GG-e3-project

January 7, 2024

1 GG E3 Project

- ist100071 Ricardo Silva (33%)
- ist99892 Andre Godinho (33%)
- ist96147 Alice Gamboa (33%)

Prof. Alessandro Gianola

Lab Shift number: PB03

1.1 PART I – Creating Views for a Dashboard

Create views over the tables in the database model, corresponding to the following relational schema.

dim_date(date, day, month, year) >IC: date corresponds to a date existing in consultations

dim_client(VAT, gender, age) > VAT: FK(client)

dim_location(zip, city) > IC: zip corresponds to a zip code existing in clients

facts_consultations(VAT, date, zip, num_diagnostic_codes, num_procedures) > VAT: FK(dim_client)

date: FK(dim_date)

zip: FK(dim_location)

[4]:
```
%load_ext sql
%sql postgresql+psycopg://clinic:clinic@postgres/clinic
```

[18]:
```
%config SqlMagic.displaylimit = 30
```

[ ]:
```
%%sql
DROP VIEW IF EXISTS consultation_details;
DROP VIEW IF EXISTS facts_consultations;
DROP VIEW IF EXISTS dim_date;
DROP VIEW IF EXISTS dim_client;
DROP VIEW IF EXISTS dim_location;

CREATE VIEW dim_date(date, day_, month_, year_)
```

```
AS
SELECT date_timestamp, EXTRACT(day FROM date_timestamp), EXTRACT(month FROM␣
  ↪date_timestamp), EXTRACT(year FROM date_timestamp)
FROM consultation;

CREATE VIEW dim_client(VAT, gender, age)
AS
SELECT VAT_ID AS VAT, gender, EXTRACT(YEAR FROM AGE(CURRENT_DATE, birth_date))␣
  ↪AS age
FROM client;

CREATE VIEW dim_location(zip, city)
AS
SELECT zip_code AS zip, city
FROM client;

CREATE VIEW facts_consultations(VAT, date, zip, num_diagnostic_codes,␣
  ↪num_procedures)
AS
SELECT dc.VAT AS VAT, dd.date AS date, dl.zip AS zip, COUNT(DISTINCT cd.ID) AS␣
  ↪num_diagnostic_codes, COUNT(pc.name) AS num_procedures
FROM consultation c
JOIN appointment a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.
  ↪date_timestamp
JOIN dim_client dc ON a.VAT_client = dc.VAT
JOIN dim_date dd ON c.date_timestamp = dd.date
JOIN client cl ON cl.VAT_ID = dc.VAT
JOIN dim_location dl ON cl.zip_code = dl.zip
LEFT JOIN consultation_diagnostic cd ON c.VAT_doctor = cd.VAT_doctor AND c.
  ↪date_timestamp = cd.date_timestamp
LEFT JOIN Procedure_in_consultation pc ON c.VAT_doctor = pc.VAT_doctor AND c.
  ↪date_timestamp = pc.date_timestamp
GROUP BY dc.VAT, dd.date, dl.zip;

CREATE VIEW consultation_details(vat_client, vat_doctor, date_timestamp,␣
  ↪description, nurse_id, soap_s, soap_o, soap_a, soap_p, diag_desc, med_name,␣
  ↪lab, dosage, pr_description)
AS
SELECT ap.vat_client,
    ap.vat_doctor,
    ap.date_timestamp,
    ap.description,
    ca.vat_nurse AS nurse_id,
    co.soap_s,
    co.soap_o,
    co.soap_a,
    co.soap_p,
    d.description AS diag_desc,
    pr.name AS med_name,
    pr.lab,
    pr.dosage,
    pr.description AS pr_description
   FROM client c
     JOIN appointment ap ON c.vat_id = ap.vat_client
     LEFT JOIN consultation co ON ap.vat_doctor = co.vat_doctor AND ap.
 ↪date_timestamp = co.date_timestamp
     LEFT JOIN consultation_assistant ca ON ap.vat_doctor = ca.vat_doctor AND␣
 ↪ap.date_timestamp = ca.date_timestamp
     LEFT JOIN consultation_diagnostic cd ON ap.vat_doctor = cd.vat_doctor AND␣
 ↪ap.date_timestamp = cd.date_timestamp
     LEFT JOIN prescription pr ON pr.vat_doctor = cd.vat_doctor AND pr.
 ↪date_timestamp = cd.date_timestamp AND pr.id = cd.id
     LEFT JOIN diagnostic_code d ON pr.id = d.id;

SELECT * FROM facts_consultations;
```

1.2 PART II – Indexes

Suggest indexes that could improve the performance of the following query:

SELECT > VAT,

COUNT(*) AS num_consultations,

SUM(num_procedures) AS total_procedures

FROM > facts_consultations

GROUP BY >VAT

ORDER BY > total_procedures;

Provide SQL instructions for implementing the most appropriate indexes. Justify your choice and provide the corresponding query plan(s).

[ ]:
```
%%sql

DROP INDEX IF EXISTS idx_client_vat;
DROP INDEX IF EXISTS idx_client_zip;
DROP INDEX IF EXISTS idx_consultation;
DROP INDEX IF EXISTS idx_appointment;
DROP INDEX IF EXISTS idx_consultation_diagnostic;
DROP INDEX IF EXISTS idx_procedure_in_consultation;

CREATE INDEX idx_client_vat ON client(VAT_ID);
CREATE INDEX idx_client_zip ON client(zip_code);
CREATE INDEX idx_consultation ON consultation(VAT_doctor, date_timestamp);
CREATE INDEX idx_appointment ON appointment(VAT_doctor, date_timestamp);
CREATE INDEX idx_consultation_diagnostic ON consultation_diagnostic(VAT_doctor,␣
 ↪date_timestamp);
CREATE INDEX idx_procedure_in_consultation ON␣
 ↪procedure_in_consultation(VAT_doctor, date_timestamp);

EXPLAIN(ANALYZE, BUFFERS)
SELECT VAT,
       COUNT(*) AS num_consultations,
       SUM(num_procedures) AS total_procedures
FROM facts_consultations
GROUP BY VAT
ORDER BY total_procedures;
```

Index idx_client_vat: The VAT_ID column in the client table is constantly used in joins involving the client table and the dim_client view and in the group by clause in the query.

Index idx_client_zip: The zip column in the client table is constantly used in joins involving the client table and the dim_location view.

Index on idx_appointment: This query involves joining with the appointment table (because of the facts_consultations view). A composite index on (VAT_doctor, date_timestamp) in this table could improve join performance.

Index idx_consultation: This query involves joining with the consultation table (because of the facts_consultations view). A composite index on (VAT_doctor, date_timestamp) in this table could improve join performance.

Index idx_consultation_diagnostic: The consultation_diagnostic table uses VAT_doctor and date_timestamp for joins. This index helps the facts_consultations view optimizing the order by total_procedures.

Index idx_procedure_in_consultation: The procedure_in_consultation table also uses VAT_doctor and date_timestamp for joins. This index helps the count clause in the facts_consultations view.

1.2.1 QUERY PLAN

Sort (cost=12.27..12.30 rows=11 width=98) (actual time=0.129..0.133 rows=9 loops=1) Sort Key: (sum(COALESCE(proc_info.num_procedures, '0'::bigint))) Sort Method: quicksort Memory: 25kB Buffers: shared hit=8 -> HashAggregate (cost=11.94..12.08 rows=11 width=98) (actual time=0.118..0.124 rows=9 loops=1) Group Key: client.vat_id Batches: 1 Memory Usage: 24kB Buffers: shared hit=8 -> Hash Left Join (cost=7.62..11.85 rows=13 width=66) (actual time=0.063..0.114 rows=13 loops=1) Hash Cond: (((c.vat_doctor)::text = (proc_info.vat_doctor)::text) AND (c.date_timestamp = proc_info.date_timestamp)) Buffers:

shared hit=8 -> Hash Left Join (cost=6.36..10.52 rows=13 width=124) (actual time=0.050..0.096 rows=13 loops=1) Hash Cond: ((cl.zip_code)::text = (client_1.zip_code)::text) Buffers: shared hit=7 -> Hash Join (cost=5.11..9.09 rows=13 width=166) (actual time=0.044..0.086 rows=13 loops=1) Hash Cond: ((a.vat_client)::text = (cl.vat_id)::text) Buffers: shared hit=6 -> Hash Left Join (cost=3.87..7.80 rows=13 width=134) (actual time=0.028..0.065 rows=13 loops=1) Hash Cond: (c.date_timestamp = consultation.date_timestamp) Buffers: shared hit=5 -> Hash Join (cost=2.57..6.33 rows=13 width=134) (actual time=0.021..0.056 rows=13 loops=1) Hash Cond: ((a.vat_client)::text = (client.vat_id)::text) Buffers: shared hit=4 -> Hash Join (cost=1.32..5.03 rows=13 width=76) (actual time=0.015..0.044 rows=13 loops=1) Hash Cond: (((a.vat_doctor)::text = (c.vat_doctor)::text) AND (a.date_timestamp = c.date_timestamp)) Buffers: shared hit=3 -> Seq Scan on appointment a (cost=0.00..3.12 rows=112 width=28) (actual time=0.003..0.011 rows=112 loops=1) Buffers: shared hit=2 -> Hash (cost=1.13..1.13 rows=13 width=66) (actual time=0.007..0.008 rows=13 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9kB Buffers: shared hit=1 -> Seq Scan on consultation c (cost=0.00..1.13 rows=13 width=66) (actual time=0.002..0.004 rows=13 loops=1) Buffers: shared hit=1 -> Hash (cost=1.11..1.11 rows=11 width=58) (actual time=0.005..0.005 rows=11 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9kB Buffers: shared hit=1 -> Seq Scan on client (cost=0.00..1.11 rows=11 width=58) (actual time=0.001..0.002 rows=11 loops=1) Buffers: shared hit=1 -> Hash (cost=1.13..1.13 rows=13 width=8) (actual time=0.004..0.004 rows=13 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9kB Buffers: shared hit=1 -> Seq Scan on consultation (cost=0.00..1.13 rows=13 width=8) (actual time=0.001..0.002 rows=13 loops=1) Buffers: shared hit=1 -> Hash (cost=1.11..1.11 rows=11 width=100) (actual time=0.013..0.013 rows=11 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9kB Buffers: shared hit=1 -> Seq Scan on client cl (cost=0.00..1.11 rows=11 width=100) (actual time=0.006..0.008 rows=11 loops=1) Buffers: shared hit=1 -> Hash (cost=1.11..1.11 rows=11 width=42) (actual time=0.004..0.004 rows=11 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9kB Buffers: shared hit=1 -> Seq Scan on client client_1 (cost=0.00..1.11 rows=11 width=42) (actual time=0.001..0.002 rows=11 loops=1) Buffers: shared hit=1 -> Hash (cost=1.19..1.19 rows=5 width=74) (actual time=0.012..0.012 rows=4 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9kB Buffers: shared hit=1 -> Subquery Scan on proc_info (cost=1.09..1.19 rows=5 width=74) (actual time=0.008..0.010 rows=4 loops=1) Buffers: shared hit=1 -> HashAggregate (cost=1.09..1.14 rows=5 width=74) (actual time=0.008..0.009 rows=4 loops=1) Group Key: procedure_in_consultation.vat_doctor, procedure_in_consultation.date_timestamp Batches: 1 Memory Usage: 24kB Buffers: shared hit=1 -> Seq Scan on procedure_in_consultation (cost=0.00..1.05 rows=5 width=66) (actual time=0.002..0.003 rows=5 loops=1) Buffers: shared hit=1 Planning: Buffers: shared hit=94 read=5 Planning Time: 2.109 ms Execution Time: 0.216 ms

Although there is not any reference to indexes in the query plan, we believe that these indexes could actually improve the query performance in a more populated database than ours.

