

Large Scale Data Driven Applications	Interactive Dashboard Documentation	Version 1.0
Sameer Uddin 20004135		

Contents

Purpose	3
Design	3
Implementation	5
Frontend	6
Backend	6
Database	8
Interactions	8
Filtering Data and Interpreting Data	10
Overview Page	10
Patients Page	11
Appointments Page	11
Neighbourhoods Page	12
Testing	13
Overview Page	13
Patients Page	14
Appointments Page	17
Testing Screenshots	18
Overview Page	18
Patients Page	20
Appointments Page	26
Figure 1 - Homepage Design	3
Figure 2 - Patients Page	4
Figure 3 - Appointments Page	4
Figure 4 - Neighbourhood Page	5
Figure 5 - Database Schema	6
Figure 6 - MongoDB Library	8
Figure 7 - Patients Dashboard	8
Figure 8 - Overview Page	10
Figure 9 - Patients Page	11

Figure 10 - Appointments over Months	12
Figure 11 - Appointments Page	12
Figure 12 - Testing Total Patients	18
Figure 13 - Testing Total Appointments	18
Figure 14 - Testing Neighbourhood Count	19
Figure 15 - Testing Diabetes Count.....	19
Figure 16 - Testing Appointments over Months	20
Figure 17 - Testing Diabetes Filter	20
Figure 18 - Testing Gender Filter.....	21
Figure 19 - Testing Age Filter (Patients Aged 1).....	21
Figure 20 - Testing Age Filter (Patients Aged 1 & 2)	22
Figure 21 - Testing Age Filter (Aged 200).....	22
Figure 22 - Testing Multiple Conditions.....	23
Figure 23 - Testing Multiple Conditions of Different Categories.....	23
Figure 24 - Testing PatientID Search	24
Figure 25 - Testing Neighbourhood Search.....	24
Figure 26 - Testing Reset Filter	25
Figure 27 - Testing String Search	25
Figure 28 - Testing Appointments Date	26
Figure 29 - Testing Appointments for April.....	26

Purpose

The purpose of this coursework is to implement the learnings from the first part of this coursework into an interactive dashboard that can be used by the clinic to gain valuable insight into their patients, appointment history and how this may affect no-shows for appointments. The dashboard should help the clinic gain insight into various factors that can affect no-shows by factors such as patient demographic, medical conditions, reminder effectiveness and appointment history.

The interactive dashboard will visualise data by using graphs (D3.js) and give the clinic filtering options to further explore what contributes to no-show rates. The graphs and information provided should also help the clinic identify what is causing no-show rates and should give them ideas on what can be done to reduce these.

The ultimate objective of this dashboard is to help reduce no-shows which should improve the overall quality of service that can be provided by the clinic. This will be achieved by increasing the patients turn out rate and reducing the amount of time that is wasted. This dashboard will provide the clinic and its staff with key information to increase their efficiency.

Design

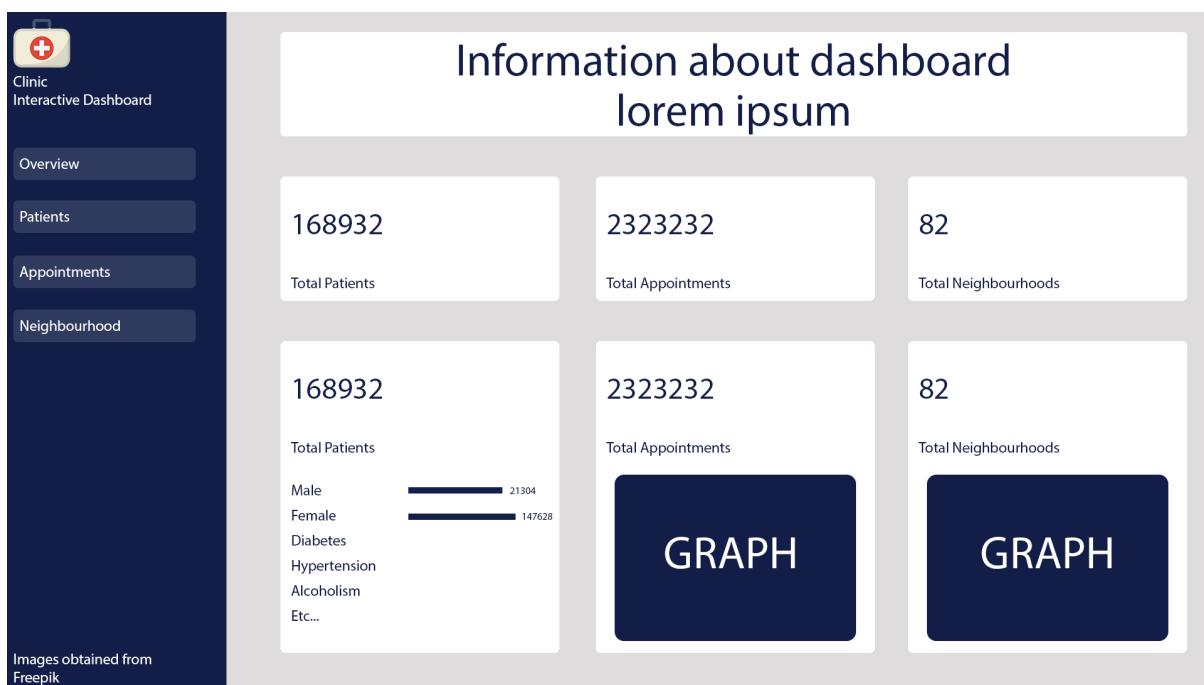


Figure 1 - Homepage Design

The landing page or the home page will act as an overview of the database. It will provide insight into the data. This page will offer quick access to important metrics and include graphs that users can view to get a quick overview of the database.

Figure 2 - Patients Page

This page will provide information on the patients in the database. This page will not provide information on appointment history. This page will allow the clinic to search for patients based on their ID (as name is not available). They can also filter the table by other categories such as medical conditions, age, gender, scholarship and neighbourhood.

Figure 3 - Appointments Page

This page will provide information on appointments. The table will have key information on appointments such as the patients ID, appointment ID and other information like key dates. Based on the design the user should be able to filter by date range, show status, neighbourhood, reminders and the date difference between booking date and scheduled date.

