2024-01-15	10/100	View Details
Jane Smith	2024-01-14	80/100	View Details
Mike Johnson	2024-01-13	60/100	View Details
Sarah Wilson	2024-01-12	40/100	View Details

Patient’s Quiz Results



Patient Profiles

Manage and view patient information

Emily

Age:	45
Condition:	Dementia
Last Visit:	2024-01-15

Beans

Age:	32
Condition:	Alzheimer's
Last Visit:	2024-01-14

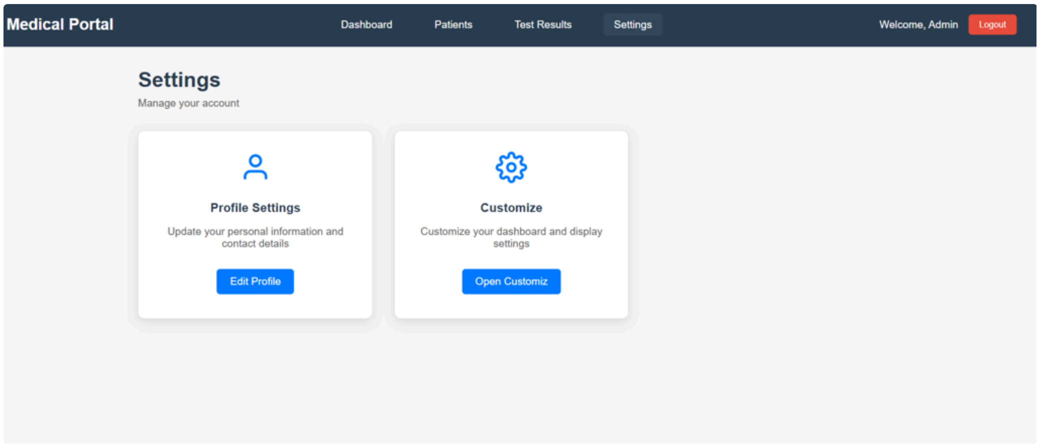
Awbrie

Age:	58
Condition:	Dementia
Last Visit:	2024-01-13

Sean

Age:	29
Condition:	Alzheimer's
Last Visit:	2024-01-12

Patients Page



Settings

Manage your account

Profile Settings

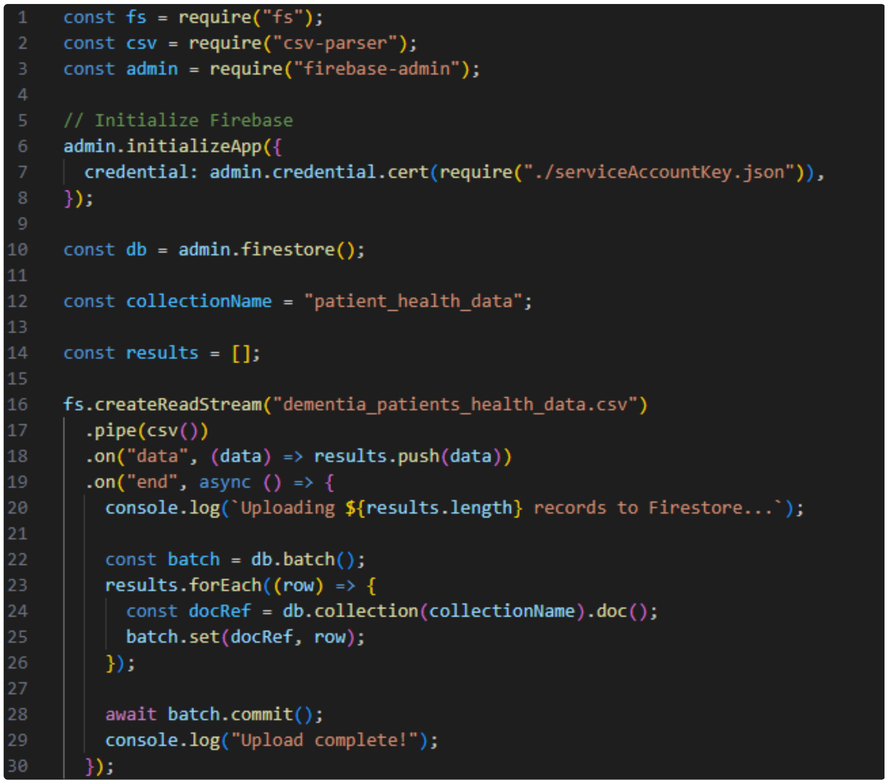
Update your personal information and contact details