1.3 PART III - A Web Application Using the Database

Develop a simple Web-based application leveraging the database created and populated in Part 2 to perform the following tasks:

1. A client comes to the clinic asking for an appointment. We need to check if the client already exists in the database, and check if there is a doctor available for the desired date/time. You should therefore create Web pages to support these verifications and the search for matching clients based on different information elements: given the VAT, a (part of the) name for the client, and/or the (parts of the) address, you should display the records of matching clients.

On the displayed result, include the possibility of registering a new appointment for the listed clients. Include also options for adding new clients to the database, and for listing the doctors that are available for consultations at a given date/time (you can consider that doctors can give consultations on any day, and that all consultations last for one hour, starting at exact hours between 9AM and 5PM).

2. Create a set of Web pages to support the access and registry of information associated with an appointment/consultation. Selecting (or clicking on) a given client in the results page of the previous interaction should lead to another page listing, chronologically, all previous appointments and consultations for that client. Selecting (or clicking on) an appointment/consult from this list should lead to a page presenting all information on the appointment/consult, including the SOAP notes, existing diagnostic codes, and any existing prescriptions. There should also be an option for adding information for a new consultation (i.e., add information to an appointment created to the interaction from the previous point), including the assisting nurse(s), SOAP notes, diagnostic codes, and prescriptions.
3. Create a Dashboard Web page that uses the facts_consultations View and OLAP queries.

[ ]:
```
###App.py
#!/usr/bin/python3
# Copyright (c) BDist Development Team
# Distributed under the terms of the Modified BSD License.
import os
from logging.config import dictConfig
import psycopg
from flask import Flask, flash, jsonify, redirect, render_template, request,␣
 ↪url_for
from psycopg.rows import namedtuple_row
from datetime import datetime
# postgres://{user}:{password}@{hostname}:{port}/{database-name}
DATABASE_URL = os.environ.get("DATABASE_URL", "postgres://clinic:
 ↪clinic@postgres/clinic")

dictConfig(
    {
        "version": 1,
        "formatters": {
            "default": {
                "format": "[%(asctime)s] %(levelname)s in %(module)s:%(lineno)s␣
 ↪- %(funcName)20s(): %(message)s",
            }
        },
        "handlers": {
            "wsgi": {
                "class": "logging.StreamHandler",
                "stream": "ext://flask.logging.wsgi_errors_stream",
                "formatter": "default",
            }
        },
        "root": {"level": "INFO", "handlers": ["wsgi"]},
    }
)

app = Flask(__name__)
app.config.from_prefixed_env()
log = app.logger


def is_decimal(s):
    """Returns True if string is a parseable float number."""
    try:
        float(s)
        return True
    except ValueError:
        return False

def fetch_clients(vat_id=None, name=None, location=None):
    log.debug(f"Vat Id:{vat_id} Name: {name} Location:{location}")
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            if vat_id and name and location:
                cur.execute(
                    """
                    SELECT vat_id, name, street, city, zip_code
                    FROM client
                    WHERE vat_id = %(vat_id)s AND name ILIKE %(name)s AND␣
 ↪(street ILIKE %(location)s OR city ILIKE %(location)s OR zip_code ILIKE␣
 ↪%(location)s);
                    """,
                    {"vat_id": vat_id, "name": name, "location":␣
 ↪f"%{location}%"},
                )
            elif vat_id and name:
                cur.execute(
                    """
                    SELECT vat_id, name, street, city, zip_code
                    FROM client
                    WHERE vat_id = %(vat_id)s AND name ILIKE %(name)s;
                    """,
                    {"vat_id": vat_id, "name": f"%{name}%"},
                )
            elif name and location:
                cur.execute(
                    """
                    SELECT vat_id, name, street, city, zip_code
                    FROM client
                    WHERE name ILIKE %(name)s AND (street ILIKE %(location)s OR␣
    ↪city ILIKE %(location)s OR zip_code ILIKE %(location)s);
                    """,
                    {"name": f"%{name}%", "location": f"%{location}%"},
                )
            elif vat_id and location:
                cur.execute(
                    """
                    SELECT vat_id, name, street, city, zip_code
                    FROM client
                    WHERE vat_id = %(vat_id)s AND (street ILIKE %(location)s OR␣
    ↪city ILIKE %(location)s OR zip_code ILIKE %(location)s);
                    """,
                    {"vat_id": vat_id, "location": f"%{location}%"},
                )
            elif vat_id:
                cur.execute(
                    """
                    SELECT vat_id, name, street, city, zip_code
                    FROM client
                    WHERE vat_id = %(vat_id)s;
                    """,
                    {"vat_id": vat_id},
                )
            elif name:
                cur.execute(
                    """
                    SELECT vat_id, name, street, city, zip_code
                    FROM client
                    WHERE name ILIKE %(name)s;
                    """,
                    {"name": f"%{name}%"},
                )
            elif location:
                cur.execute(
                    """
                    SELECT vat_id, name, street, city, zip_code
                    FROM client
                    WHERE street ILIKE %(location)s OR city ILIKE %(location)s␣
    ↪OR zip_code ILIKE %(location)s;
                    """,
                    {"location": f"%{location}%"},
                )
            else:
                cur.execute(
                    """
                SELECT vat_id, name, street, city, zip_code
                FROM client;
                """
            )
            clients = cur.fetchall()
            log.debug(f"Found {cur.rowcount} rows.")

    return clients

def create_client(vat_id, name, birth_date, street, city=None, zip_code=None,␣
↪gender=None):
    log.debug(f"Vat Id:{vat_id} Name: {name} BirthDate: {birth_date} Street:
↪{street} City: {city} ZipCode: {zip_code} Gender: {gender} lenght:␣
↪{len(gender)}")
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            query = """
                    INSERT INTO client
                    VALUES (%s, %s, %s, %s, %s, %s, %s);
                    """
            cur.execute(query, (vat_id, name, birth_date, street, city,␣
↪zip_code, gender))
            conn.commit()
    return


def fetch_doctors(timestamp=None):
    log.debug(f"Time:{timestamp}")
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            if timestamp:
                cur.execute(
                    """
                    SELECT d.VAT_ID AS vat_id, e.name AS name
                    FROM doctor AS d
                    JOIN employee AS e ON e.VAT_ID = d.VAT_ID
                    LEFT JOIN appointment AS ap ON d.VAT_ID = ap.VAT_doctor AND␣
↪ap.date_timestamp = %(timestamp)s
                    WHERE ap.VAT_doctor IS NULL;
                    """,
                    {"timestamp": timestamp},
                )
            else:
                cur.execute(
                    """
                    SELECT d.VAT_ID AS vat_id, e.name AS name
                    FROM doctor AS d
                    JOIN employee AS e ON e.VAT_ID = d.VAT_ID
                    """,
                    {},
                )
            doctors = cur.fetchall()
    return doctors


def fetch_nurses():
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute(
                """
                SELECT n.vat_id AS vat_id, e.name AS name
                FROM nurse AS n
                JOIN employee AS e ON e.vat_id = n.vat_id
                """,
                {},
            )
            nurses = cur.fetchall()
    return nurses

def fetch_diag():
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute(
                """
                SELECT ID, description
                FROM diagnostic_code
                """,
                {},
            )
            diag = cur.fetchall()
    return diag


def fetch_meds():
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute(
                """
                SELECT name, lab
                FROM medication
                """,
                {},
            )
            meds = cur.fetchall()
    return meds

def fetch_appointments_consultations(client_id):
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
            with conn.cursor(row_factory=namedtuple_row) as cur:
                cur.execute(
                    """
                    SELECT ap.vat_doctor AS vat_doctor, e.name AS doctor_name,␣
 ↪ap.date_timestamp AS date_timestamp, ap.description AS description
                    FROM client AS c
                    JOIN appointment AS ap ON c.vat_id = ap.vat_client
                    JOIN employee AS e ON e.vat_id = ap.vat_doctor
                    LEFT JOIN consultation AS co ON ap.vat_doctor = co.
 ↪vat_doctor AND ap.date_timestamp = co.date_timestamp
                    WHERE ap.vat_client = %(client_id)s
                    ORDER BY ap.date_timestamp;
                    """,
                    {"client_id":client_id},
                )
                appointments = cur.fetchall()
                return appointments

def fetch_consultation_details(client_id, consultation_date, doctor_id):
    log.debug(f"Client ID: {client_id} DateTime: {consultation_date} Doctor ID:␣
 ↪{doctor_id}")
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute(
                """
                SELECT * FROM consultation_details
                WHERE vat_client = %(client_id)s AND vat_doctor = %(doctor_id)s␣
 ↪AND date_timestamp = %(consultation_date)s ;
                """,
                {"client_id":client_id, "doctor_id":doctor_id,␣
 ↪"consultation_date":consultation_date},
            )
            appointment_details = cur.fetchall()
        return appointment_details

def query_lab_for_medication(med_name):
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute(
                """
                SELECT lab
                FROM medication
                WHERE name = %(med_name)s;
                """,
                {"med_name": med_name}),
            lab = cur.fetchone()
    return lab

def create_appointment(doctor_id, timestamp_str, client_id, description):
    log.debug(f"Client Id:{client_id} Timestamp: {timestamp_str} Doctor Id:␣
↪{doctor_id} Description:{description}")
    try:
        with psycopg.connect(conninfo=DATABASE_URL) as conn:
            with conn.cursor(row_factory=namedtuple_row) as cur:
                query = """
                        INSERT INTO appointment
                        VALUES (%s, %s, %s, %s);
                        """
                cur.execute(query, (doctor_id, timestamp_str, client_id,␣
↪description))
                conn.commit()
    except Exception as e:
        log.error(f"Error in create_appointment: {e}")

def create_consultation(doctor_id, nurse_id, consultation_date, diag_id,␣
↪soap_s, soap_o, soap_a, soap_p, prescriptions):
    log.debug(f"Doctor Id:{doctor_id} Nurse Id:{nurse_id} Timestamp:␣
↪{consultation_date} ID: {diag_id}, SOAP S: {soap_s} SOAP O: {soap_o} SOAP A:␣
↪{soap_a} SOAP P: {soap_p} Prescriptions: {prescriptions}")
    try:
        with psycopg.connect(conninfo=DATABASE_URL) as conn:
            with conn.cursor(row_factory=namedtuple_row) as cur:
                query = """
                        INSERT INTO consultation
                        VALUES (%s, %s, %s, %s, %s, %s);
                        """
                cur.execute(query, (doctor_id, consultation_date, soap_s,␣
↪soap_o, soap_a, soap_p))
                conn.commit()

                query = """
                        INSERT INTO consultation_assistant
                        VALUES (%s, %s, %s);
                        """
                cur.execute(query, (doctor_id, consultation_date, nurse_id))
                conn.commit()

                cur.execute("""
                    INSERT INTO consultation_diagnostic
                    VALUES (%s, %s, %s);
                """, (doctor_id, consultation_date, diag_id))
                conn.commit()

                for presc in prescriptions:
                    cur.execute("""
                        INSERT INTO prescription
                        VALUES (%s, %s, %s, %s, %s, %s, %s);
                    """, (doctor_id, consultation_date, diag_id,␣
 ↪presc['med_name'], presc['med_lab'], presc['dosage'], presc['description']))
                conn.commit()

    except Exception as e:
        log.error(f"Error in create_consultation: {e}")

def fetch_client_cons(client_id):
    try:
        with psycopg.connect(conninfo=DATABASE_URL) as conn:
            with conn.cursor(row_factory=namedtuple_row) as cur:
                cur.execute(
                    """
                    SELECT dd.year_ AS year, SUM(fc.num_diagnostic_codes) AS␣
 ↪total_diagnostic_codes, SUM(fc.num_procedures) AS total_procedures
                    FROM facts_consultations fc
                    JOIN dim_date dd ON fc.date = dd.date
                    WHERE fc.VAT = %(client_id)s
                    GROUP BY ROLLUP(dd.year_)
                    ORDER BY year;
                    """,
                    {"client_id":client_id},
                )
                client_cons_data = cur.fetchall()
            return client_cons_data
    except Exception as e:
        log.error(f"Error in create_consultation: {e}")

def aggregate_consultations(clients_data):
    aggregated_clients = []
    for client in clients_data:
        update_idx = None
        client_id, name, _, _, _ = client
        client_cons_data = fetch_client_cons(client_id)
        log.debug(client_cons_data)

        for idx, aggregated_client in enumerate(aggregated_clients):
            if aggregated_client['id'] == client_id:
                update_idx = idx
                break
        if update_idx is not None:
            continue
        else:
            # Client doesn't exist, create a new entry
            new_client_entry = {'id': client_id, 'name': name,
                                'consultations':
                                    [{
                                        'year': int(row.year) if row.year is␣
↪not None else None,
                                        'total_diagnostic_codes': int(row.
↪total_diagnostic_codes) if row.total_diagnostic_codes is not None else 0,
                                        'total_procedures': int(row.
↪total_procedures) if row.total_procedures is not None else 0
                                    }
                                     for row in client_cons_data]
                                }
            aggregated_clients.append(new_client_entry)

    return aggregated_clients

@app.route("/", methods=("GET", "POST",))
@app.route("/clients", methods=("GET", "POST",))
def clients_index():
    """Show all the clients."""
    if request.method == "POST":
        vat_id = request.form.get("vat_id")
        name = request.form.get("name")
        location = request.form.get("location")
        clients = fetch_clients(vat_id, name, location)
    else:
        clients = fetch_clients()

    return render_template("clinic/index.html", clients=clients)

@app.route("/doctors", methods=("GET", "POST",))
def doctors_index():
    """Show all the doctors."""
    if request.method == "POST":
        date = datetime.strptime(request.form.get("date"), "%Y-%m-%d")
        time = datetime.strptime(request.form.get("time"), "%H:%M").time()
        combined_datetime = datetime.combine(date, time)
        timestamp_str = combined_datetime.strftime('%Y-%m-%d %H:%M:%S')
        doctors = fetch_doctors(timestamp_str)
    else:
        doctors = fetch_doctors()

    return render_template("clinic/doctors.html", doctors=doctors)
```

```
@app.route("/dashboard", methods=("GET",))
def dashboard_index():
    """Show the dashboard."""
    client_facts = fetch_clients()
    clients = aggregate_consultations(client_facts)
    return render_template("clinic/dashboard.html", clients=clients)

@app.route("/create_client", methods=("GET","POST",))
def create_client_view():
    """Create clients."""
    if request.method == "POST":
        vat_id = request.form.get("vat_id")
        name = request.form.get("name")
        birth_date = datetime.strptime(request.form.get("birth_date"),␣
 ↪"%Y-%m-%d")
        street = request.form.get("street")
        city = request.form.get("city")
        zip_code = request.form.get("zip_code")
        gender = request.form.get("gender")
        create_client(vat_id, name, birth_date, street, city, zip_code, gender)
        return redirect(url_for("clients_index"))

    return render_template("clinic/create_client.html")

@app.route("/appointments/<client_id>", methods=("GET",))
def client_appointment_view(client_id):
    """Show client's appointments and consultations."""

    appointments = fetch_appointments_consultations(client_id)
    return render_template("clinic/appointment.html", client_id=client_id,␣
 ↪appointments=appointments)

@app.route("/create_appointment/<client_id>", methods=("GET","POST",))
def create_appointment_view(client_id):
    """Show client's appointments and consultations."""
    doctors = []
    if request.method == "POST":
        log.debug("POSTTTT")
        log.debug(request.form.get("date"))
        # Check if all required fields are present
        if request.form.get("date") != None:
            date = datetime.strptime(request.form.get("date"), "%Y-%m-%d")
        if request.form.get("time") != None:
            time = datetime.strptime(request.form.get("time"), "%H:%M").time()
        combined_datetime = datetime.combine(date, time)
        timestamp_str = combined_datetime.strftime('%Y-%m-%d %H:%M:%S')
        doctor_id = request.form.get("doctor_id")
        description = request.form.get("description")

        # Assuming you have some logic to fetch doctors
        doctors = fetch_doctors(timestamp_str)
        if timestamp_str != None and doctor_id != None and description != None:
            create_appointment(doctor_id, timestamp_str, client_id, description)
            return redirect(url_for('client_appointment_view',␣
 ↪client_id=client_id))

    return render_template("clinic/create_appointment.html",␣
 ↪client_id=client_id, doctors=doctors)

@app.route("/create_consultation/<client_id>/<doctor_id>/<consultation_date>",␣
 ↪methods=("GET","POST",))
def create_consultation_view(client_id, doctor_id, consultation_date):
    """Show client's appointments and consultations."""
    log.debug(f"Doctor ID: {doctor_id} Date: {consultation_date}")
    nurses = fetch_nurses()
    medications = fetch_meds()
    diagnostic_codes = fetch_diag()
    if request.method == "POST":
        nurse_id = request.form.get("nurse_id")
        soap_s = request.form.get("soap_s")
        soap_o = request.form.get("soap_o")
        soap_a = request.form.get("soap_a")
        soap_p = request.form.get("soap_p")
        diag_id = request.form.get("diag_code")
        # Diagnostic Codes
        num_presc = int(request.form.get("num_presc"))
        prescriptions = []
        for i in range(num_presc):
            med_name = request.form.getlist("med_name[]")[i]
            med_lab = query_lab_for_medication(med_name)[0]
            dosage = request.form.getlist("dosage[]")[i]
            description = request.form.getlist("description[]")[i]
            log.debug(f"MED NAME: {med_name} MED LAB: {med_lab} DOSAGE:␣
 ↪{dosage} DESCRIPTION: {description}")
            prescriptions.append({
                "med_name": med_name,
                "med_lab": med_lab,
                "dosage": dosage,
                "description": description
            })
        create_consultation(doctor_id, nurse_id, consultation_date, diag_id,␣
 ↪soap_s, soap_o, soap_a, soap_p, prescriptions)
```

```
        return redirect(url_for('consultation_details_view',␣
↪client_id=client_id, doctor_id=doctor_id,␣
↪consultation_date=consultation_date))

    else:
        return render_template("clinic/create_consultation.html",␣
↪client_id=client_id, doctor_id=doctor_id,␣
↪consultation_date=consultation_date, nurses=nurses, medications=medications,␣
↪diagnostic_codes=diagnostic_codes)

@app.route("/consultation/<client_id>/<doctor_id>/<consultation_date>",␣
↪methods=("GET",))
def consultation_details_view(client_id, doctor_id, consultation_date):
    """Show client's consultation details."""
    log.debug(f"Doctor ID: {doctor_id} Date: {consultation_date}")
    appointment_details = fetch_consultation_details(client_id,␣
↪consultation_date, doctor_id)
    prescription = None
    if appointment_details[0].med_name != None:
        prescription = "\n".join([
            f"{appointment_detail.med_name}, {appointment_detail.lab},␣
↪{appointment_detail.dosage}, {appointment_detail.pr_description}"
            for appointment_detail in appointment_details
        ])
    return render_template("clinic/consultation_profile.html",␣
↪client_id=client_id, appointment_details=appointment_details[0],␣
↪prescription=prescription)

if __name__ == "__main__":
    app.run(debug=False)
```