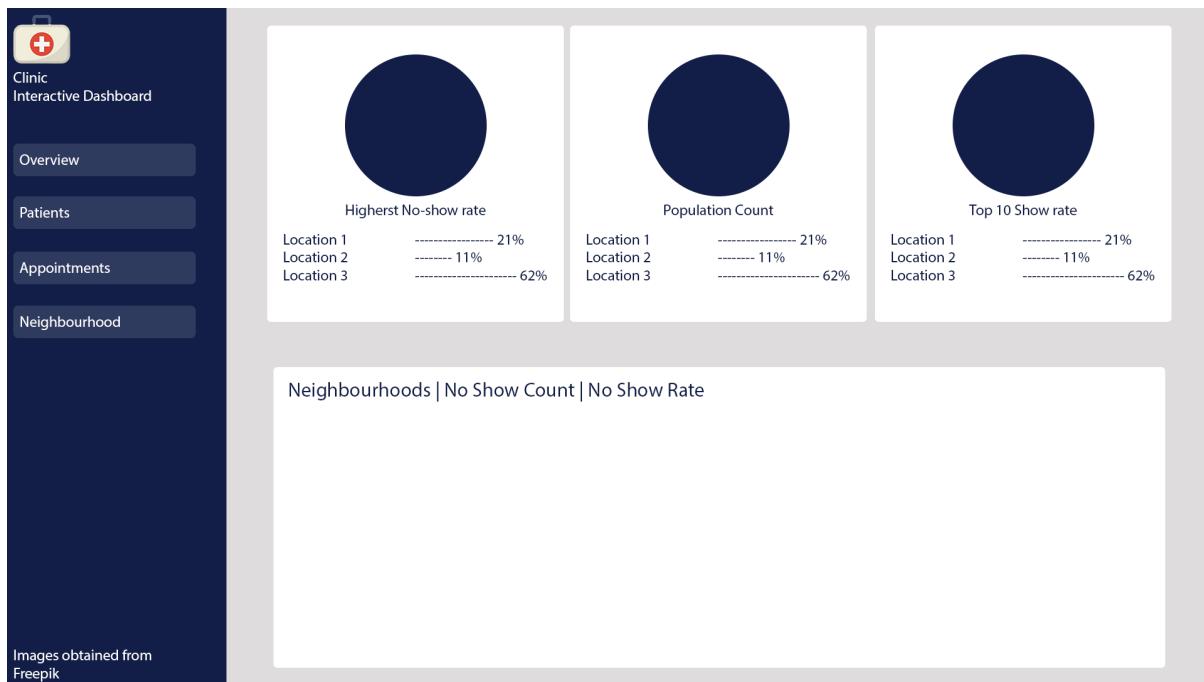


Figure 4 - Neighbourhood Page

This page will provide information on neighbourhoods. This page should have a table with all the lists of neighbourhoods and provide key information based on these neighbourhoods. This page will display metrics such as the neighbourhoods with the highest no-show rates, population of each neighbourhood and the neighbourhoods that have the lowest no-show rates.

Implementation

The implementation of the dashboard is done through the use of frontend languages such as:

- HTML
- CSS
- Bootstrap
- JavaScript

Database connections and filtering data is handled by PHP. PHP is used in this project to connect to the database and retrieve the information from the database which can then be used in the frontend of the application.

The database used in for this project is MongoDB. This database stores the information for both the patients and appointments. Each page utilises each table to fetch and populate tables with information. The overview page uses both databases to summarise key information that were queried in the first part of the coursework.

Appointments	
⌚_id	objectid
⌚ AppointmentDay	isodate
⌚ Patient	object
⌚ SMS_received	boolean
⌚ Showed_up	boolean
⌚ AppointmentID	int32
⌚ Patient.Age	int32
⌚ ScheduledDay	isodate
⌚ DateDiff	int32
⌚ Patient.Alcoholism	boolean
⌚ Patient.Diabetes	boolean
⌚ Patient.Gender	string
⌚ Patient.Handcap	boolean
⌚ Patient.Hipertension	boolean
⌚ Patient.Neighbourhood	string
⌚ Patient.PatientId	int64
⌚ Patient.Scholarship	boolean

Patients	
⌚_id	objectid
⌚ Diabetes	boolean
⌚ PatientId	int64
⌚ Scholarship	boolean
⌚ Age	int32
⌚ Gender	string
⌚ Alcoholism	boolean
⌚ Handcap	boolean
⌚ Hipertension	boolean
⌚ Neighbourhood	string

Figure 5 - Database Schema

Frontend

HTML and some basic CSS is used to structure the pages. The basic CSS used is used to align some items and change colours of components.

Bootstrap was utilised in this project as the grid system is extremely useful and a lot of components will be added to this page. The overview contains a lot of graphs and the grid system ensures that graphs do not overlap each other. The patients and appointments pages have a row for filters and a row for the table that will be generated. The grid system also made it easier to implement the navigation bar to the left side and have the right side of the page entirely dedicated to the contents of the given page.

JavaScript was another important part of this project as it allows for interactivity. It also handles user activities such as resetting the filters, applying the data, populating the data in the table by using PHP scripts and generating the dynamic graphs for the overview page. JavaScript also enables us to use pagination which will reduce the load on the database. This database is large (appointments contains over 100,000 records) and pagination allows the load to be spread over pages.

Backend

The backend is handled by using PHP. Some essential files include the “db_connection.php” file as this file is used to connect to the remote database.

```

<?php
require 'vendor/autoload.php'; // include Composer's autoloader

use MongoDB\Client; // Import the MongoDB Client class

class Database // Create a Database class to handle database
connections. Used in most other files
{
    public $database; // Create a public variable to store the database
connection

    public function __construct() // Create a constructor method to
connect to the database
    {
        $uri =
"mongodb+srv://su042002:r0m2ZGaYscBSPW9r@cluster0.wkif8.mongodb.net/";
// MongoDB connection URI
        $client = new Client($uri); // Create a new MongoDB client
        $this->database = $client->Courswork1; // Select the database
to use
    } etc...

```

This file also contains some methods used to filter data from the database whether it is simply counting documents returned or using the aggregation pipeline function of MongoDB.

```

public function getTotalNeighbourhoods() // Method to get the total
number of neighbourhoods in the database
{
    $patients = $this->database->Patients;

    $result = $patients->aggregate([
        // Use aggregation to group by
        Neighbourhood and count the number of groups
        [
            '$group' => [
                '_id' => '$Neighbourhood' // Group by Neighbourhood
field
            ]
        ],
        [
            '$count' => 'TotalNeighbourhoods' // Count the number of
groups
        ]
    ]);
}

```

Other key files include “filter_patients.php” and “filter_appointments.php” as these files are used by the front end to fetch filtered data that is return as JSON data. They also check if certain filters are empty or if they contain values to enable the user use a mixture of filters to fetch information.

Database

The database used is MongoDB. This database uses embedded design as designed in part 1 of this coursework. The two tables' patients and appointments contain all the data needed for the dashboard. The connection is created using via Composer to allow to enable the MongoDB PHP library.

This screenshot shows the PHPinfo page from a browser, specifically highlighting the MongoDB, MySQL, and MySQLnd sections. The MongoDB section displays various configuration directives and their values. The MySQL and MySQLnd sections show the status of MySQL extensions and their respective configurations.