Customize

Customize your dashboard and display settings

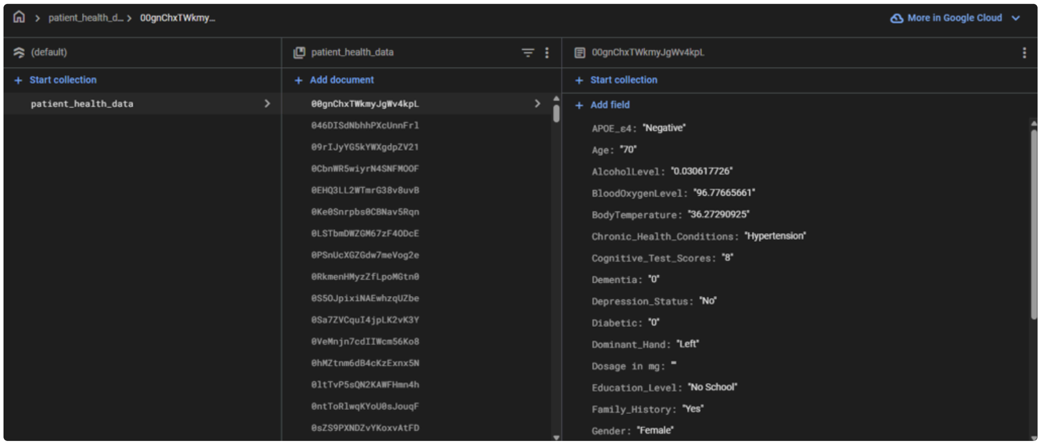
Settings Page

Sean: To store and manage our project's data, we used Firebase Firestore by Google. I started by creating a new Firebase project in the Firebase console, then enabled Cloud Firestore so we could store the data in the cloud. I made every member of our group a owner of the Firebase database which allows us to edit it as we see fit. To automate the data upload process, I made a folder that included a Node.js script called import.js and a JSON key file. The script uses 2 main packages: firebase-admin and csv-parser, to download it use the command npm install firebase-admin csv-parser.



```
const fs = require("fs");
const csv = require("csv-parser");
const admin = require("firebase-admin");

// Initialize Firebase
admin.initializeApp({
  credential: admin.credential.cert(require("./serviceAccountKey.json")),
});

const db = admin.firestore();

const collectionName = "patient_health_data";

const results = [];

fs.createReadStream("dementia_patients_health_data.csv")
  .pipe(csv())
  .on("data", (data) => results.push(data))
  .on("end", async () => {
    console.log(`Uploading ${results.length} records to Firestore...`);

    const batch = db.batch();
    results.forEach((row) => {
      const docRef = db.collection(collectionName).doc();
      batch.set(docRef, row);
    });

    await batch.commit();
    console.log("Upload complete!");
  });
```

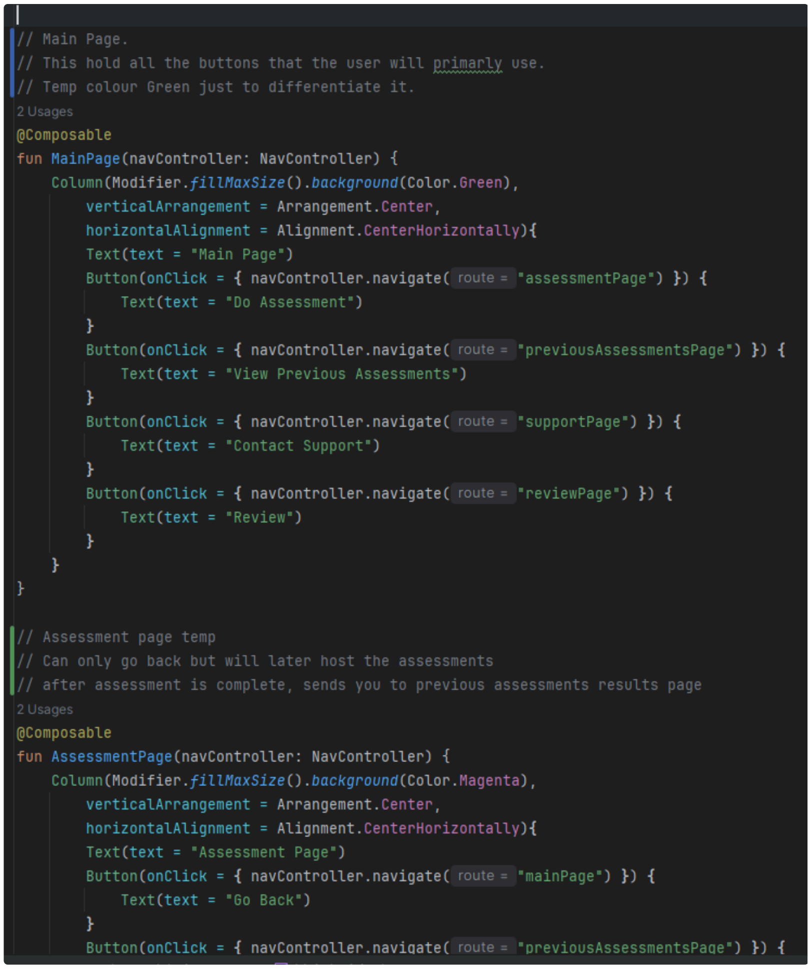
After setting everything up, I ran the script using node import.js and it connected to the Firestore database and uploaded all the CSV rows as documents. Each column in the CSV became a corresponding field in Firestore.



```
00gnChxTWkmyJgWv4kpL
046DISdNbhhPXcUnnFrl
09rIJyYG5kYWXgdpZV21
0CbnWR5wiyrN4SNFMOOF
0EHQ3LL2WTmrG38v8uvB
0Ke0SnrpbsOCBNav5Rqn
0LSTbmDWZGM67zF40DcE
0PSnUcXGZGdw7meVog2e
0RkmenHMyzZfLpoMGtn0
0S5OJpixiNAEwhzqUZbe
0Sa7ZVCquI4jpLK2vK3Y
0VeMnjn7cdIIWcm56Ko8
0hMZtnm6dB4cKzExnx5N
0ltTvP5sQN2KAWFHmn4h
0ntToRlwqKYoU0sJouqF
0sZS9PXNDZvYKoxvAtFD
```