[ ]:
```
###base.html
<!doctype html>
<title>{% block title %}{% endblock %} - Clinic</title>
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
<nav>
  <h1><a href="/">Clinic</a></h1>
  <ul>
    <li><a href="{{ url_for('clients_index') }}">Clients</a>
      <li><a href="{{ url_for('doctors_index') }}">Doctors</a>
    </ul>
  </nav>
  <section class="content">
    <header>
      {% block header %}{% endblock %}
    </header>
    {% for message in get_flashed_messages() %}
      <div class="flash">{{ message }}</div>
    {% endfor %}
    {% block content %}{% endblock %}
  </section>
```

[ ]:
```
###index.html
{% extends 'base.html' %}

{% block header %}
  <h1>{% block title %}Clients{% endblock %}</h1>
  <a class="view-link back-link" href="{{ url_for('dashboard_index')␣
↪}}">DashBoard</a>
  <a class="view-link back-link" href="{{ url_for('create_client_view')␣
↪}}">Create Client</a>
  <a class="view-link" href="/">Back</a>
{% endblock %}

{% block content %}
  <form method="post" action="{{ url_for('clients_index') }}">
    <input name="vat_id" id="vat_id" type="text" size="20" value="{{ request.
↪form['vat_id'] }}" placeholder="Vat Id">
    <input name="name" id="name" type="text" size="80" value="{{ request.
↪form['name'] }}" placeholder="Name">
    <input name="location" id="location" type="text" size="255" value="{{␣
↪request.form['location'] }}" placeholder="Location">
    <input type="submit" value="Search">
  </form>
  <hr>
  {% for client in clients %}
    <article class="post">
      <header>
        <div>
          <h1>{{ client['vat_id'] }}</h1>
          <div class="about">{{ client['name'] }}</div>
          <div class="about">{{ client['city'] +", "+ client['street'] +", "+␣
↪client['zip_code'] }}</div>
        </div>
        <a class="view-link" href="{{ url_for('client_appointment_view',␣
↪client_id=client['vat_id']) }}">View Profile</a>
      </header>
    </article>
    {% if not loop.last %}
    <hr>
    {% endif %}
  {% endfor %}
{% endblock %}
```

[ ]:
```
###doctors.html
{% extends 'base.html' %}

{% block header %}
  <h1>{% block title %}Doctors | Availability{% endblock %}</h1>
{% endblock %}

{% block content %}
  <form method="post" action="{{ url_for('doctors_index') }}">
    <input name="date" id="date" type="date" value="{{ request.form['date'] }}">
    <label for="time">Time:</label>
    <select id="time" name="time">
        {% for hour in range(9, 18) %} <!-- 9 AM to 5 PM -->
        <option value="{{ '%02d:00' % hour }}">{{ '%02d:00' % hour }}</option>
        {% endfor %}
    </select>
    <input type="submit" value="Search">
  </form>
  {% for doctor in doctors %}
    <article class="post">
      <header>
        <div>
          <h1>{{ doctor['vat_id'] }}</h1>
          <div class="about">{{ doctor['name'] }}</div>
        </div>
      </header>
    </article>
    {% if not loop.last %}
    <hr>
    {% endif %}
  {% endfor %}
{% endblock %}
```

[ ]:
```
###dashboard.html
{% extends 'base.html' %}

{% block header %}
  <h1>{% block title %}Dashboard | Consultations{% endblock %}</h1>
  <a class="view-link" href="/">Back</a>
{% endblock %}


{% block content %}
    {% for client in clients %}
        <div class="client-box" onclick="toggleDetails('{{ client.id }}')">
```

```
            <h3>{{ client.name }}</h3>
            <div id="{{ client.id }}" class="consultation-details">
                {% if client.consultations|length > 1 %}
                <table border="2">
                    <thead>
                        <tr>
                            <th>Year</th>
                            <th>Number of Diagnostic Codes</th>
                            <th>Number of Procedures</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for consultation in client.consultations %}
                            <tr>
                                <td>{% if consultation.year %}{{ consultation.
↪year }}{% else %}Total{% endif %}</td>
                                <td>{{ consultation.total_diagnostic_codes }}</
↪td>
                                <td>{{ consultation.total_procedures }}</td>
                            </tr>
                        {% endfor %}
                    </tbody>
                </table>
            {% else %}
                <p>No consultations for this client.</p>
            {% endif %}
            </div>
        </div>
    {% endfor %}

    <script>
        function toggleDetails(clientId) {
            var details = document.getElementById(clientId);
            details.style.display = details.style.display === 'none' ? 'block' :
↪ 'none';
        }
    </script>
{% endblock %}
```