Directive	Local Value	Master Value
mbstring.http_outoutconv_mimetypes	*text/*application/xml+*xml	*text/*application/xml+*xml
mbstring.internal_encoding	no value	no value
mbstring.language	neutral	neutral
mbstring.regex_retry_limit	1000000	1000000
mbstring.regex_stack_limit	100000	100000
mbstring.strict_detection	Off	Off
mbstring.substitute_character	no value	no value

MongoDB support	enabled
MongoDB extension version	1.28.1
MongoDB extension stability	stable
libbson bundled version	1.28.1
libmongoc bundled version	1.28.1
libmongoc SSL	enabled
libmongoc SSL library	OpenSSL
libmongoc crypto	enabled
libmongoc crypto library	lbcrypto
libmongoc crypto system profile	disabled
libmongoc SASL	disabled
libmongoc SRV	enabled
libmongoc compression	disabled
libmongocrypt bundled version	1.11.0
libmongocrypt crypto	enabled
libmongocrypt crypto library	lbcrypto
crypt_shared library version	unknown

Directive	Local Value	Master Value
mongodb.debug	no value	no value

mysqld	mysqld 8.4.1
Version	mysqld 8.4.1
Compression	supported
core SSL	supported
extended SSL	supported
Command buffer size	4096
Read buffer size	32768
Read timeout	88400
Collecting statistics	Yes

Figure 6 - MongoDB Library

Interactions

The user interacts with the front end of the application by using the filters:

This screenshot shows the Patients dashboard of the Clinic Interactive Dashboard. It features a sidebar with navigation links for Overview, Patients (which is selected), and Appointments. The main area has three boxes: Total Patients (60270), Total Appointments (106987), and Total Neighbourhoods (81). A sidebar note states: "Images used on the website were obtained from FreePic". Below these are two filter panels: one for "Filters" and another for "Patient ID". The "Filters" panel includes fields for Medical Condition (checkboxes for Diabetes, Hypertension, Alcoholism, Handicap), Age (Min: 1, Max: 1000), Gender (radio buttons for Male, Female), Scholarship (radio buttons for Scholarship (True) and Scholarship (False)), and Neighbourhood (dropdown menu set to All). Buttons for "Reset Filters" and "Apply Filters" are present. The "Patient ID" panel lists 60270 entries, each with columns for PatientID, Age, Gender, Neighbourhood, Alcoholism, Handicap, Hypertension, Diabetes, and Scholarship. The first few rows show details for patients with IDs 29872499824296, 7542951368435, 78124564369297, etc. At the bottom, a pagination bar shows pages 1 through 60270.

PatientID	Age	Gender	Neighbourhood	Alcoholism	Handicap	Hypertension	Diabetes	Scholarship
29872499824296	62	F	JARDIM DA PENHA	No	No	Yes	No	No
7542951368435	29	M	NOVA PALESTINA	No	No	No	No	No
78124564369297	19	F	CONQUISTA	No	No	No	No	No
8734857996885	65	F	TABUAZEIRO	No	No	No	No	No
311284853849	12	M	NOVA PALESTINA	No	No	No	No	Yes
72984587621439	63	F	SÃO CRISTÓVÃO	No	No	Yes	Yes	No
274164858852	78	F	SÃO CRISTÓVÃO	No	No	Yes	Yes	No
18271722734941	19	F	GRANDE VITÓRIA	No	No	No	No	No
41799315536436	2	M	NOVA PALESTINA	No	No	No	No	No
9291167893717	8	M	NOVA PALESTINA	No	No	No	No	Yes

Figure 7 - Patients Dashboard

When the webpage originally loads the JavaScript files loads all the data (limit 10) with the help of pagination. This prevents all the data from being loaded at once as this would use too much ram and slow down the application. When the user applies the filters JavaScript sends a fetch request to the corresponding PHP filter file. This fetch request is processed against the database and the data is returned as JSON data that can be used by the JavaScript files. The data is then displayed on the frontend by updating the HTML tables by utilising their ID.

When the user applies a filter the JavaScript file sends a fetch request:

```
fetch('filter_patients.php', { // Fetch the filter_patients.php file
    method: 'POST', // Use the POST method
    headers: { 'Content-Type': 'application/json' }, // Set the Content-Type header
    body: JSON.stringify(filters) // Set the body to the filters object as a JSON string
})
    .then(response => response.json()) // Parse the response as JSON
    .then(data => { // Handle the parsed data
        console.log('Filtered data:', data); // Log the data to the console
        displayPatients(data.data); // Call the displayPatients function with the data
        updatePaginationControls(data.totalCount, data.currentPage, data.limit); // Call the updatePaginationControls function with the total count, current page, and limit
        document.getElementById('resultCount').textContent = `Results: ${data.totalCount}`; // Set the result count text content
    })
    .catch(error => { // Handle any errors
        console.error('Error:', error); // Log the error to the console
        alert('An error occurred while applying filters.'); // Show an alert to the user
    });
}
```

The PHP file queries the database:

```
// Age Range Filter
if (!empty($filters['AgeMin']) && !empty($filters['AgeMax'])) { // Age Range is not empty
    $query['Age'] = ['$gte' => intval($filters['AgeMin']), '$lte' => intval($filters['AgeMax'])]; // Add the Age Range to the query
}
```

The data is then sending back the data based on the filters applied and returns it in JSON format:

```

// Execute the query
$result = $patients->find($query, ['skip' => $skip, 'limit' =>
$limit])->toArray(); // Get the records based on the query and limit
the records per page
$totalCount = $patients->countDocuments($query); // Get the total count
of records based on the query

echo json_encode([
    // Return the result as JSON
    'data' => $result, // Records
    'totalCount' => $totalCount, // Total count of records
    'currentPage' => $page, // Current page
    'limit' => $limit // Limit of records per page
]);

```

The dashboard has been designed to allow the users to gain insight into patient and appointment insights by applying filters to the data. The overview page provided quick access along with graphs of common enquiries.

Filtering Data and Interpreting Data

Overview Page

The main page of the application is where all the charts and visualisations are stored. This page was created as a quick way to interpret interesting data. The charts are dynamically updated when the database is refreshed. A lot of these charts were created from questions that were asked in part 1 of the coursework. This page can be furthered developed to include more charts that will allow users to quickly answer any frequently asked questions or to query interesting information.

This page includes total patients' information, total appointments, total neighbourhoods and other information such as no-show rates.