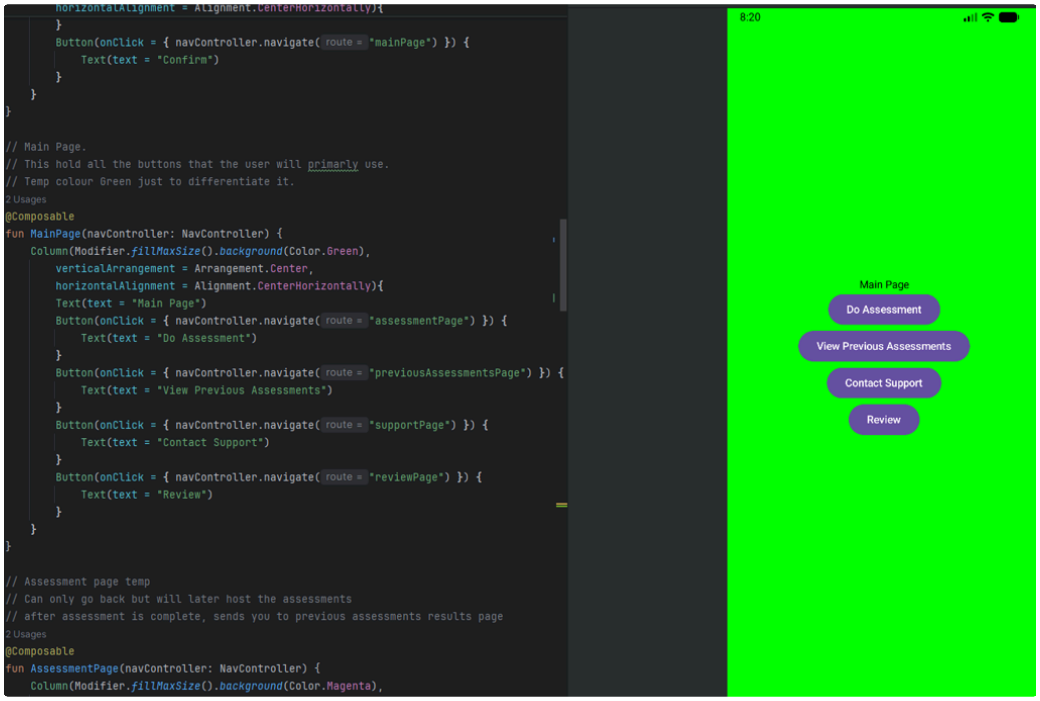
```
APOE_ε4: "Negative"
Age: "70"
AlcoholLevel: "0.030617726"
BloodOxygenLevel: "96.77665661"
BodyTemperature: "36.27290925"
Chronic_Health_Conditions: "Hypertension"
Cognitive_Test_Scores: "8"
Dementia: "0"
Depression_Status: "No"
Diabetic: "0"
Dominant_Hand: "Left"
Dosage in mg: ""
Education_Level: "No School"
Family_History: "Yes"
Gender: "Female"
```

Mikolaj (Aubrey): For this project I was assigned the mobile application. This application uses Kotlin and Android Studio. The app will be a way to provide tests remotely to the patients. The tests will have results calculated with an LLM. For this sprint the bones of the application was created with almost every page being navigable to.



```
// Main Page.
// This hold all the buttons that the user will primarly use.
// Temp colour Green just to differentiate it.
2 Usages
@Composable
fun MainPage(navController: NavController) {
    Column(Modifier.fillMaxSize().background(Color.Green),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally){
        Text(text = "Main Page")
        Button(onClick = { navController.navigate( route = "assessmentPage") }) {
            Text(text = "Do Assessment")
        }
        Button(onClick = { navController.navigate( route = "previousAssessmentsPage") }) {
            Text(text = "View Previous Assessments")
        }
        Button(onClick = { navController.navigate( route = "supportPage") }) {
            Text(text = "Contact Support")
        }
        Button(onClick = { navController.navigate( route = "reviewPage") }) {
            Text(text = "Review")
        }
    }
}

// Assessment page temp
// Can only go back but will later host the assessments
// after assessment is complete, sends you to previous assessments results page
2 Usages
@Composable
fun AssessmentPage(navController: NavController) {
    Column(Modifier.fillMaxSize().background(Color.Magenta),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally){
        Text(text = "Assessment Page")
        Button(onClick = { navController.navigate( route = "mainPage") }) {
            Text(text = "Go Back")
        }
        Button(onClick = { navController.navigate( route = "previousAssessmentsPage") }) {
```