[ ]:
```
###create_consultation.html
{% extends 'base.html' %}

{% block header %}
  <h1>{% block title %}Client {{ client_id }} | Create Consultation{% endblock␣
↪%}</h1>
  <a class="view-link" href="/consultation/{{ client_id }}/{{ doctor_id }}/{{␣
↪consultation_date }}">Back</a>
```

```
{% endblock %}
{% block content %}
<form action="{{ url_for('create_consultation_view', client_id=client_id,␣
↪doctor_id=doctor_id, consultation_date=consultation_date)}}" method="post">

    <label for="nurse_id">Consultation Assistant*</label>
    <select id="nurse_id" name="nurse_id" required>
        {% for nurse in nurses %}
        <option value="{{ nurse['vat_id'] }}">{{ nurse['name'] }}</option>
        {% endfor %}
    </select>

    <label for="soap_s">Subjective Observation*</label>
    <textarea name="soap_s" id="soap_s" type="text" required>{{ request.
↪form['soap_s'] }}</textarea>

    <label for="soap_o">Objective Observation*</label>
    <textarea name="soap_o" id="soap_o" type="text" required>{{ request.
↪form['soap_o'] }}</textarea>

    <label for="soap_a">Assessment*</label>
    <textarea name="soap_a" id="soap_a" type="text" required>{{ request.
↪form['soap_a'] }}</textarea>

    <label for="soap_p">Plan*</label>
    <textarea name="soap_p" id="soap_p" type="text" required>{{ request.
↪form['soap_p'] }}</textarea>

    <label for="diag_code">Diagnostic Code*</label>
    <select id="diag_code" name="diag_code" required>
        {% for diagnostic_code in diagnostic_codes %}
        <option value="{{ diagnostic_code['id'] }}">{{␣
↪diagnostic_code['description'] }}</option>
        {% endfor %}
    </select>

    <div id="consultation-diagnostics">
        <!-- Dynamic section for consultation diagnostics -->
        {% for index in range(1) %}
        <div class="diagnostic-section">
                <h2>Consultation Diagnostic {{ index + 1 }}</h2>

                <label for="med_name">Medication Name*</label>
                <select id="med_name" name="med_name[]" required>
                    {% for medication in medications %}
                    <option value="{{ medication['name'] }}">{{␣
↪medication['name'] }}</option>
                    {% endfor %}
                </select>

                <label for="dosage">Dosage*</label>
                <input name="dosage[]" id="dosage" type="text" size="20"␣
↪required>

                <label for="description">Description*</label>
                <textarea name="description[]" id="description" type="text"␣
↪required></textarea>

            </div>
        {% endfor %}
    </div>

    <button type="button" id="add-diagnostic">Add Another Prescription</button>

    <input type="hidden" id="num-prescriptions" name="num_presc" value="1">

    <input class="save-link" type="submit" value="Save">
</form>

<script>
    var diagCodesSelect = document.getElementById("diag_code");
    diagCodesSelect.size = Math.min(diagCodesSelect.length, 10); // Set a␣
↪maximum size, e.g., 5

    // Add event listener for the "Add Another Diagnostic" button
    var addDiagnosticButton = document.getElementById("add-diagnostic");
    addDiagnosticButton.addEventListener("click", function() {
        var consultationDiagnostics = document.
↪getElementById("consultation-diagnostics");

        // Clone the first diagnostic section
        var newDiagnosticSection = consultationDiagnostics.querySelector(".
↪diagnostic-section").cloneNode(true);

        // Increment the diagnostic number in the header
        var diagnosticNumber = consultationDiagnostics.querySelectorAll(".
↪diagnostic-section").length + 1;
        newDiagnosticSection.querySelector("h2").innerText = "Consultation␣
↪Diagnostic " + diagnosticNumber;

        // Clear values in the cloned section
        newDiagnosticSection.querySelectorAll("select, input, textarea").
↪forEach(function(element) {
            element.value = "";
        });

        consultationDiagnostics.appendChild(newDiagnosticSection);

        // Update the number of prescriptions hidden field
        document.getElementById("num-prescriptions").value = diagnosticNumber;

    });
</script>
{% endblock %}
```

[ ]:
```
###create_client.html
{% extends 'base.html' %}

{% block header %}
  <h1>{% block title %}Create Client{% endblock %}</h1>
{% endblock %}

{% block content %}
<form action="{{ url_for('create_client_view')}}" method="post">
    <label for="vat_id">VAT ID*</label>
    <input name="vat_id" id="vat_id" type="text" size="20" placeholder="vat id"␣
↪value="{{ request.form['vat_id'] }}" required>

    <label for="name">Name*</label>
    <input name="name" id="name" type="text" size="80" placeholder="name"␣
↪value="{{ request.form['name'] }}" required>

    <label for="birth_date">Birth Date*</label>
    <input name="birth_date" id="birth_date" type="date"␣
↪placeholder="YYYY-MM-DD" value="{{ request.form['birth_date'] }}" required>

    <label for="street">Street*</label>
    <input name="street" id="street" type="text" size="30" placeholder="street"␣
↪value="{{ request.form['street'] }}" required>

    <label for="city">City</label>
    <input name="city" id="city" type="text" size="30" placeholder="city"␣
↪value="{{ request.form['city'] }}" >

    <label for="zip_code">Zip Code</label>
    <input name="zip_code" id="zip_code" type="text" size="12"␣
↪placeholder="XXXX-XXX" value="{{ request.form['zip_code'] }}" >
```

```
    <label for="gender">Gender</label>
    <input name="gender" id="gender" type="text" size="1" placeholder="M/F"␣
 ↪value="{{ request.form['gender'] }}" >

    <input type="submit" value="Save">
  </form>
{% endblock %}
```

[ ]:
```
###create_appointment.html
{% extends 'base.html' %}

{% block header %}
  <h1>{% block title %}Create Appointment | Client {{ client_id }}{% endblock␣
 ↪%}</h1>
{% endblock %}

{% block content %}
{% if not doctors %}
<form method="post" action="{{url_for('create_appointment_view', client_id =␣
 ↪client_id)}}">
    <label for="date">Date:</label>
    <input type="date" id="date" name="date" value="{{ request.form['date'] }}"␣
 ↪required>

    <label for="time">Time:</label>
    <select id="time" name="time" required>
        {% for hour in range(9, 18) %}
        <option value="{{ '%02d:00' % hour }}">{{ '%02d:00' % hour }}</option>
        {% endfor %}
    </select>
    <input type="submit" value="Check Next">
</form>
{% endif %}

{% if doctors %}
<form method="post" action="{{url_for('create_appointment_view', client_id =␣
 ↪client_id)}}">
    <input type="hidden" name="date" value="{{ request.form['date'] }}">
    <input type="hidden" name="time" value="{{ request.form['time'] }}">

    <label for="doctor_id">Doctors*</label>
    <select id="doctor_id" name="doctor_id" required>
        {% for doctor in doctors %}
        <option value="{{ doctor['vat_id'] }}">{{ doctor['name'] }}</option>
        {% endfor %}
    </select>
    <label for="description">Description*</label>
    <textarea id="description" name="description" size="512" required>{{␣
↪request.form['description'] }}</textarea>

    <input type="submit" value="Create Appointment" name="create_appointment" >
</form>
{% endif %}

{% if error_message %}
<div>{{ error_message }}</div>
{% endif %}

{% endblock %}
```

[ ]:
```
###consultation_profile.html
{% extends 'base.html' %}

{% block header %}
  <h1>{% block title %}Client {{ client_id }} | Consultation Details: {{␣
↪appointment_details['date_timestamp'] }}{% endblock %}</h1>
    {% if not appointment_details['nurse_id'] %}
    <a class="view-link back-link"href="{{ url_for('create_consultation_view',␣
↪client_id=client_id, doctor_id=appointment_details['vat_doctor'],␣
↪consultation_date=appointment_details['date_timestamp']) }}">Create␣
↪Consultation</a>
    {% endif %}
    <a class="view-link" href="/appointments/{{ client_id }}">Back</a>
{% endblock %}

{% block content %}
  <form method="post">
    <label for="doctor_id">Doctor</label>
    <input name="doctor_id" id="doctor_id" type="text" value="{{␣
↪appointment_details['vat_doctor']}}" disabled>
    <label for="nurse_id">Assistant Nurse</label>
    <input name="nurse_id" id="nurse_id" type="text" value="{{ request.
↪form['nurse_id']  or appointment_details['nurse_id']}}" disabled>
    <label for="description">Appointment Description</label>
    <textarea name="description" id="description" type="text" disabled>{{␣
↪appointment_details['description']}}</textarea>
    <label for="soap_o">Objective Observation</label>
    <textarea name="soap_s" id="soap_s" type="text" disabled>{{␣
↪appointment_details['soap_s']}}</textarea>
    <label for="soap_o">Objective Observation</label>
    <textarea name="soap_o" id="soap_o" type="text" disabled>{{␣
↪appointment_details['soap_o']}}</textarea>
    <label for="soap_a">Assessment</label>
    <textarea name="soap_a" id="soap_a" type="text" disabled>{{␣
↪appointment_details['soap_a']}}</textarea>
    <label for="soap_p">Plan</label>
    <textarea name="soap_p" id="soap_p" type="text" disabled>{{␣
↪appointment_details['soap_p']}}</textarea>
    <label for="diag_desc">Diagnostic Description</label>
    <input name="diag_desc" id="diag_desc" type="text" value="{{␣
↪appointment_details['diag_desc']}}" disabled>
    <label for="prescription">Prescription(s)</label>
    <textarea name="prescription" id="prescription" type="text" disabled>{{␣
↪prescription}}</textarea>
  </form>
{% endblock %}
```

[ ]:
```
###appointment.html
{% extends 'base.html' %}

{% block header %}
  <h1>{% block title %}Client {{ client_id }} | Appointments & Consultations{%␣
↪endblock %}</h1>
    <a class="view-link back-link" href="{{ url_for('create_appointment_view',␣
↪client_id = client_id) }}">Create Appointment</a>
    <a class="view-link" href="/">Back</a>
{% endblock %}

{% block content %}
  {% for appointment in appointments %}
    <article class="post">
      <header>
        <div>
          <h1>{{ appointment['vat_doctor'] }}</h1>
          <div class="about">{{ appointment['doctor_name'] }}</div>
          <div class="about">{{ appointment['date_timestamp'] }}</div>
        </div>
        <a class="view-link" href="{{ url_for('consultation_details_view',␣
↪client_id=client_id, consultation_date=appointment['date_timestamp'],␣
↪doctor_id=appointment['vat_doctor']) }}">View Details</a>
      </header>
    </article>
    {% if not loop.last %}
      <hr>
    {% endif %}
  {% endfor %}
{% endblock %}
```