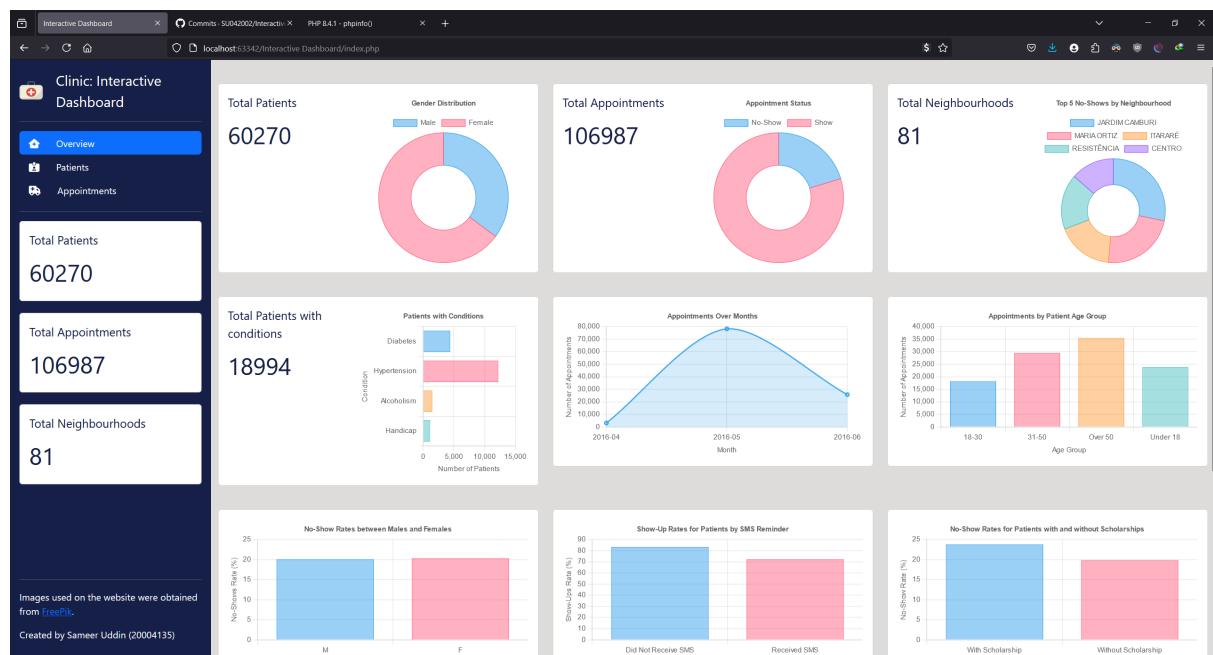


Figure 8 - Overview Page

Patients Page

The patients page includes many filters to allow the user to gain insight into patient information. The result badge in the top right of the filter panel shows the total amount of patients in the database. The user can individually search for users in the database using their Patient ID. They can also filter by using the other criteria provided such as medical condition and age. One example of a query would be searching for users that are under 18 with diabetes in the neighbourhood “Bela Vista”:

The screenshot shows the 'Interactive Dashboard' application. On the left, there's a sidebar with 'Overview', 'Patients' (which is selected), and 'Appointments'. Below these are summary boxes for 'Total Patients' (60270), 'Total Appointments' (106987), and 'Total Neighbourhoods' (81). The main content area is titled 'Filters' and includes fields for 'Enter Patient ID' (with a result count of 2), 'Medical Condition' (with 'Diabetes' checked), 'Age' (Min: 1, Max: 18), 'Gender' (Male and Female options), 'Scholarship' (with 'Scholarship (True)' checked), and 'Neighbourhood' (set to 'BELA VISTA'). There are 'Reset Filters' and 'Apply Filters' buttons. Below the filters is a table with columns: PatientID, Age, Gender, Neighbourhood, Alcoholism, Handicap, Hypertension, Diabetes, and Scholarship. Two rows of data are shown:

PatientID	Age	Gender	Neighbourhood	Alcoholism	Handicap	Hypertension	Diabetes	Scholarship
7223647931958	16	F	BELA VISTA	No	No	Yes	Yes	Yes
98191156369538	15	F	BELA VISTA	No	No	No	Yes	No

Figure 9 - Patients Page

The table is populated when the user applies the filter. In this case the query returned 2 patients who fall under this category.

Appointments Page

The same logic from the patient’s page is applied to the appointments page. The user can apply queries to the appointments page to gain insight into the data. The database on load contains 106987 rows of data. The user can filter by appointment date, showed up, SMS received, date difference, patient ID and appointment ID. This table does not contain information such as patient neighbourhood as this data can be queried in the patient’s page. The default range for appointment date is currently set for 2016-04 to 2017-07 to encompass all appointments (information obtained from overview page):



Figure 10 - Appointments over Months

PatientID	Appointment ID	Scheduled Day	Appointment Day	Date Difference	SMS Received	Showed-Up
558997776694438	5642503	2016-04-29	2016-04-29	0	No	Yes
867951213174	5642828	2016-04-29	2016-04-29	0	No	Yes
8841196448183	5642494	2016-04-29	2016-04-29	0	No	Yes
95985133231274	5626772	2016-04-27	2016-04-29	2	No	Yes
78124564369297	5629123	2016-04-27	2016-04-29	2	No	Yes
863229818887631	5616091	2016-04-25	2016-04-29	4	Yes	No
8734857996685	5641780	2016-04-29	2016-04-29	0	No	Yes
342815551642	5628068	2016-04-27	2016-04-29	2	No	Yes
311284853849	5628907	2016-04-27	2016-04-29	2	No	No
72984587621439	5637975	2016-04-29	2016-04-29	0	No	Yes

Figure 11 - Appointments Page

Neighbourhoods Page

The original design had another page for neighbourhood but I got rid of this page as it was repetitive. Neighbourhood information can be interpreted from the patient's page. The overview page also contains graphs and some data about neighbourhoods such as how many neighbourhoods there are. The patient's database also has distinct rows of patients which means information such as how many patients are from a neighbourhood can be found through the patients' page.

Testing

Overview Page

Test Case	Test Scenario	Expected Results	Actual Results	Screenshot (hyperlink)
Total patients count displaying correctly on reload of page	<p>Reload the page and see if the total count is correct compared to the original Excel sheet.</p> <p>This will be done by selecting all the rows and checking the count in Excel.</p>	The total count should be 60270.	The total count is 60270.	Figure 12 - Testing Total Patients
Total appointments count displaying correctly on reload of page	<p>Reload the page and see if the total count is correct compared to the original Excel sheet used to build the database.</p> <p>This will be done by selecting all the rows and checking the count in Excel.</p>	The total count should be 106987.	The total count is 106987.	Figure 13 - Testing Total Appointments
Total neighbourhood count displaying correctly on reload of page	<p>Reload the page and see if the total count is correct compared to the original Excel sheet.</p> <p>Duplicate records will be removed by using Excel's remove duplicate function.</p>	There should be 81 distinct neighbourhoods.	<p>There are 81 neighbourhoods.</p> <p>The Excel sheet says 82 unique values because it includes the string "Neighbourhood".</p>	Figure 14 - Testing Neighbourhood Count

Is total patients with Conditions graph generated correctly	Reload the page and see what values the graph produces.	Diabetes = 4416 Hypertension = 12, 242 Alcoholism = 1506 Handicap = 1132	The expected results are displayed in the application.	Figure 15 - Testing Diabetes Count
Is total appointments over months generated correctly	Reload the page and see what values the graph produces. Filter the months using Excel and count all the selected values.	April = 3104 May = 78202 June = 25681	The expected results are displayed correctly in the line graph.	Figure 16 - Testing Appointments over Months

The values being displayed are correct and the graphs generated are accurate when compared with the original data. The values and charts are generated using similar methods so they are also accurate and are displaying results as intended.