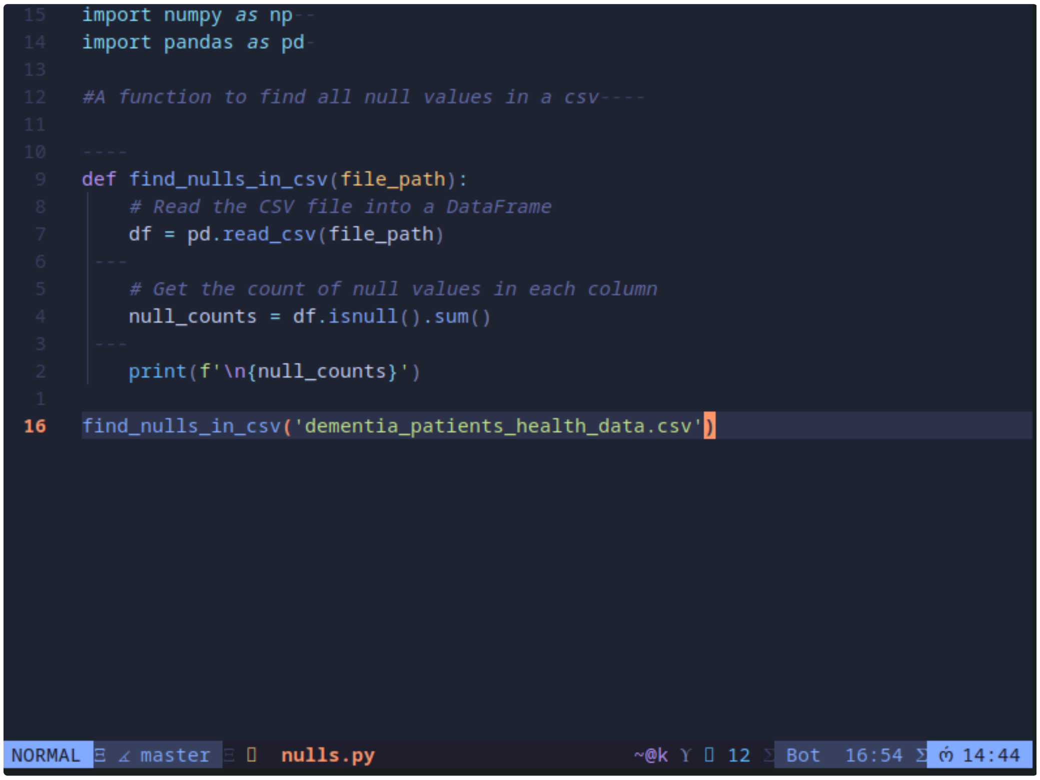
The pages currently hold work in progress colours as to differentiate them.



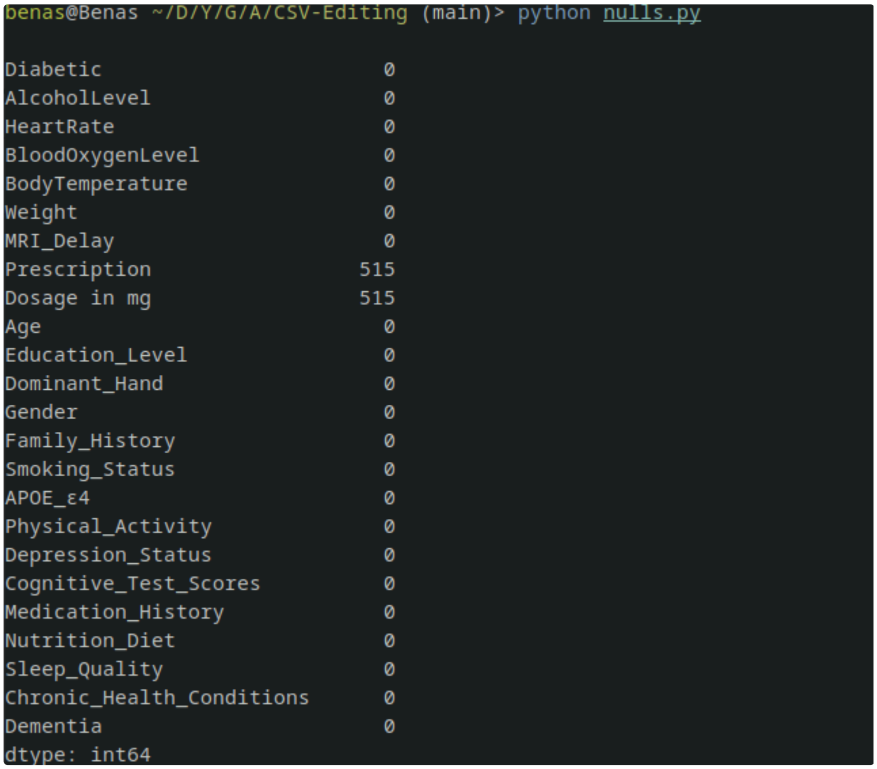
```
        horizontalAlignment = Alignment.CenterHorizontally){
        }
        Button(onClick = { navController.navigate( route = "mainPage") }) {
            Text(text = "Confirm")
        }
    }
}

// Main Page.
// This hold all the buttons that the user will primarly use.
// Temp colour Green just to differentiate it.
2 Usages
@Composable
fun MainPage(navController: NavController) {
    Column(Modifier.fillMaxSize().background(Color.Green),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally){
        Text(text = "Main Page")
        Button(onClick = { navController.navigate( route = "assessmentPage") }) {
            Text(text = "Do Assessment")
        }
        Button(onClick = { navController.navigate( route = "previousAssessmentsPage") }) {
            Text(text = "View Previous Assessments")
        }
        Button(onClick = { navController.navigate( route = "supportPage") }) {
            Text(text = "Contact Support")
        }
        Button(onClick = { navController.navigate( route = "reviewPage") }) {
            Text(text = "Review")
        }
    }
}

// Assessment page temp
// Can only go back but will later host the assessments
// after assessment is complete, sends you to previous assessments results page
2 Usages
@Composable
fun AssessmentPage(navController: NavController) {
    Column(Modifier.fillMaxSize().background(Color.Magenta),
```

The next sprint will focus on creating a reusable format and layout for the pages so they look better and provide some functionality. There will also be a link between the website and mobile app.

Benas: Our plan is in the upcoming sprints to implement an LLM chat for GPs and admin users that can see the patients results and predict a dementia/alzheimers diagnosis. Before starting on that I wrote a few python scripts to get a better grasp of the data we are dealing with. Firstly i wrote a script to identify which columns may have null values.