[ ]:
```
###style.css
html {
  font-family: sans-serif;
  background: #eee;
  padding: 1rem;
}

body {
  max-width: 960px;
  margin: 0 auto;
  background: white;
}

h1,
h2,
h3,
h4,
h5,
h6 {
  font-family: serif;
  color: rgb(67, 156, 245);
  margin: 1rem 0;
}

a {
  color: rgb(67, 156, 245);
}

hr {
  border: none;
  border-top: 1px solid lightgray;
}

nav {
  background: lightgray;
  display: flex;
  align-items: center;
  padding: 0 0.5rem;
}

nav h1 {
  flex: auto;
  margin: 0;
}
```

```css
nav h1 a {
  text-decoration: none;
  padding: 0.25rem 0.5rem;
}

nav ul {
  display: flex;
  list-style: none;
  margin: 0;
  padding: 0;
}

nav ul li a,
nav ul li span,
header .action {
  display: block;
  padding: 0.5rem;
}

.content {
  padding: 0 1rem 1rem;
}

.content > header {
  border-bottom: 1px solid lightgray;
  display: flex;
  align-items: flex-end;
}

.content > header h1 {
  flex: auto;
  margin: 1rem 0 0.25rem 0;
}

.flash {
  margin: 1em 0;
  padding: 1em;
  background: #cae6f6;
  border: 1px solid #5b98e9;
}

.post > header {
  display: flex;
  align-items: flex-end;
  font-size: 0.85em;
}
```

```css
.post > header > div:first-of-type {
  flex: auto;
}

.post > header h1 {
  font-size: 1.5em;
  margin-bottom: 0;
}

.post .about {
  color: slategray;
  font-style: italic;
}

.post .body {
  white-space: pre-line;
}

.content:last-child {
  margin-bottom: 0;
}

.content form {
  margin: 1em 0;
  display: flex;
  flex-direction: column;
}

.content label {
  font-weight: bold;
  margin-bottom: 0.5em;
}

.content input,
.content textarea {
  margin-bottom: 1em;
}

.content textarea {
  min-height: 6em;
  resize: vertical;
}

input.danger {
  color: #cc2f2e;
}
```

```css
input[type="submit"] {
  align-self: stretch;
  min-width: 10em;
}

.diagnostic-section {
    border: 1px solid #ccc;
    padding: 10px;
    margin-bottom: 20px;
}

.diagnostic-section label {
    display: block;
    margin-bottom: 5px;
}

.diagnostic-section select,
.diagnostic-section input,
.diagnostic-section textarea {
    width: 100%;
    box-sizing: border-box;
    margin-bottom: 10px;
}

#add-diagnostic {
    margin-top: 10px;
}


.view-link {
  display: inline-block;
  padding: 8px;
  background-color: #5b98e9;
  color: white; /* White text color */
  text-decoration: none;
  border-radius: 5px;
  margin-bottom: 8px;
}

.view-link:hover {
  background-color: rgb(31, 102, 216)
}
.back-link {
  margin-right: 20px; /* Adjust the margin for more space */
}
.save-link {
  margin-top: 20px; /* Adjust the margin for more space */
}

.client-box {
  border: 1px solid #ccc;
  margin: 10px;
  padding: 10px;
  cursor: pointer;
}

.consultation-details {
  display: none;
  margin-top: 10px;
  margin-left: 20px;
}

table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 10px;
}

th, td {
  padding: 10px;
  text-align: left;
  border: 1px solid #ddd;
}

th {
  background-color: #f2f2f2;
}

tr:hover {
  background-color: #f5f5f5;
}

h3:hover {
  text-decoration: underline;
}
```