Patients Page

Test Case	Test Scenario	Expected Results	Actual Results	Screenshot (hyperlink)
Patients filter for diabetes	Apply diabetes filter to the data and see if first page and last page are users with diabetes. Another confirmation can be obtained from the patients table testing, according to the database there are 4416 patients with diabetes. 4416 results should be obtained from the database.	Only patients with diabetes should be displayed. 4416 results should be obtained after only applying the diabetes filter (all neighbourhoods selected).	First and last page only contain patients with diabetes. 4416 results were also obtained.	Figure 17 - Testing Diabetes Filter

Gender filter for patients	The overview page was confirmed to be working and that page has a chart for gender distribution. There are 21189 males and 39081 females.	I will select females and there should be 39081 results obtained in total.	The search query returned 39081 results.	Figure 18 - Testing Gender Filter
Age filter for patients that are 1	I will set the age between 1 and 1 which should only return results for patients that are 1.	All the results should return users that are 1 and nothing else.	The query returned patients that are only 1.	Figure 19 - Testing Age Filter (Patients Aged 1)
Age filter for patients that are 1 and 2	I will set the age between 1 and 2 which should return more results than the original query above.	All the results should return users are 1 and 2 and nothing else.	The query returned patients that are only 1 and 2.	Figure 20 - Testing Age Filter (Patients Aged 1 & 2)
Age filter for patients that should not exist	I will set the age value to something ridiculous like 200 to see if any results are returned.	There should be no results and the table should be empty.	The query returned no patients.	Figure 21 - Testing Age Filter (Aged 200)
Selecting multiple medical conditions	I will set two conditions so that patients with multiple conditions are only returned.	The table should only be filled with patients with diabetes and hypertension. Both conditions should be true to return results.	Only patients with hypertension and diabetes were returned.	Figure 22 - Testing Multiple Conditions
Selecting multiple filters of different categories	I will set a medical condition and an age filter to find patients aged 1 to 12 with diabetes.	The table should only be filled with users aged 1-12 with diabetes.	11 results were returned and all of them met both requirements.	Figure 23 - Testing Multiple Conditions of Different Categories

Searching for patient using PatientID	I will search for a specific patient using their PatientID.	The table should only have 1 patient because there are no duplicate patient entries in the database.	1 result was returned with the correct PatientID.	Figure 24 - Testing PatientID Search
Searching for patient by neighbourhood	I will search for patients in only 1 neighbourhood. For this test I will only select "Aeroporto".	The table should return 7 values as that is how many there are for that location in the Excel sheet.	7 results were returned.	Figure 25 - Testing Neighbourhood Search
Pagination	Pressing the button should switch pages.	The table should be updated with new results.	The table changes values when the next button is clicked.	
Reset Filters Button	Set all the values to something and then press reset to see if it is updated.	All the values should be reset to their default values such as empty string or something else like 1 for age.	All the values were reset correctly.	Figure 26 - Testing Reset Filter
Apply Filters Button	Select filters and press apply.	The table should be updated according to the filters applied.	The filter button works correctly.	
Real time updates	Selecting filters and then press apply and then repeat the process with different filters to see if the table is updated. I will select male first and then filter. Then select females.	The results should change once the 1 st filter applied. Then the table should update when the 2 nd filter is applied.	The table updates properly after every update to the filter.	

Invalid Input	Type in none existent patient filter and see if anything is returned. A string of characters will be sent instead of int.	Nothing should be returned. An error prompt could be provided for invalid input.	No data is returned but there is no prompt for an invalid input that is checked before querying the database	Figure 27 - Testing String Search
----------------------	----------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------	-----------------------------------

Appointments Page

Test Case	Test Scenario	Expected Results	Actual Results	Screenshot (hyperlink)
Search for appointments with default appointment day date filter	The default values are set to encompass all the values in the database.	The total amount of appointments returned should be 106987.	106987 results were returned.	Figure 28 - Testing Appointments Date
Search for appointments for duration of 1 month	For the month of April there were 3104 appointments according to the graph in the overview page. The date range will be applied in the appointments page for April.	The database should return 3104 rows of data.	The table has 3104 rows of data.	Figure 29 - Testing Appointments for April

The logic and methods for this page are very similar to the patient's page. That page was tested and works which also means that this page functions as intended. The only problem is that there is no data validation on the frontend before the database is queried.