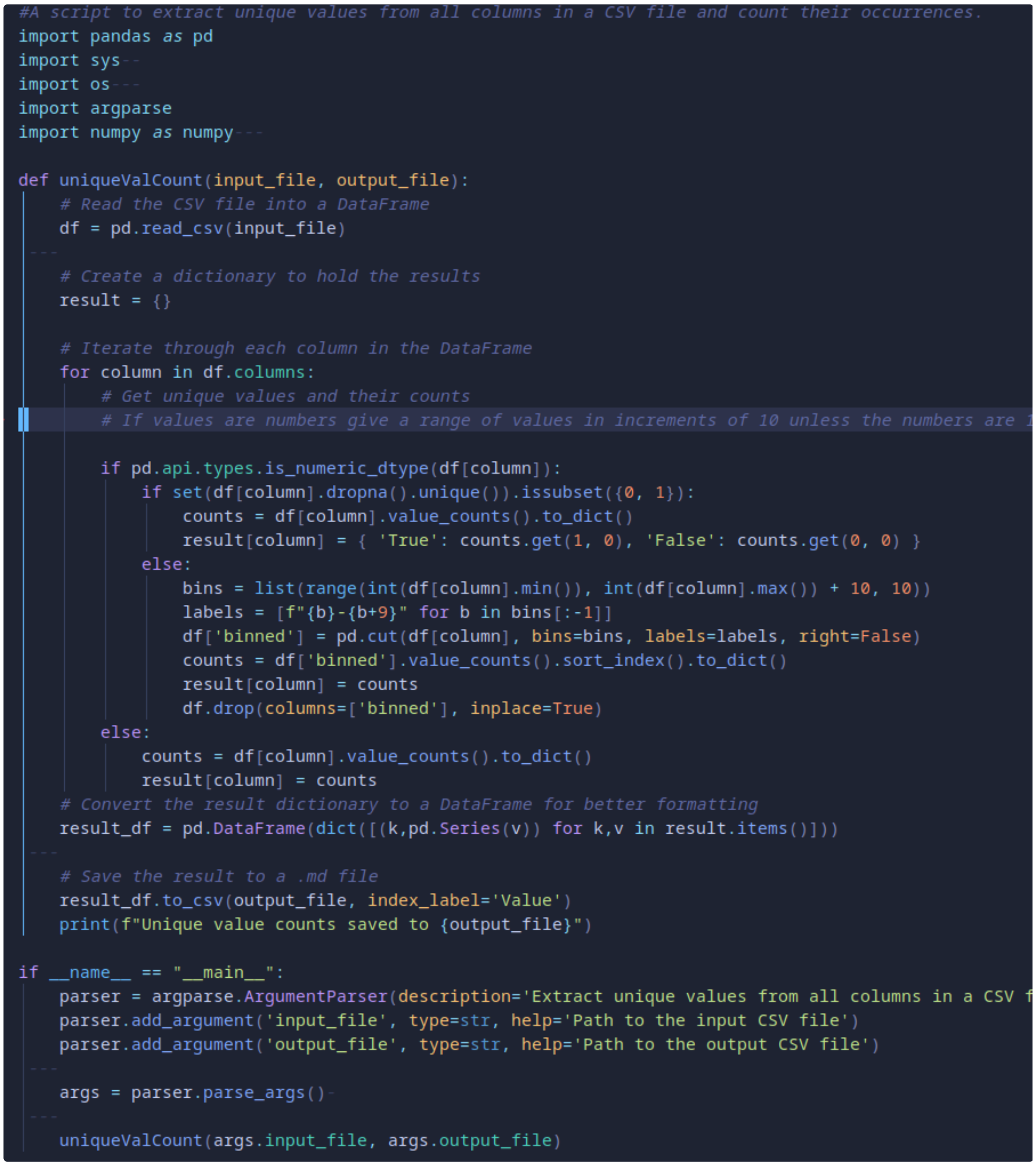
```
import numpy as np
import pandas as pd

#A function to find all null values in a csv


def find_nulls_in_csv(file_path):
    # Read the CSV file into a DataFrame
    df = pd.read_csv(file_path)

    # Get the count of null values in each column
    null_counts = df.isnull().sum()

    print(f'\n{null_counts}')

find_nulls_in_csv('dementia_patients_health_data.csv')
```



```
benas@Benas ~/D/Y/G/A/CSV-Editing (main)> python nulls.py

Diabetic                       0
AlcoholLevel                   0
HeartRate                      0
BloodOxygenLevel               0
BodyTemperature                0
Weight                         0
MRI_Delay                      0
Prescription                 515
Dosage in mg                 515
Age                            0
Education_Level                0
Dominant_Hand                  0
Gender                         0
Family_History                 0
Smoking_Status                 0
APOE_ε4                        0
Physical_Activity              0
Depression_Status              0
Cognitive_Test_Scores          0
Medication_History             0
Nutrition_Diet                 0
Sleep_Quality                  0
Chronic_Health_Conditions      0
Dementia                       0
dtype: int64
```