[ ]:
```
###clinic.sql
DROP TABLE IF EXISTS client CASCADE;
DROP TABLE IF EXISTS phone_number_client CASCADE;
DROP TABLE IF EXISTS employee CASCADE;
DROP TABLE IF EXISTS phone_number_employee CASCADE;
DROP TABLE IF EXISTS receptionist CASCADE;
DROP TABLE IF EXISTS nurse CASCADE;
DROP TABLE IF EXISTS doctor CASCADE;
DROP TABLE IF EXISTS permanent_doctor CASCADE;
DROP TABLE IF EXISTS trainee_doctor CASCADE;
DROP TABLE IF EXISTS supervision_report CASCADE;
DROP TABLE IF EXISTS appointment CASCADE;
DROP TABLE IF EXISTS consultation CASCADE;
DROP TABLE IF EXISTS consultation_assistant CASCADE;
DROP TABLE IF EXISTS diagnostic_code CASCADE;
DROP TABLE IF EXISTS diagnostic_code_relation CASCADE;
DROP TABLE IF EXISTS consultation_diagnostic CASCADE;
DROP TABLE IF EXISTS medication CASCADE;
DROP TABLE IF EXISTS prescription CASCADE;
DROP TABLE IF EXISTS procedure_ CASCADE;
DROP TABLE IF EXISTS procedure_in_consultation CASCADE;
DROP TABLE IF EXISTS teeth CASCADE;
DROP TABLE IF EXISTS procedure_charting CASCADE;
DROP TABLE IF EXISTS procedure_imaging CASCADE;

----------------------------------------
-- Table Creation
----------------------------------------
CREATE TABLE client (
  VAT_ID VARCHAR(20),
  name VARCHAR(80),
  birth_date DATE,
  street VARCHAR(255),
  city VARCHAR(30),
  zip_code VARCHAR(12),
  gender CHAR(1),
  PRIMARY KEY (VAT_ID),
  CHECK (gender in ('M', 'F'))
);

CREATE TABLE phone_number_client (
  VAT_ID VARCHAR(20),
  phone VARCHAR(15),
  PRIMARY KEY (VAT_ID, phone),
  FOREIGN KEY(VAT_ID) REFERENCES client(VAT_ID)
);

CREATE TABLE employee (
  VAT_ID VARCHAR(20),
  name VARCHAR(80),
  birth_date DATE,
  street VARCHAR(255),
  city VARCHAR(30),
  zip_code VARCHAR(12),
  IBAN VARCHAR(30) NOT NULL,
  salary NUMERIC(12,4),
  UNIQUE(IBAN),
  PRIMARY KEY (VAT_ID),
  CHECK (salary > 0)

  -- Every employee must exist either in the table 'receptionist' or in the␣
 ↪table 'nurse' or in the table 'doctor'
  -- No employee can exist at the same time in the both the table␣
 ↪'receptionist' or in the table 'nurse' or in the table 'doctor'
);

CREATE TABLE phone_number_employee (
  VAT_ID VARCHAR(20),
  phone VARCHAR(15),
  PRIMARY KEY (VAT_ID, phone),
  FOREIGN KEY(VAT_ID) REFERENCES employee(VAT_ID)
);

CREATE TABLE receptionist (
  VAT_ID VARCHAR(20),
  PRIMARY KEY (VAT_ID),
  FOREIGN KEY(VAT_ID) REFERENCES employee(VAT_ID)
);

CREATE TABLE nurse (
  VAT_ID VARCHAR(20),
  PRIMARY KEY (VAT_ID),
  FOREIGN KEY(VAT_ID) REFERENCES employee(VAT_ID)
);

CREATE TABLE doctor (
  VAT_ID VARCHAR(20),
  specialization VARCHAR(254),
  biography VARCHAR(512),
  email_address VARCHAR(254) NOT NULL,
  UNIQUE(email_address),
  PRIMARY KEY (VAT_ID),
  FOREIGN KEY(VAT_ID) REFERENCES employee(VAT_ID)

  -- Every doctor must exist either in the table 'Trainee' or in the table␣
 ↪'Permanent'
  -- No doctor can exist at the same time in the both the table 'Trainee' or in␣
 ↪the table 'Permanent'
);

CREATE TABLE permanent_doctor (
  VAT_ID VARCHAR(20),
  years DATE,
  PRIMARY KEY (VAT_ID),
  FOREIGN KEY(VAT_ID) REFERENCES doctor(VAT_ID)
);

CREATE TABLE trainee_doctor (
  VAT_ID VARCHAR(20),
  supervisor VARCHAR(20) NOT NULL,
  PRIMARY KEY (VAT_ID),
  FOREIGN KEY(VAT_ID) REFERENCES doctor(VAT_ID),
  FOREIGN KEY(supervisor) REFERENCES permanent_doctor(VAT_ID)
);

CREATE TABLE supervision_report(
  VAT_ID VARCHAR(20),
  date_timestamp TIMESTAMP,
  description VARCHAR(512),
  evaluation NUMERIC(3,1),
  PRIMARY KEY(VAT_ID, date_timestamp),
  FOREIGN KEY(VAT_ID) REFERENCES trainee_doctor(VAT_ID),
  CHECK (evaluation >= 1 AND evaluation <= 5)
);

CREATE TABLE appointment (
  VAT_doctor VARCHAR(20),
  date_timestamp TIMESTAMP,
  VAT_client VARCHAR(20),
  description VARCHAR(512),
  PRIMARY KEY (VAT_doctor, date_timestamp),
  FOREIGN KEY(VAT_doctor) REFERENCES doctor(VAT_ID),
  FOREIGN KEY(VAT_client) REFERENCES client(VAT_ID)
);

CREATE TABLE consultation(
  VAT_doctor VARCHAR(20),
  date_timestamp TIMESTAMP,
  SOAP_S TEXT,
  SOAP_O TEXT,
  SOAP_A TEXT,
  SOAP_P TEXT,
  PRIMARY KEY (VAT_doctor, date_timestamp),
  FOREIGN KEY(VAT_doctor, date_timestamp) REFERENCES appointment(VAT_doctor, 
  ↪date_timestamp)

  -- consultations are always assigned to at least one assistant nurse
);
```

```sql
CREATE TABLE consultation_assistant(
  VAT_doctor VARCHAR(20),
  date_timestamp TIMESTAMP,
  VAT_nurse VARCHAR(20),
  PRIMARY KEY (VAT_doctor, date_timestamp),
  FOREIGN KEY(VAT_doctor, date_timestamp) REFERENCES appointment(VAT_doctor,␣
 ↪date_timestamp),
  FOREIGN KEY(VAT_nurse) REFERENCES nurse(VAT_ID)
);

CREATE TABLE diagnostic_code(
  ID INTEGER,
  description VARCHAR(255),
  PRIMARY KEY (ID)
);

CREATE TABLE diagnostic_code_relation(
  ID1 INTEGER,
  ID2 INTEGER,
  type VARCHAR(255),
  PRIMARY KEY (ID1, ID2),
  FOREIGN KEY(ID1) REFERENCES diagnostic_code(ID),
  FOREIGN KEY(ID2) REFERENCES diagnostic_code(ID)
);

CREATE TABLE consultation_diagnostic(
  VAT_doctor VARCHAR(20),
  date_timestamp TIMESTAMP,
  ID INTEGER,
  UNIQUE(VAT_doctor, date_timestamp, ID),
  PRIMARY KEY(VAT_doctor, date_timestamp),
  FOREIGN KEY (VAT_doctor, date_timestamp) REFERENCES consultation(VAT_doctor,␣
 ↪date_timestamp),
  FOREIGN KEY (ID) REFERENCES diagnostic_code(ID)
);

CREATE TABLE medication(
  name VARCHAR(64),
  lab VARCHAR(255),
  PRIMARY KEY(name, lab)
);

CREATE TABLE prescription(
  VAT_doctor VARCHAR(20),
  date_timestamp TIMESTAMP,
  ID INTEGER,
  name VARCHAR(64),
  lab VARCHAR(255),
  dosage VARCHAR(20),
  description VARCHAR(512),
  PRIMARY KEY(VAT_doctor, date_timestamp, ID, name, lab),
  FOREIGN KEY (VAT_doctor, date_timestamp, ID) REFERENCES␣
 ↪consultation_diagnostic(VAT_doctor, date_timestamp, ID),
  FOREIGN KEY (name, lab) REFERENCES medication(name, lab)
);

CREATE TABLE procedure_(
  name VARCHAR(255),
  type VARCHAR(255),
  PRIMARY KEY(name)
);

CREATE TABLE procedure_in_consultation(
  name VARCHAR(255),
  VAT_doctor VARCHAR(20),
  date_timestamp TIMESTAMP,
  description VARCHAR(512),
  PRIMARY KEY(name, VAT_doctor, date_timestamp),
  FOREIGN KEY(name) REFERENCES procedure_(name),
  FOREIGN KEY(VAT_doctor, date_timestamp) REFERENCES consultation(VAT_doctor,␣
 ↪date_timestamp)
);

CREATE TABLE teeth(
  quadrant VARCHAR(64),
  number CHAR(1),
  name VARCHAR(255),
  PRIMARY KEY(quadrant, number)
);

CREATE TABLE procedure_charting(
  name VARCHAR(255),
  VAT_doctor VARCHAR(20),
  date_timestamp TIMESTAMP,
  quadrant VARCHAR(64),
  number CHAR(1),
  desc_ VARCHAR(512),
  measure NUMERIC(6,3),
  PRIMARY KEY(name, VAT_doctor, date_timestamp, quadrant, number),
  FOREIGN KEY(name, VAT_doctor, date_timestamp) REFERENCES␣
 ↪procedure_in_consultation(name, VAT_doctor, date_timestamp),
  FOREIGN KEY(quadrant, number) REFERENCES teeth(quadrant, number)
);
```

```
CREATE TABLE procedure_imaging(
  name VARCHAR(255),
  VAT_doctor VARCHAR(20),
  date_timestamp TIMESTAMP,
  file VARCHAR(512),
  PRIMARY KEY(name, VAT_doctor, date_timestamp, file),
  FOREIGN KEY(name, VAT_doctor, date_timestamp) REFERENCES␣
 ↪procedure_in_consultation(name, VAT_doctor, date_timestamp)
);

----------------------------------------
-- Populate Relations
----------------------------------------

-- Insert into client
INSERT INTO client VALUES('123456789', 'John Doe', '1955-03-12', 'Rua Principal␣
 ↪16', 'Lisbon', '1000-000', 'M');
INSERT INTO client VALUES('983654320', 'Laura Appleton', '1960-06-25', 'Avenida␣
 ↪Liberdade 25', 'Lisbon', '1030-500', 'F');
INSERT INTO client VALUES('223456789', 'Mia White', '1972-01-30', 'Travessa dos␣
 ↪Salgueiros 35', 'Sintra', '2710-000', 'F');
INSERT INTO client VALUES('323456789', 'Noah Black', '1980-11-15', 'Largo das␣
 ↪Oliveiras 45', 'Oeiras', '2780-000', 'M');
INSERT INTO client VALUES('423456789', 'Sophia Hill', '1996-08-09', 'Rua dos␣
 ↪Cedros 37', 'Coimbra', '3000-000', 'F');
INSERT INTO client VALUES('523456789', 'Liam Pine', '1988-09-23', 'Rua do␣
 ↪Carvalho 15', 'Faro', '8000-000', 'M');
INSERT INTO client VALUES('823456789', 'Ivy Rose', '2003-02-17', 'Rua dos␣
 ↪Pinheiros 46', 'Funchal', '9000-000', 'F');
INSERT INTO client VALUES('923456789', 'Jack Frost', '1950-12-05', 'Estrada do␣
 ↪Carvalhal 32', 'Viseu', '3500-000', 'M');

INSERT INTO client VALUES('624456789', 'Alice Waters', '1983-07-07', 'Rua do␣
 ↪Rio 3', 'Guimarães', '4800-000', 'F');
INSERT INTO client VALUES('724456789', 'Maxwell Bright', '1992-12-14', 'Rua da␣
 ↪Encosta 5', 'Santarém', '2000-000', 'M');
INSERT INTO client VALUES('824456789', 'Iris Bloom', '1979-04-22', 'Rua do␣
 ↪Jardim 8', 'Bragança', '5300-000', 'F');


-- Insert into phone_number_client
INSERT INTO phone_number_client VALUES('123456789', '912345678');
INSERT INTO phone_number_client VALUES('223456789', '913345678');
INSERT INTO phone_number_client VALUES('323456789', '914456789');
INSERT INTO phone_number_client VALUES('423456789', '915678901');
INSERT INTO phone_number_client VALUES('523456789', '916789012');
INSERT INTO phone_number_client VALUES('823456789', '919876543');
INSERT INTO phone_number_client VALUES('923456789', '920987654');

INSERT INTO phone_number_client VALUES('624456789', '917654321');
INSERT INTO phone_number_client VALUES('724456789', '918765432');
INSERT INTO phone_number_client VALUES('824456789', '919876543');

-- Insert into employee
INSERT INTO employee VALUES('987654321', 'Robin Smith', '1980-05-15', 'Rua das␣
 ↪Ameixas 45', 'Lisboa', '4000-123', '0002 0123 1234 5678 9015 4', 900.00);
INSERT INTO employee VALUES('987654320', 'Jane Sweettooth', '1968-04-16',␣
 ↪'Avenida da Liberdade 36', 'Lisboa', '1050-153', '0002 0223 1234 5678 9015␣
 ↪4', 3000.00);
INSERT INTO employee VALUES('987654319', 'Jonh Red', '1991-10-26', 'Largo das␣
 ↪Camélias 6', 'Lisboa', '1100-026', '0002 0123 1234 5678 1015 4', 1200.00);
INSERT INTO employee VALUES('983654320', 'Bob Sweden', '1991-02-17', 'Rua dos␣
 ↪Cravos 82', 'Lisboa', '1150-103', '0002 0223 1244 5678 9015 4', 1600.00);
INSERT INTO employee VALUES('876543210', 'Emma Green', '1985-07-20', 'Praceta␣
 ↪das Rosas 1', 'Lisboa', '4001-123', '0002 0333 1234 5678 9015 4', 1200.00);
INSERT INTO employee VALUES('876543211', 'Olivia Brown', '1990-08-22', 'Rua das␣
 ↪Margaridas 4', 'Lisboa', '1051-153', '0002 0444 1234 5678 9015 4', 1800.00);
INSERT INTO employee VALUES('876543212', 'Sarah Clark', '1980-12-05', 'Avenida␣
 ↪das Tulipas 22', 'Lisboa', '4050-000', '0002 0555 1234 5678 9015 4', 3200.
 ↪00);
INSERT INTO employee VALUES('876543213', 'Luke Grayson', '1978-07-19', 'Rua␣
 ↪Azul 33', 'Lisbon', '1100-000', '0002 0666 1234 5678 9015 4', 3400.00);
INSERT INTO employee VALUES('876543214', 'Ethan Storm', '1989-08-15', 'Rua do␣
 ↪Vento 45', 'Setúbal', '4700-000', '0002 0777 1234 5678 9015 4', 1300.00);
INSERT INTO employee VALUES('876543215', 'Chloe Brooks', '1993-05-20', 'Rua do␣
 ↪Sol 67', 'Setúbal', '3810-000', '0002 0888 1234 5678 9015 4', 1350.00);
INSERT INTO employee VALUES('876543216', 'Ava Taylor', '1995-10-11', 'Avenida␣
 ↪da Chuva 12', 'Setúbal', '2900-000', '0002 0999 1234 5678 9015 4', 1280.00);