Testing Screenshots

Overview Page

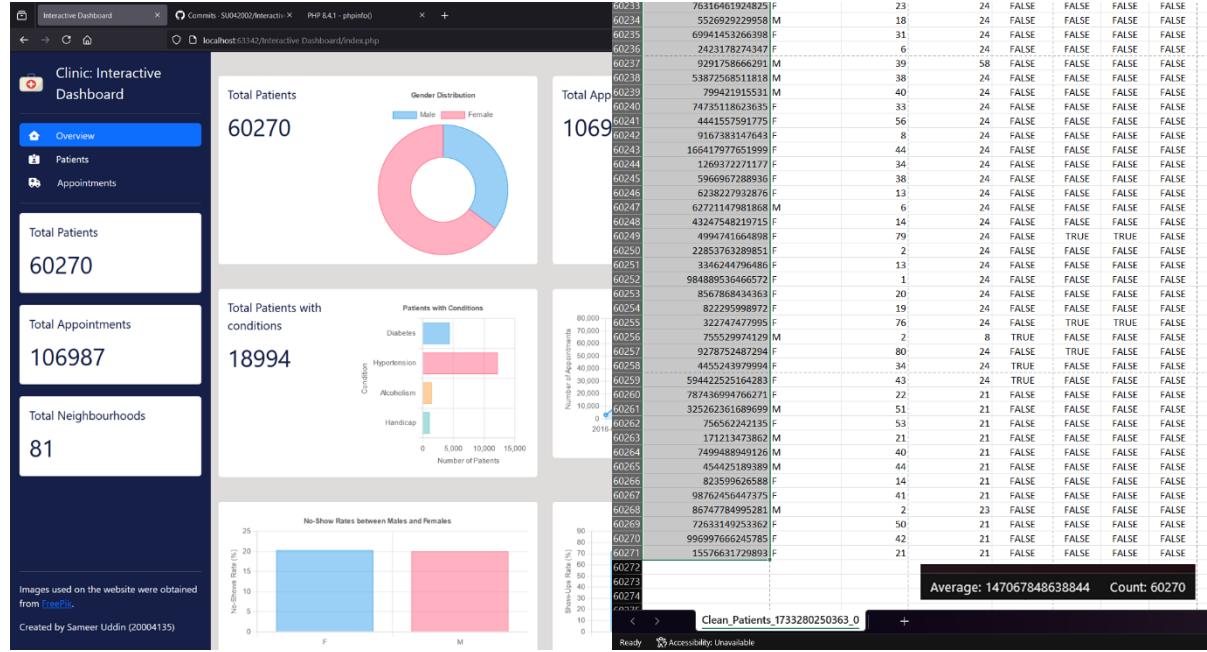


Figure 12 - Testing Total Patients

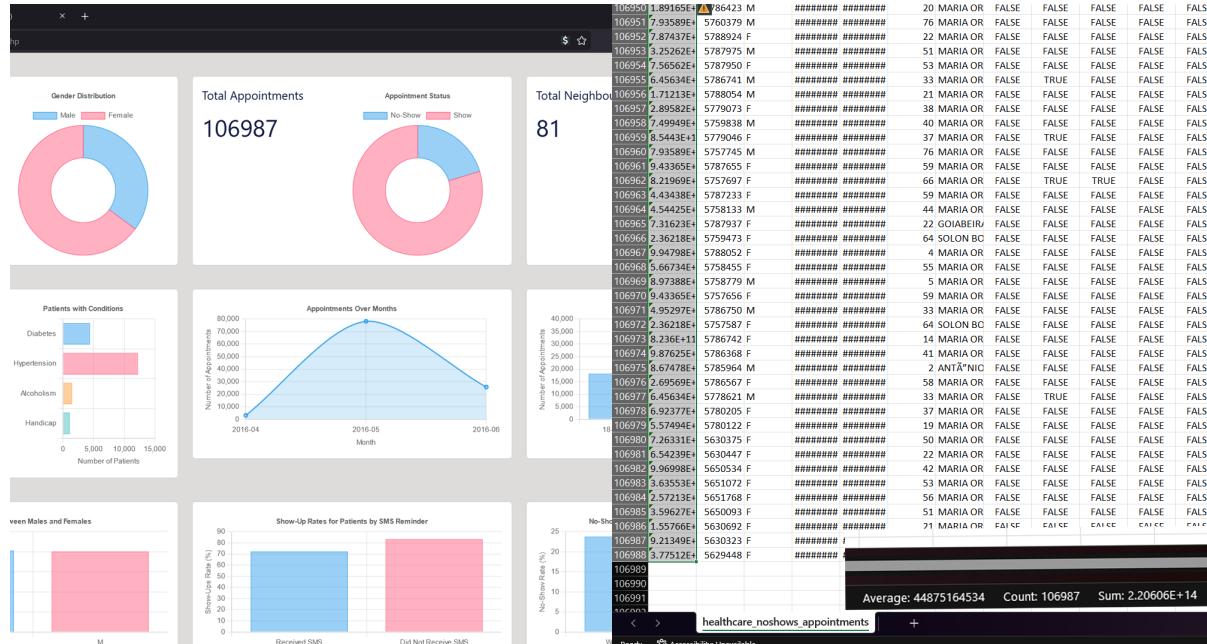


Figure 13 - Testing Total Appointments

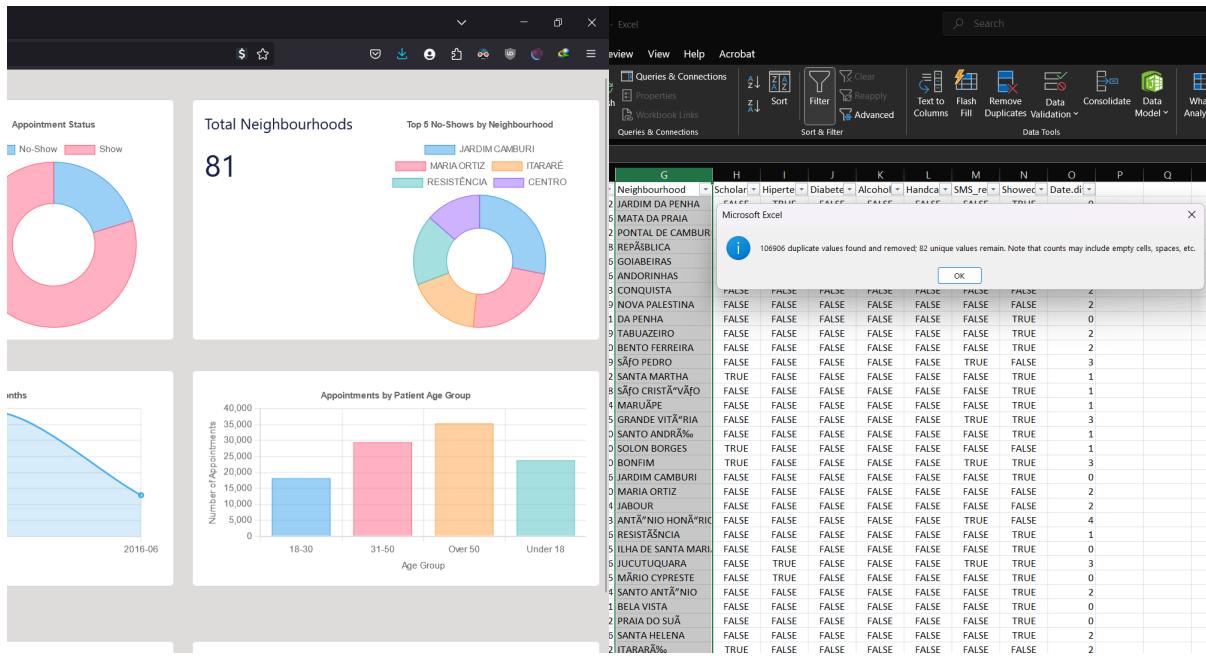


Figure 14 - Testing Neighbourhood Count

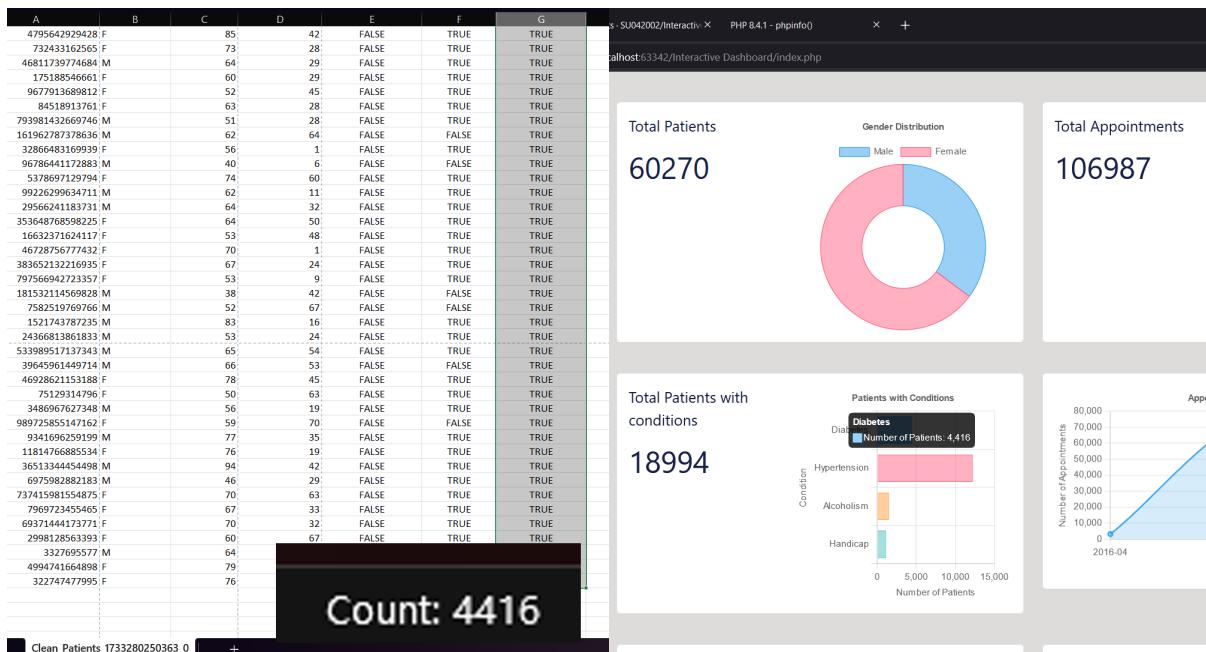


Figure 15 - Testing Diabetes Count

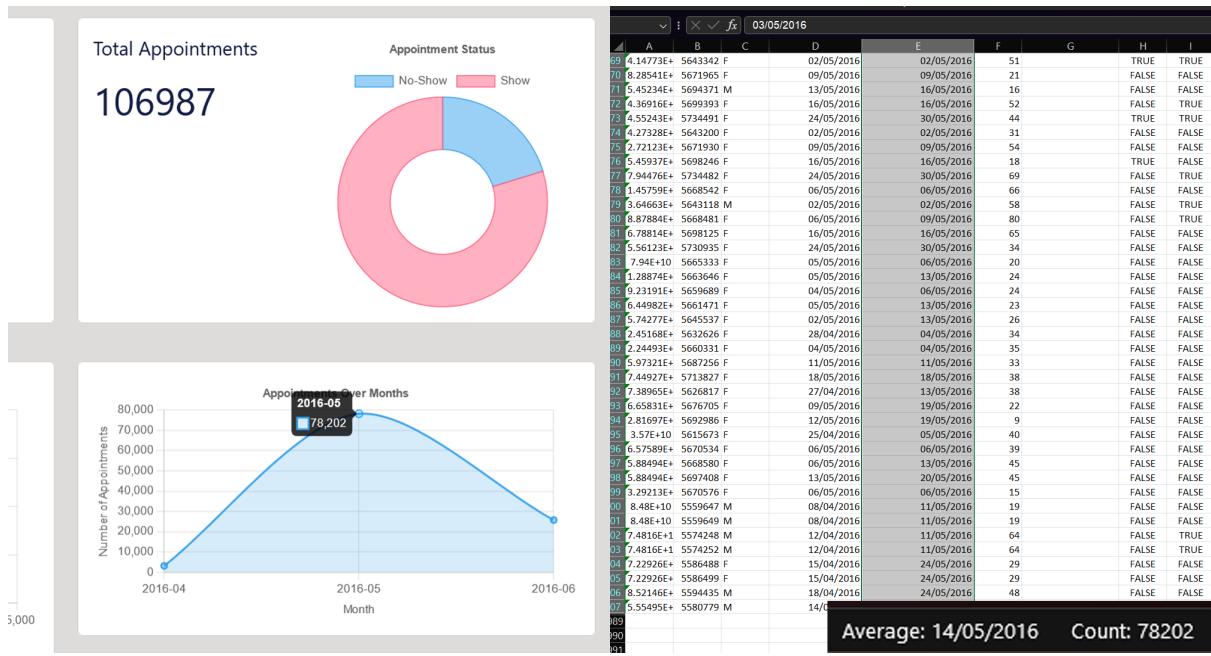


Figure 16 - Testing Appointments over Months

Patients Page

The Patients page includes the following features:

- Overview:** Shows Total Patients (60270) and Total Appointments (106987).
- Filters:** A search bar for "Enter Patient ID" and a "Results: 4416" indicator. Below it are filters for Medical Condition (Diabetes, Hypertension, Alcoholism, Handicap), Age (Min: 1, Max: 1000), Gender (Male, Female), Scholarship (True, False), and Neighbourhood (All). Buttons for "Reset Filters" and "Apply Filters" are present.
- Table:** A grid of patient data with columns: PatientID, Age, Gender, Neighbourhood, Alcoholism, Handicap, Hypertension, Diabetes, and Scholarship. The table lists 4416 patients.
- Text at the bottom:** "Images used on the website were obtained from FreePic." and "Created by Sameer Uddin (20004135)"

Figure 17 - Testing Diabetes Filter

The screenshot shows the Clinic Interactive Dashboard with the 'Patients' tab selected. On the left sidebar, there are three main sections: 'Total Patients' (60270), 'Total Appointments' (106987), and 'Total Neighbourhoods' (81). The main content area features a 'Filters' section with dropdown menus for 'Medical Condition', 'Age', 'Gender', 'Scholarship', and 'Neighbourhood'. Under 'Gender', the 'Female' option is selected. Below the filters is a table of patient data with columns: PatientID, Age, Gender, Neighbourhood, Alcoholism, Handicap, Hypertension, Diabetes, and Scholarship. The table shows 3909 results. At the bottom of the table, there is a navigation bar with page numbers (1, 2, 3, ..., 3909).

Figure 18 - Testing Gender Filter

This screenshot is similar to Figure 18 but with a different filter applied. The 'Age' dropdown is set to '1'. The table below shows 765 results for patients aged 1. The columns are identical to Figure 18: PatientID, Age, Gender, Neighbourhood, Alcoholism, Handicap, Hypertension, Diabetes, and Scholarship. The table lists various patient records, including names like CONQUISTA, JUCUTUQUARA, and SANTOS DUMONT.

Figure 19 - Testing Age Filter (Patients Aged 1)