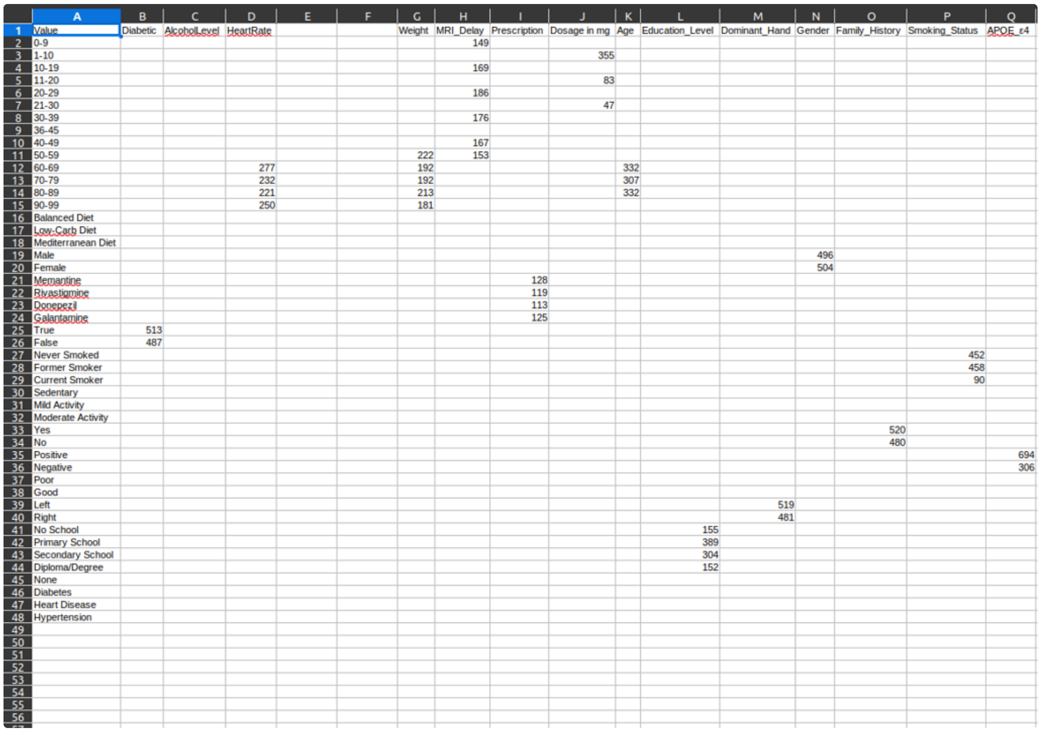
As not everyone is on a prescription and therefore does not have a dosage tied to that prescription, i determined that null values will not make a significant impact to these columns.

Next i wrote a script to count the frequency of all unique values in the CSV and save them to a new one.



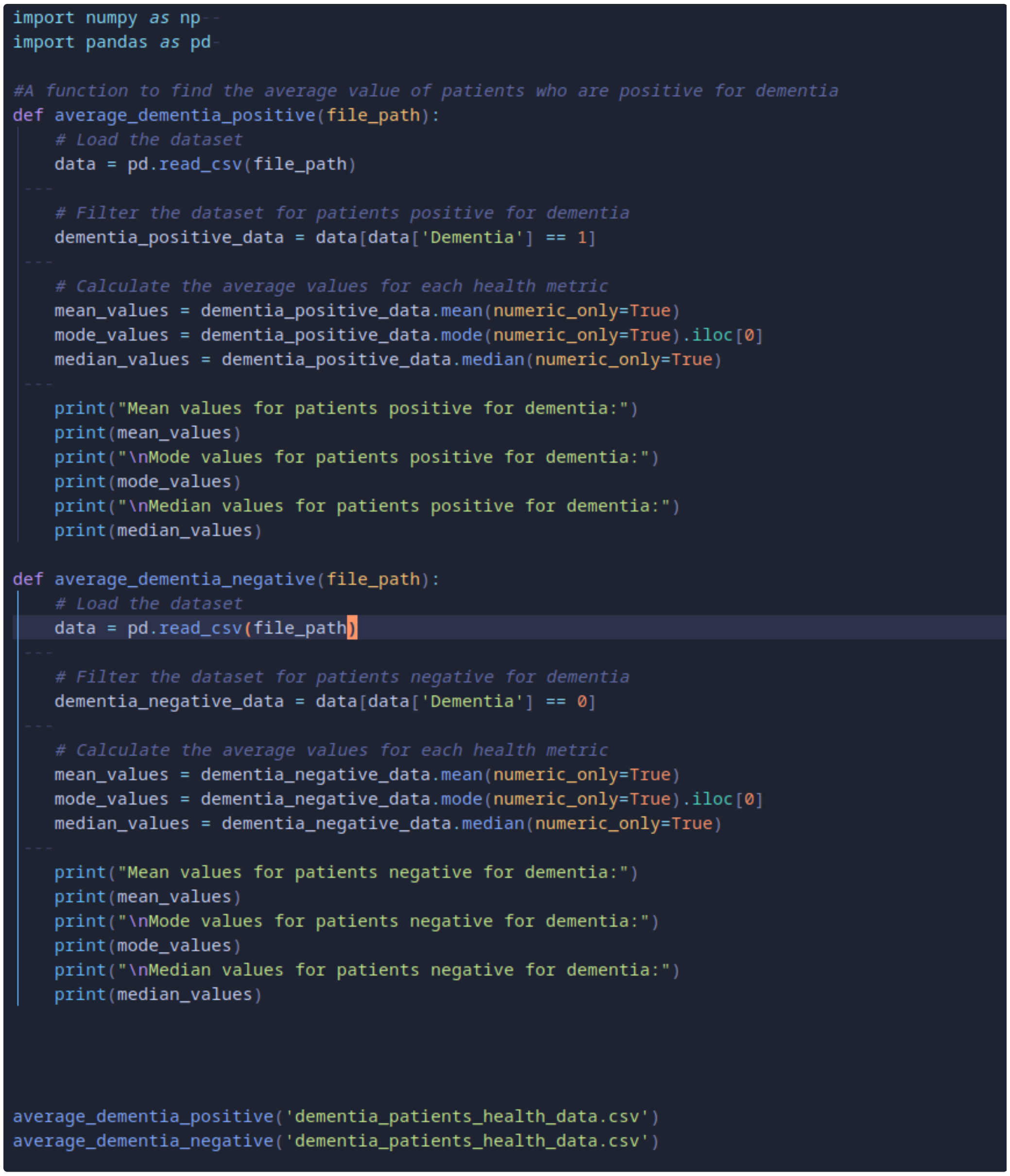
```
#A script to extract unique values from all columns in a CSV file and count their occurrences.
import pandas as pd
import sys
import os
import argparse
import numpy as numpy

def uniqueValCount(input_file, output_file):
    # Read the CSV file into a DataFrame
    df = pd.read_csv(input_file)

    # Create a dictionary to hold the results
    result = {}

    # Iterate through each column in the DataFrame
    for column in df.columns:
        # Get unique values and their counts
        # If values are numbers give a range of values in increments of 10 unless the numbers are 1

        if pd.api.types.is_numeric_dtype(df[column]):
            if set(df[column].dropna().unique()).issubset({0, 1}):
                counts = df[column].value_counts().to_dict()
                result[column] = { 'True': counts.get(1, 0), 'False': counts.get(0, 0) }
            else:
                bins = list(range(int(df[column].min()), int(df[column].max()) + 10, 10))
                labels = [f"{b}-{b+9}" for b in bins[:-1]]
                df['binned'] = pd.cut(df[column], bins=bins, labels=labels, right=False)
                counts = df['binned'].value_counts().sort_index().to_dict()
                result[column] = counts
                df.drop(columns=['binned'], inplace=True)
        else:
            counts = df[column].value_counts().to_dict()
            result[column] = counts
    # Convert the result dictionary to a DataFrame for better formatting
    result_df = pd.DataFrame(dict([(k,pd.Series(v)) for k,v in result.items()]))

    # Save the result to a .md file
    result_df.to_csv(output_file, index_label='Value')
    print(f"Unique value counts saved to {output_file}")

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Extract unique values from all columns in a CSV f
    parser.add_argument('input_file', type=str, help='Path to the input CSV file')
    parser.add_argument('output_file', type=str, help='Path to the output CSV file')

    args = parser.parse_args()

    uniqueValCount(args.input_file, args.output_file)
```