INSERT INTO employee VALUES('876543217', 'Ryan Leaf', '1976-03-30', 'Rua das␣
 ↪Flores 90', 'Setúbal', '8001-000', '0002 1010 1234 5678 9015 4', 950.00);


-- Insert into phone_number_employee
INSERT INTO phone_number_employee VALUES('987654321', '931234567');
INSERT INTO phone_number_employee VALUES('987654320', '931254567');
INSERT INTO phone_number_employee VALUES('987654319', '931214567');
INSERT INTO phone_number_employee VALUES('983654320', '911214567');
INSERT INTO phone_number_employee VALUES('876543210', '932334567');
INSERT INTO phone_number_employee VALUES('876543211', '932445678');
INSERT INTO phone_number_employee VALUES('876543212', '933556677');
INSERT INTO phone_number_employee VALUES('876543213', '944667788');
INSERT INTO phone_number_employee VALUES('876543214', '935556677');
INSERT INTO phone_number_employee VALUES('876543215', '936667788');
INSERT INTO phone_number_employee VALUES('876543216', '937778899');
INSERT INTO phone_number_employee VALUES('876543217', '938889910');


-- Insert into receptionist
-- Note: A receptionist is also an employee, so make sure the VAT_ID exists in␣
 ↪the employee table.
INSERT INTO receptionist VALUES('987654321');
INSERT INTO receptionist VALUES('876543217');

-- Insert into nurse
-- Note: A nurse is also an employee, so make sure the VAT_ID exists in the␣
 ↪employee table.
-- Example with a different employee for the nurse role
INSERT INTO nurse VALUES('987654319');
INSERT INTO nurse VALUES('876543214');
INSERT INTO nurse VALUES('876543215');
INSERT INTO nurse VALUES('876543216');

-- Insert into doctor
INSERT INTO doctor VALUES('987654320', 'Dentistry', 'Experienced dentist',␣
 ↪'jane.sweettooth@email.com');
INSERT INTO doctor VALUES('983654320', 'Dentistry', 'Trainee dentist', 'bob.
 ↪sweden@email.com');
INSERT INTO doctor VALUES('876543210', 'General Medicine', 'Experienced GP',␣
 ↪'emma.green@email.com');
INSERT INTO doctor VALUES('876543211', 'Pediatrics', 'Pediatric specialist',␣
 ↪'olivia.brown@email.com');
INSERT INTO doctor VALUES('876543212', 'Orthodontics', 'Orthodontic␣
 ↪specialist', 'sarah.ortho@email.com');
INSERT INTO doctor VALUES('876543213', 'Oral Surgery', 'Experienced oral␣
 ↪surgeon', 'luke.surgeon@email.com');


-- Insert into permanent_doctor
INSERT INTO permanent_doctor VALUES('987654320', '2000-01-10');
INSERT INTO permanent_doctor VALUES('876543210', '2010-02-15');
INSERT INTO permanent_doctor VALUES('876543212', '2008-08-20');
INSERT INTO permanent_doctor VALUES('876543213', '2010-10-25');

-- Insert into trainee_doctor
-- Note: Ensure the supervisor exists in the permanent_doctor table.
INSERT INTO trainee_doctor VALUES('983654320', '876543210');  -- Bob Sweden␣
 ↪supervised by Emma Green
```

```
INSERT INTO trainee_doctor VALUES('876543211', '987654320');  -- Olivia Brown␣
 ↪supervised by Jane Sweettooth

-- Insert into supervision_report
INSERT INTO supervision_report VALUES('983654320', '2023-12-10 09:00:00', 'Good␣
 ↪progress in training.', 4.0);
INSERT INTO supervision_report VALUES('983654320', '2023-11-05 14:00:00',␣
 ↪'Needs improvement in patient communication.', 2.5);
INSERT INTO supervision_report VALUES('983654320', '2023-10-20 10:00:00',␣
 ↪'Insufficient knowledge in dental procedures.', 2.0);
INSERT INTO supervision_report VALUES('876543211', '2023-09-15 11:30:00',␣
 ↪'Excellent in pediatric care.', 4.5);



-- Insert into appointment
INSERT INTO appointment VALUES('876543213', '2019-06-10 11:00:00', '923456789',␣
 ↪'Oral surgery consultation');
INSERT INTO appointment VALUES('876543212', '2019-06-20 15:00:00', '823456789',␣
 ↪'Orthodontic check-up');
INSERT INTO appointment VALUES('983654320', '2022-11-01 14:00:00', '223456789',␣
 ↪'Routine dental check-up');
INSERT INTO appointment VALUES('987654320', '2023-01-10 10:00:00', '123456789',␣
 ↪'Routine dental check-up');
INSERT INTO appointment VALUES('987654320', '2023-01-11 11:00:00', '123456789',␣
 ↪'Pain management consultation');
INSERT INTO appointment VALUES('987654320', '2023-01-12 09:30:00', '223456789',␣
 ↪'Regular dental check-up');
INSERT INTO appointment VALUES('987654320', '2023-01-12 12:00:00', '123456789',␣
 ↪'Follow-up dental check-up');
INSERT INTO appointment VALUES('987654320', '2023-01-15 10:00:00', '323456789',␣
 ↪'Regular dental check-up');
INSERT INTO appointment VALUES('987654320', '2023-01-16 11:00:00', '423456789',␣
 ↪'Dental check-up');
INSERT INTO appointment VALUES('987654320', '2023-01-17 12:00:00', '523456789',␣
 ↪'Dental check-up');
INSERT INTO appointment VALUES('876543212', '2023-01-20 14:30:00', '323456789',␣
 ↪'Cavity check-up');
INSERT INTO appointment VALUES('876543213', '2023-01-21 16:00:00', '423456789',␣
 ↪'Infection check-up');

INSERT INTO appointment (VAT_doctor, date_timestamp, VAT_client, description)␣
 ↪VALUES
('987654320', '2019-01-02 15:00:00', '123456789', 'Routine dental check-up'),
('987654320', '2019-01-04 15:00:00', '983654320', 'Dental cleaning'),
('987654320', '2019-01-06 15:00:00', '223456789', 'Cavity consultation'),
('987654320', '2019-01-08 15:00:00', '323456789', 'Tooth extraction␣
 ↪consultation'),
('987654320', '2019-01-10 15:00:00', '423456789', 'Routine dental check-up'),
('987654320', '2019-01-12 15:00:00', '523456789', 'Dental cleaning'),
('987654320', '2019-01-14 15:00:00', '823456789', 'Cavity filling'),
('987654320', '2019-01-16 15:00:00', '923456789', 'Teeth whitening␣
 ↪consultation'),
('987654320', '2019-01-18 15:00:00', '123456789', 'Dental implant␣
 ↪consultation'),
('987654320', '2019-01-20 15:00:00', '983654320', 'Routine dental check-up'),
('987654320', '2019-02-02 15:00:00', '123456789', 'Routine dental check-up'),
('987654320', '2019-02-04 15:00:00', '983654320', 'Dental cleaning'),
('987654320', '2019-02-06 15:00:00', '223456789', 'Cavity consultation'),
('987654320', '2019-02-08 15:00:00', '323456789', 'Tooth extraction␣
 ↪consultation'),
('987654320', '2019-02-10 15:00:00', '423456789', 'Routine dental check-up'),
('987654320', '2019-02-12 15:00:00', '523456789', 'Dental cleaning'),
('987654320', '2019-02-14 15:00:00', '823456789', 'Cavity filling'),
('987654320', '2019-02-16 15:00:00', '923456789', 'Teeth whitening␣
 ↪consultation'),
('987654320', '2019-02-18 15:00:00', '123456789', 'Dental implant␣
 ↪consultation'),
('987654320', '2019-02-20 15:00:00', '983654320', 'Routine dental check-up'),
('987654320', '2019-03-02 15:00:00', '123456789', 'Routine dental check-up'),
('987654320', '2019-03-04 15:00:00', '983654320', 'Dental cleaning'),
('987654320', '2019-03-06 15:00:00', '223456789', 'Cavity consultation'),
('987654320', '2019-03-08 15:00:00', '323456789', 'Tooth extraction␣
 ↪consultation'),
('987654320', '2019-03-10 15:00:00', '423456789', 'Routine dental check-up'),
('987654320', '2019-03-12 15:00:00', '523456789', 'Dental cleaning'),
('987654320', '2019-03-14 15:00:00', '823456789', 'Cavity filling'),
('987654320', '2019-03-16 15:00:00', '923456789', 'Teeth whitening␣
 ↪consultation'),
('987654320', '2019-03-18 15:00:00', '123456789', 'Dental implant␣
 ↪consultation'),
('987654320', '2019-03-20 15:00:00', '983654320', 'Routine dental check-up'),
('987654320', '2019-04-02 15:00:00', '123456789', 'Routine dental check-up'),
('987654320', '2019-04-04 15:00:00', '983654320', 'Dental cleaning'),
('987654320', '2019-04-06 15:00:00', '223456789', 'Cavity consultation'),
('987654320', '2019-04-08 15:00:00', '323456789', 'Tooth extraction␣
 ↪consultation'),
('987654320', '2019-04-10 15:00:00', '423456789', 'Routine dental check-up'),
('987654320', '2019-04-12 15:00:00', '523456789', 'Dental cleaning'),
('987654320', '2019-04-14 15:00:00', '823456789', 'Cavity filling'),
('987654320', '2019-04-16 15:00:00', '923456789', 'Teeth whitening␣
 ↪consultation'),
('987654320', '2019-04-18 15:00:00', '123456789', 'Dental implant␣
 ↪consultation'),
('987654320', '2019-04-20 15:00:00', '983654320', 'Routine dental check-up'),
('987654320', '2019-05-02 15:00:00', '123456789', 'Routine dental check-up'),
('987654320', '2019-05-04 15:00:00', '983654320', 'Dental cleaning'),
('987654320', '2019-05-06 15:00:00', '223456789', 'Cavity consultation'),
('987654320', '2019-05-08 15:00:00', '323456789', 'Tooth extraction␣
 ↪consultation'),
('987654320', '2019-05-10 15:00:00', '423456789', 'Routine dental check-up'),
('987654320', '2019-05-12 15:00:00', '523456789', 'Dental cleaning'),
('987654320', '2019-05-14 15:00:00', '823456789', 'Cavity filling'),
('987654320', '2019-05-16 15:00:00', '923456789', 'Teeth whitening␣
 ↪consultation'),
('987654320', '2019-05-18 15:00:00', '123456789', 'Dental implant␣
 ↪consultation'),
('987654320', '2019-05-20 15:00:00', '983654320', 'Routine dental check-up'),
('987654320', '2019-06-02 15:00:00', '123456789', 'Routine dental check-up'),
('987654320', '2019-06-04 15:00:00', '983654320', 'Dental cleaning'),
('987654320', '2019-06-06 15:00:00', '223456789', 'Cavity consultation'),
('987654320', '2019-06-08 15:00:00', '323456789', 'Tooth extraction␣
 ↪consultation'),
('987654320', '2019-06-10 15:00:00', '423456789', 'Routine dental check-up'),
('987654320', '2019-06-12 15:00:00', '523456789', 'Dental cleaning'),
('987654320', '2019-06-14 15:00:00', '823456789', 'Cavity filling'),
('987654320', '2019-06-16 15:00:00', '923456789', 'Teeth whitening␣
 ↪consultation'),
('987654320', '2019-06-18 15:00:00', '123456789', 'Dental implant␣
 ↪consultation'),
('987654320', '2019-06-20 15:00:00', '983654320', 'Routine dental check-up'),
('987654320', '2019-07-02 15:00:00', '123456789', 'Routine dental check-up'),
('987654320', '2019-07-04 15:00:00', '983654320', 'Dental cleaning'),
('987654320', '2019-07-06 15:00:00', '223456789', 'Cavity consultation'),
('987654320', '2019-07-08 15:00:00', '323456789', 'Tooth extraction␣
 ↪consultation'),
('987654320', '2019-07-10 15:00:00', '423456789', 'Routine dental check-up'),
('987654320', '2019-07-12 15:00:00', '523456789', 'Dental cleaning'),
('987654320', '2019-07-14 15:00:00', '823456789', 'Cavity filling'),
('987654320', '2019-07-16 15:00:00', '923456789', 'Teeth whitening␣
 ↪consultation'),
('987654320', '2019-07-18 15:00:00', '123456789', 'Dental implant␣
 ↪consultation'),
('987654320', '2019-07-20 15:00:00', '983654320', 'Routine dental check-up'),
('987654320', '2019-08-02 15:00:00', '123456789', 'Routine dental check-up'),
('987654320', '2019-08-04 15:00:00', '983654320', 'Dental cleaning'),
('987654320', '2019-08-06 15:00:00', '223456789', 'Cavity consultation'),
('987654320', '2019-08-08 15:00:00', '323456789', 'Tooth extraction␣
 ↪consultation'),
('987654320', '2019-08-10 15:00:00', '423456789', 'Routine dental check-up'),
('987654320', '2019-08-12 15:00:00', '523456789', 'Dental cleaning'),
('987654320', '2019-08-14 15:00:00', '823456789', 'Cavity filling'),
('987654320', '2019-08-16 15:00:00', '923456789', 'Teeth whitening␣
 ↪consultation'),
('987654320', '2019-08-18 15:00:00', '123456789', 'Dental implant␣
 ↪consultation'),
('987654320', '2019-08-20 15:00:00', '983654320', 'Routine dental check-up'),