The screenshot shows the Clinic Interactive Dashboard with the 'Patients' tab selected. On the left sidebar, key statistics are displayed: Total Patients (60270), Total Appointments (106987), and Total Neighbourhoods (81). The main content area features a filter panel with fields for Medical Condition, Age (Min: 1, Max: 2), Gender (Male selected), Scholarship (All), and Neighbourhood (All). Below the filters is a table of patient data with columns: PatientID, Age, Gender, Neighbourhood, Alcoholism, Handicap, Hypertension, Diabetes, and Scholarship. The table lists 1316 results, showing patients aged 1 or 2 across various neighborhoods like REPÚBLICA, ESTRELINEHA, and INHANGUETÁ. A footer note states: "Images used on the website were obtained from FreePik." and "Created by Sameer Uddin (20004135)".

Figure 20 - Testing Age Filter (Patients Aged 1 & 2)

The screenshot shows the same dashboard setup as Figure 20, but with a different filter configuration. In the filter panel, the 'Age' range is set to 200 (Min) and 1 (Max). The 'Gender' field is set to Male. The 'Scholarship' and 'Neighbourhood' dropdowns are set to All. The resulting table below shows 'No patients found'.

Figure 21 - Testing Age Filter (Aged 200)

The screenshot shows a web-based dashboard titled "Clinic: Interactive Dashboard". On the left sidebar, there are three main menu items: "Overview", "Patients" (which is currently selected), and "Appointments". Below the sidebar, there are three summary boxes: "Total Patients" (60270), "Total Appointments" (106987), and "Total Neighbourhoods" (81). A note at the bottom states: "Images used on the website were obtained from FreePik." and "Created by Sameer Uddin (20004135)".

The main content area is titled "Filters" and includes a search bar for "Enter Patient ID" and a results count of "Results: 3583". The filter section allows setting conditions for Medical Condition (Diabetes, Hypertension, Alcoholism, Handicap), Age (Min: 1, Max: 1000), Gender (Male, Female), Scholarship (True, False), and Neighbourhood (All). Buttons for "Reset Filters" and "Apply Filters" are present.

A table below the filters displays patient data with columns: PatientID, Age, Gender, Neighbourhood, Alcoholism, Handicap, Hypertension, Diabetes, and Scholarship. The table contains 3583 rows of data, with the first few rows shown:

PatientID	Age	Gender	Neighbourhood	Alcoholism	Handicap	Hypertension	Diabetes	Scholarship
72984587621439	63	F	SÃO CRISTÓVÃO	No	No	Yes	Yes	No
274164858852	78	F	SÃO CRISTÓVÃO	No	No	Yes	Yes	No
589458459955	39	F	MARUÍPE	No	No	Yes	Yes	No
8841186448183	56	F	JARDIM DA PENHA	No	No	Yes	Yes	No
1578131861739	64	F	TABUAZERIO	No	No	Yes	Yes	Yes

At the bottom of the table, there is a navigation bar with page numbers (1, 2, 3, ..., 359, >).