Output:



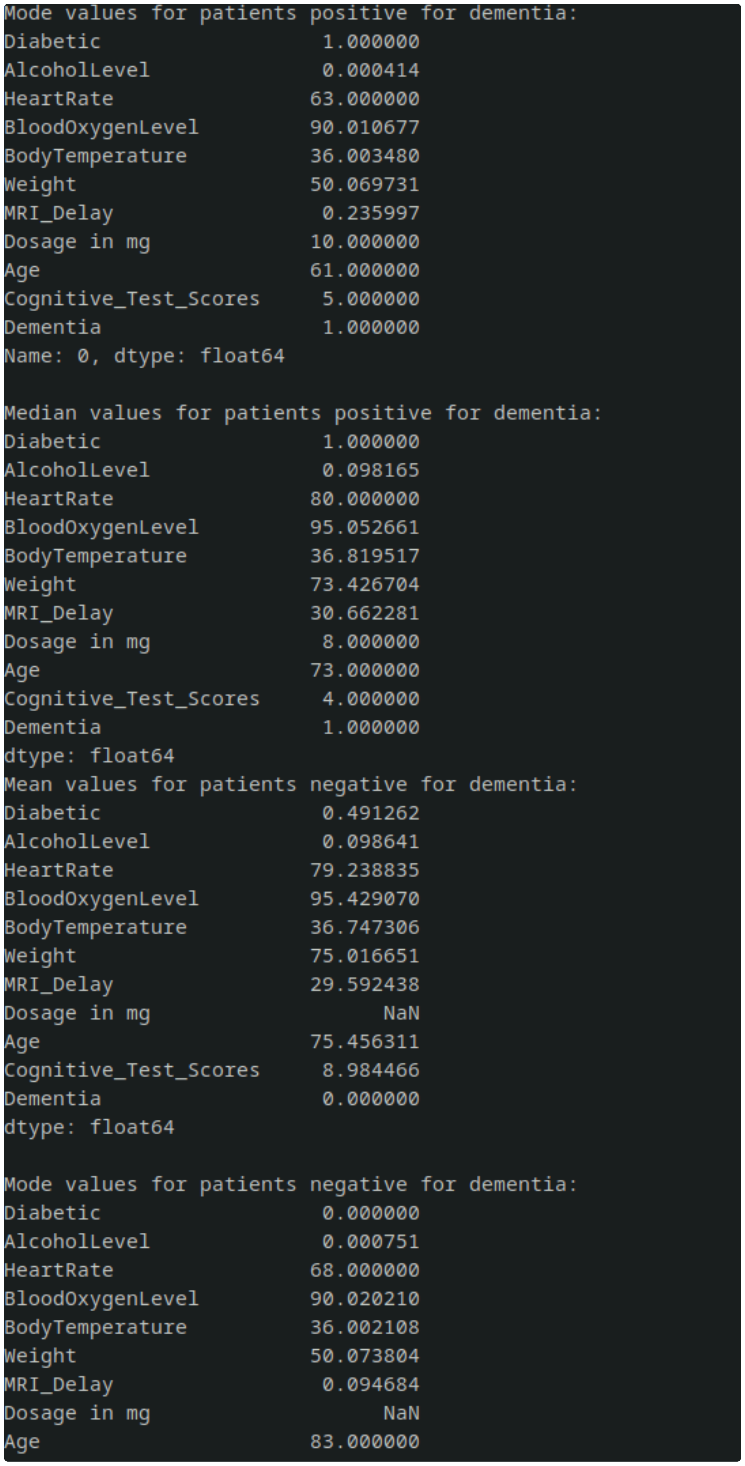
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Value	Diabetic	AlcoholLevel	HeartRate			Weight	MRI_Delay	Prescription	Dosage in mg	Age	Education_Level	Dominant_Hand	Gender	Family_History	Smoking_Status	APOE_ε4
2	0-9							149									
3	1-10									355							
4	10-19							169									
5	11-20									83							
6	20-29							186									
7	21-30									47							
8	30-39							176									
9	36-45																
10	40-49							167									
11	50-59						222	153									
12	60-69			277			192				332						
13	70-79			232			192				307						
14	80-89			221			213				332						
15	90-99			250			181										
16	Balanced Diet																
17	Low-Carb Diet																
18	Mediterranean Diet																
19	Male													496			
20	Female													504			
21	Memantine								128								
22	Rivastigmine								119								
23	Donepezil								113								
24	Galantamine								125								
25	True	513															
26	False	487															
27	Never Smoked															452	
28	Former Smoker															458	
29	Current Smoker															90	
30	Sedentary																
31	Mild Activity																
32	Moderate Activity																
33	Yes														520		
34	No														480		
35	Positive																694
36	Negative																306
37	Poor																
38	Good																
39	Left												519				
40	Right												481				
41	No School											155					
42	Primary School											389					
43	Secondary School											304					
44	Diploma/Degree											152					
45	None																
46	Diabetes																
47	Heart Disease																
48	Hypertension																
49																	
50																	
51																	
52																	
53																	
54																	
55																	
56																	