Figure 22 - Testing Multiple Conditions

This screenshot is nearly identical to Figure 22, showing the same dashboard layout and patient data table. The primary difference is in the filter settings: the "Hypertension" checkbox is now checked, while "Alcoholism" and "Handicap" are unchecked. The results count has changed to "Results: 11".

The table below the filters displays patient data with columns: PatientID, Age, Gender, Neighbourhood, Alcoholism, Handicap, Hypertension, Diabetes, and Scholarship. The table contains 11 rows of data, with the first few rows shown:

PatientID	Age	Gender	Neighbourhood	Alcoholism	Handicap	Hypertension	Diabetes	Scholarship
27422621565119	12	F	JESUS DE NAZARETH	No	No	No	Yes	No
64334195318317	11	F	SÃO JOSÉ	No	No	No	Yes	No
44499636496829	10	F	JOANA D'ARC	No	No	No	Yes	No
154337881614368	1	M	GURIGICA	No	No	No	Yes	No
451646488362969	7	M	SÃO CRISTÓVÃO	No	No	Yes	Yes	No

At the bottom of the table, there is a navigation bar with page numbers (1, 2, >).

Figure 23 - Testing Multiple Conditions of Different Categories

The screenshot shows a web-based dashboard titled "Clinic: Interactive Dashboard". On the left sidebar, there are three main menu items: "Overview", "Patients" (which is currently selected), and "Appointments". Below the sidebar, there are three summary boxes: "Total Patients" (60270), "Total Appointments" (106987), and "Total Neighbourhoods" (81). The main content area is titled "Filters" and displays a search result for PatientID 444996336496829. The search results table includes columns for PatientID, Age, Gender, Neighbourhood, Alcoholism, Handicap, Hypertension, Diabetes, and Scholarship. The result for PatientID 444996336496829 is highlighted in yellow.

Figure 24 - Testing PatientID Search

This screenshot shows the same dashboard interface as Figure 24. The "Patients" menu item is still selected. The main content area now displays a search result for the Neighbourhood "AEROPORTO". The search results table has columns for PatientID, Age, Gender, Neighbourhood, and Alcoholism. The results show 7 entries for the AEROPORTO neighbourhood. A large black box with the text "Count: 7" is overlaid on the right side of the results table.

Figure 25 - Testing Neighbourhood Search

BEFORE

Filters

Medical Condition		Age		Gender		Scholarship	Neighbourhood	Results: 7	
<input checked="" type="checkbox"/> Diabetes	<input type="checkbox"/> Hypertension	<input type="checkbox"/> Alcoholism	<input type="checkbox"/> Handicap	Min 1	Max 1000	<input checked="" type="radio"/> Male	<input type="radio"/> Female	<input checked="" type="checkbox"/> Scholarship (True)	<input type="checkbox"/> Scholarship (False)
						BENTO FERREIRA			
Reset Filters						Apply Filters			

After

Filters

Medical Condition		Age		Gender		Scholarship	Neighbourhood	Results: 7	
<input type="checkbox"/> Diabetes	<input type="checkbox"/> Hypertension	<input type="checkbox"/> Alcoholism	<input type="checkbox"/> Handicap	Min 1	Max 1000	<input type="radio"/> Male	<input type="radio"/> Female	<input type="checkbox"/> Scholarship (True)	<input type="checkbox"/> Scholarship (False)
						All			
Reset Filters						Apply Filters			

Figure 26 - Testing Reset Filter

Filters

Medical Condition		Age		Gender		Scholarship	Neighbourhood	Results: 0	
<input type="checkbox"/> Diabetes	<input type="checkbox"/> Hypertension	<input type="checkbox"/> Alcoholism	<input type="checkbox"/> Handicap	Min 1	Max 1000	<input type="radio"/> Male	<input type="radio"/> Female	<input type="checkbox"/> Scholarship (True)	<input type="checkbox"/> Scholarship (False)
						All			
Reset Filters						Apply Filters			

PatientID	Age	Gender	Neighbourhood	Alcoholism	Handicap	Hypertension	Diabetes	Scholarship
No patients found								

Figure 27 - Testing String Search

Appointments Page

The screenshot shows the Clinic Interactive Dashboard with the 'Appointments' tab selected. On the left, there are summary boxes for Total Patients (60270), Total Appointments (106987), and Total Neighbourhoods (81). The main area displays a table of appointment details with 10699 rows. At the top of the table area, there are filters for 'Select Range for Appointment Date' (set to 2016-04-01 - 2016-07-01), 'Showed Up' (True or False), 'SMS Received' (True or False), and 'Date Difference' (Min and Max fields). Below the filters are 'Reset Filters' and 'Apply Filters' buttons. The table columns include PatientID, Appointment ID, Scheduled Day, Appointment Day, Date Difference, SMS Received, and Showed-Up.

PatientID	Appointment ID	Scheduled Day	Appointment Day	Date Difference	SMS Received	Showed-Up
55899776694438	5642503	2016-04-29	2016-04-29	0	No	Yes
867951213174	5642828	2016-04-29	2016-04-29	0	No	Yes
8841186448183	5642494	2016-04-29	2016-04-29	0	No	Yes
959851133231274	5626772	2016-04-27	2016-04-29	2	No	Yes
78124564369297	5629123	2016-04-27	2016-04-29	2	No	Yes
863229618887631	5616091	2016-04-25	2016-04-29	4	Yes	No
8734857996885	5641780	2016-04-29	2016-04-29	0	No	Yes
342815551642	5628068	2016-04-27	2016-04-29	2	No	Yes
311284853849	5628907	2016-04-27	2016-04-29	2	No	No
72984587621439	5637975	2016-04-29	2016-04-29	0	No	Yes

Figure 28 - Testing Appointments Date

This screenshot shows the same dashboard setup as Figure 28, but with a much smaller dataset. The filters are set to a date range from 2016-04-01 to 2016-04-30, resulting in 311 total rows. The table structure is identical, displaying appointment details for each row.

PatientID	Appointment ID	Scheduled Day	Appointment Day	Date Difference	SMS Received	Showed-Up
55899776694438	5642503	2016-04-29	2016-04-29	0	No	Yes
867951213174	5642828	2016-04-29	2016-04-29	0	No	Yes
8841186448183	5642494	2016-04-29	2016-04-29	0	No	Yes
959851133231274	5626772	2016-04-27	2016-04-29	2	No	Yes
78124564369297	5629123	2016-04-27	2016-04-29	2	No	Yes
863229618887631	5616091	2016-04-25	2016-04-29	4	Yes	No
8734857996885	5641780	2016-04-29	2016-04-29	0	No	Yes
342815551642	5628068	2016-04-27	2016-04-29	2	No	Yes
311284853849	5628907	2016-04-27	2016-04-29	2	No	No
72984587621439	5637975	2016-04-29	2016-04-29	0	No	Yes

Figure 29 - Testing Appointments for April