Finally i wrote a script to find the mean, mode and median values for dementia positive and negative patients.



```
import numpy as np
import pandas as pd

#A function to find the average value of patients who are positive for dementia
def average_dementia_positive(file_path):
    # Load the dataset
    data = pd.read_csv(file_path)

    # Filter the dataset for patients positive for dementia
    dementia_positive_data = data[data['Dementia'] == 1]

    # Calculate the average values for each health metric
    mean_values = dementia_positive_data.mean(numeric_only=True)
    mode_values = dementia_positive_data.mode(numeric_only=True).iloc[0]
    median_values = dementia_positive_data.median(numeric_only=True)

    print("Mean values for patients positive for dementia:")
    print(mean_values)
    print("\nMode values for patients positive for dementia:")
    print(mode_values)
    print("\nMedian values for patients positive for dementia:")
    print(median_values)

def average_dementia_negative(file_path):
    # Load the dataset
    data = pd.read_csv(file_path)

    # Filter the dataset for patients negative for dementia
    dementia_negative_data = data[data['Dementia'] == 0]

    # Calculate the average values for each health metric
    mean_values = dementia_negative_data.mean(numeric_only=True)
    mode_values = dementia_negative_data.mode(numeric_only=True).iloc[0]
    median_values = dementia_negative_data.median(numeric_only=True)

    print("Mean values for patients negative for dementia:")
    print(mean_values)
    print("\nMode values for patients negative for dementia:")
    print(mode_values)
    print("\nMedian values for patients negative for dementia:")
    print(median_values)



average_dementia_positive('dementia_patients_health_data.csv')
average_dementia_negative('dementia_patients_health_data.csv')
```

Output:



```
Mode values for patients positive for dementia:
Diabetic                    1.000000
AlcoholLevel                0.000414
HeartRate                  63.000000
BloodOxygenLevel           90.010677
BodyTemperature            36.003480
Weight                     50.069731
MRI_Delay                   0.235997
Dosage in mg               10.000000
Age                        61.000000
Cognitive_Test_Scores       5.000000
Dementia                    1.000000
Name: 0, dtype: float64

Median values for patients positive for dementia:
Diabetic                    1.000000
AlcoholLevel                0.098165
HeartRate                  80.000000
BloodOxygenLevel           95.052661
BodyTemperature            36.819517
Weight                     73.426704
MRI_Delay                  30.662281
Dosage in mg                8.000000
Age                        73.000000
Cognitive_Test_Scores       4.000000
Dementia                    1.000000
dtype: float64
Mean values for patients negative for dementia:
Diabetic                    0.491262
AlcoholLevel                0.098641
HeartRate                  79.238835
BloodOxygenLevel           95.429070
BodyTemperature            36.747306
Weight                     75.016651
MRI_Delay                  29.592438
Dosage in mg                     NaN
Age                        75.456311
Cognitive_Test_Scores       8.984466
Dementia                    0.000000
dtype: float64

Mode values for patients negative for dementia:
Diabetic                    0.000000
AlcoholLevel                0.000751
HeartRate                  68.000000
BloodOxygenLevel           90.020210
BodyTemperature            36.002108
Weight                     50.073804
MRI_Delay                   0.094684
Dosage in mg                     NaN
Age                        83.000000
```

Next sprint i will be focusing on training an LLM with the data.