('987654320', '2019-10-02 15:00:00', '123456789', 'Routine dental check-up'),
('987654320', '2019-10-04 15:00:00', '983654320', 'Dental cleaning'),
('987654320', '2019-10-06 15:00:00', '223456789', 'Cavity consultation'),
('987654320', '2019-10-08 15:00:00', '323456789', 'Tooth extraction␣
 ↪consultation'),
('987654320', '2019-10-10 15:00:00', '423456789', 'Routine dental check-up'),
('987654320', '2019-10-12 15:00:00', '523456789', 'Dental cleaning'),
('987654320', '2019-10-14 15:00:00', '823456789', 'Cavity filling'),
('987654320', '2019-10-16 15:00:00', '923456789', 'Teeth whitening␣
 ↪consultation'),
('987654320', '2019-10-18 15:00:00', '123456789', 'Dental implant␣
 ↪consultation'),
('987654320', '2019-10-20 15:00:00', '983654320', 'Routine dental check-up'),
('987654320', '2019-11-02 15:00:00', '123456789', 'Routine dental check-up'),
('987654320', '2019-11-04 15:00:00', '983654320', 'Dental cleaning'),
('987654320', '2019-11-06 15:00:00', '223456789', 'Cavity consultation'),
('987654320', '2019-11-08 15:00:00', '624456789', 'Tooth extraction␣
 ↪consultation'),
('987654320', '2019-11-10 15:00:00', '724456789', 'Routine dental check-up'),
('987654320', '2019-11-12 15:00:00', '523456789', 'Dental cleaning'),
('987654320', '2019-11-14 15:00:00', '823456789', 'Cavity filling'),
('987654320', '2019-11-16 15:00:00', '923456789', 'Teeth whitening␣
 ↪consultation'),
('987654320', '2019-11-18 15:00:00', '123456789', 'Dental implant␣
 ↪consultation'),
('987654320', '2019-11-20 15:00:00', '824456789', 'Routine dental check-up');


-- Insert into consultation
INSERT INTO consultation VALUES('876543213', '2019-06-10 11:00:00', 'Oral␣
 ↪examination', 'Wisdom tooth extraction needed', 'Surgery scheduled',␣
 ↪'Post-operative care discussion');
INSERT INTO consultation VALUES('876543212', '2019-06-20 15:00:00',␣
 ↪'Orthodontic check', 'Braces adjustment needed', 'Next visit in 2 months',␣
 ↪'Follow-up for brace adjustment');
```

```
INSERT INTO consultation VALUES('983654320', '2022-11-01 14:00:00', 'Patient␣
 ↪reports discomfort', 'Mild pain observed', 'Prescription of Paracetamol',␣
 ↪'Recommendation to monitor and follow-up if pain persists');
INSERT INTO consultation VALUES('987654320', '2023-01-10 10:00:00', 'Patient␣
 ↪reports toothache', 'Cavity in molar tooth', 'Cavity filling required',␣
 ↪'Schedule follow-up after procedure');
INSERT INTO consultation VALUES('987654320', '2023-01-11 11:00:00', 'Discussion␣
 ↪about pain', 'Pain management strategies', 'Prescription of Ibuprofen',␣
 ↪'Follow-up if pain persists');
INSERT INTO consultation VALUES('987654320', '2023-01-12 09:30:00', 'Regular␣
 ↪check-up', 'Signs of gingivitis observed', 'Advice on better dental␣
 ↪hygiene', 'Schedule follow-up in 3 months');
INSERT INTO consultation VALUES('987654320', '2023-01-12 12:00:00', 'Follow-up␣
 ↪check-up notes', 'Maintaining oral health', 'Continue current hygiene␣
 ↪practices', 'Next check-up in 6 months');
INSERT INTO consultation VALUES('987654320', '2023-01-15 10:00:00', 'Regular␣
 ↪check-up', 'Early signs of periodontitis', 'Recommendation for treatment',␣
 ↪'Immediate treatment required');
INSERT INTO consultation VALUES('876543212', '2023-01-20 14:30:00', 'Cavity␣
 ↪check', 'Cavity detected in molar tooth', 'Treatment plan discussed',␣
 ↪'Follow-up in 3 weeks');
INSERT INTO consultation VALUES('876543213', '2023-01-21 16:00:00', 'Infection␣
 ↪check', 'Patient reports symptoms of infection', 'Prescription provided',␣
 ↪'Follow-up in 1 week');

INSERT INTO consultation VALUES('987654320', '2019-11-08 15:00:00', 'Patient␣
 ↪reports discomfort', 'Mild gum recession observed, potential signs of␣
 ↪gingivitis', 'Extraction needed', 'Schedule for extraction');
INSERT INTO consultation VALUES('987654320', '2019-11-10 15:00:00', 'Regular␣
 ↪check-up', 'Signs of gum inflammation, suggestive of gingivitis', 'Good oral␣
 ↪health', 'Continue current hygiene practices');
INSERT INTO consultation VALUES('987654320', '2019-11-20 15:00:00', 'Routine␣
 ↪check-up', 'Early signs of gingival recession, indicative of gingivitis',␣
 ↪'Maintain hygiene', 'Next check-up in 6 months');

-- Insert into consultation_assistant
-- Ensure that there is always on assistant per consultation
INSERT INTO consultation_assistant VALUES('876543213', '2019-06-10 11:00:00',␣
 ↪'987654319');
INSERT INTO consultation_assistant VALUES('876543212', '2019-06-20 15:00:00',␣
 ↪'876543214');
INSERT INTO consultation_assistant VALUES('983654320', '2022-11-01 14:00:00',␣
 ↪'876543215');
INSERT INTO consultation_assistant VALUES('987654320', '2023-01-10 10:00:00',␣
 ↪'987654319');
```

```
INSERT INTO consultation_assistant VALUES('987654320', '2023-01-11 11:00:00',␣
 ↪'876543216');
INSERT INTO consultation_assistant VALUES('987654320', '2023-01-12 09:30:00',␣
 ↪'876543214');
INSERT INTO consultation_assistant VALUES('987654320', '2023-01-12 12:00:00',␣
 ↪'876543215');
INSERT INTO consultation_assistant VALUES('987654320', '2023-01-15 10:00:00',␣
 ↪'876543216');
INSERT INTO consultation_assistant VALUES('876543212', '2023-01-20 14:30:00',␣
 ↪'876543214');
INSERT INTO consultation_assistant VALUES('876543213', '2023-01-21 16:00:00',␣
 ↪'876543215');
INSERT INTO consultation_assistant VALUES('987654320', '2019-11-08 15:00:00',␣
 ↪'987654319');
INSERT INTO consultation_assistant VALUES('987654320', '2019-11-10 15:00:00',␣
 ↪'987654319');
INSERT INTO consultation_assistant VALUES('987654320', '2019-11-20 15:00:00',␣
 ↪'987654319');


-- Insert into diagnostic_code
INSERT INTO diagnostic_code VALUES(1, 'Cavity');
INSERT INTO diagnostic_code VALUES(2, 'Infection');
INSERT INTO diagnostic_code VALUES(3, 'Toothache');
INSERT INTO diagnostic_code VALUES(4, 'Bruxism');
INSERT INTO diagnostic_code VALUES(5, 'Gingivitis');


-- Insert into diagnostic_code_relation
INSERT INTO diagnostic_code_relation VALUES(1, 2, 'Related');
INSERT INTO diagnostic_code_relation VALUES(1, 3, 'Related');
INSERT INTO diagnostic_code_relation VALUES(2, 3, 'Related');

-- Insert into consultation_diagnostic
INSERT INTO consultation_diagnostic VALUES('876543212', '2019-06-20 15:00:00',␣
 ↪1);
INSERT INTO consultation_diagnostic VALUES('876543213', '2019-06-10 11:00:00',␣
 ↪2);
INSERT INTO consultation_diagnostic VALUES('983654320', '2022-11-01 14:00:00',␣
 ↪3);

INSERT INTO consultation_diagnostic VALUES('987654320', '2023-01-10 10:00:00',␣
 ↪1);
INSERT INTO consultation_diagnostic VALUES('987654320', '2023-01-11 11:00:00',␣
 ↪4);
```

```
INSERT INTO consultation_diagnostic VALUES('987654320', '2023-01-12 09:30:00',␣
 ↪1);
INSERT INTO consultation_diagnostic VALUES('987654320', '2023-01-12 12:00:00',␣
 ↪2);
INSERT INTO consultation_diagnostic VALUES('987654320', '2023-01-15 10:00:00',␣
 ↪2);
INSERT INTO consultation_diagnostic VALUES('876543212', '2023-01-20 14:30:00',␣
 ↪1);
INSERT INTO consultation_diagnostic VALUES('876543213', '2023-01-21 16:00:00',␣
 ↪2);
INSERT INTO consultation_diagnostic VALUES('987654320', '2019-11-08 15:00:00',␣
 ↪5);
INSERT INTO consultation_diagnostic VALUES('987654320', '2019-11-10 15:00:00',␣
 ↪5);
INSERT INTO consultation_diagnostic VALUES('987654320', '2019-11-20 15:00:00',␣
 ↪5);

-- Insert into medication
INSERT INTO medication VALUES('Amoxicillin', 'MedLab');
INSERT INTO medication VALUES('Ibuprofen', 'PainAwayLab');
INSERT INTO medication VALUES('Paracetamol', 'FeverGoneLab');
INSERT INTO medication VALUES('Mouthwash', 'OralHealthLab');
INSERT INTO medication VALUES('CavityMed', 'CavityLab');
INSERT INTO medication VALUES('GeneralAntibiotic', 'GeneralLab');

-- Insert into prescription
INSERT INTO prescription VALUES('876543212', '2019-06-20 15:00:00', 1,␣
 ↪'CavityMed', 'CavityLab', '1 pill', 'Daily for cavity treatment');
INSERT INTO prescription VALUES('876543213', '2019-06-10 11:00:00', 2,␣
 ↪'GeneralAntibiotic', 'GeneralLab', '1 pill', 'Daily for infection␣
 ↪treatment');
INSERT INTO prescription VALUES('983654320', '2022-11-01 14:00:00', 3,␣
 ↪'Paracetamol', 'FeverGoneLab', '500mg', 'Take one tablet every 8 hours');
INSERT INTO prescription VALUES('987654320', '2023-01-10 10:00:00', 1,␣
 ↪'Amoxicillin', 'MedLab', '500mg', 'Take twice daily');
INSERT INTO prescription VALUES('987654320', '2023-01-11 11:00:00', 4,␣
 ↪'Ibuprofen', 'PainAwayLab', '400mg', 'Take one tablet every 6 hours');
INSERT INTO prescription VALUES('987654320', '2023-01-12 09:30:00', 1,␣
 ↪'Amoxicillin', 'MedLab', '500mg', 'Take three times a day');
INSERT INTO prescription VALUES('987654320', '2023-01-12 12:00:00', 2,␣
 ↪'Mouthwash', 'OralHealthLab', 'Use twice daily', 'For oral hygiene');
INSERT INTO prescription VALUES('987654320', '2023-01-15 10:00:00', 2,␣
 ↪'Mouthwash', 'OralHealthLab', 'Use twice daily', 'For oral hygiene');
INSERT INTO prescription VALUES('876543212', '2023-01-20 14:30:00', 1,␣
 ↪'CavityMed', 'CavityLab', '1 pill', 'Daily for cavity treatment');
```

```
INSERT INTO prescription VALUES('876543213', '2023-01-21 16:00:00', 2,␣
↪'GeneralAntibiotic', 'GeneralLab', '1 pill', 'Daily for infection␣
↪treatment');


-- Insert into procedure_
INSERT INTO procedure_ VALUES('Tooth Filling', 'Restorative');
INSERT INTO procedure_ VALUES('Dental X-Ray', 'Diagnostic');
INSERT INTO procedure_ VALUES ('Gum Measurement', 'Diagnostic');


-- Insert into procedure_in_consultation
INSERT INTO procedure_in_consultation VALUES('Tooth Filling', '987654320',␣
↪'2023-01-10 10:00:00', 'Filling of molar tooth');
INSERT INTO procedure_in_consultation VALUES('Dental X-Ray', '987654320',␣
↪'2023-01-10 10:00:00', 'Dental X-ray imaging');
INSERT INTO Procedure_in_consultation VALUES ('Gum Measurement', '987654320',␣
↪'2019-11-08 15:00:00', 'Gum measurement for tooth 3 in the Lower Left␣
↪quadrant');
INSERT INTO Procedure_in_consultation VALUES ('Gum Measurement', '987654320',␣
↪'2019-11-10 15:00:00', 'Gum measurement for tooth 1 in the Upper Right␣
↪quadrant');
INSERT INTO Procedure_in_consultation VALUES ('Gum Measurement', '987654320',␣
↪'2019-11-20 15:00:00', 'Gum measurement for tooth 2 in the Lower Right␣
↪quadrant');

-- Insert into teeth
INSERT INTO teeth VALUES('Upper Right', '2', 'Second Molar');
INSERT INTO teeth VALUES ('Lower Left', '3', 'Third Molar');
INSERT INTO teeth VALUES ('Upper Right', '1', 'First Molar');
INSERT INTO teeth VALUES('Lower Right', '2', 'Second Molar');


-- Insert into procedure_charting
INSERT INTO procedure_charting VALUES('Tooth Filling', '987654320', '2023-01-10␣
↪10:00:00', 'Upper Right', '2', 'Filling completed', 2.0);
INSERT INTO procedure_charting VALUES('Gum Measurement', '987654320',␣
↪'2019-11-08 15:00:00', 'Lower Left', '3', 'Mild gum recession', 4.2);
INSERT INTO procedure_charting VALUES('Gum Measurement', '987654320',␣
↪'2019-11-10 15:00:00', 'Upper Right', '1', 'Signs of gum inflammation', 4.1);
INSERT INTO procedure_charting VALUES('Gum Measurement', '987654320',␣
↪'2019-11-20 15:00:00', 'Lower Right', '2', 'Early signs of gingival␣
↪recession', 4.3);

-- Insert into procedure_imaging
```

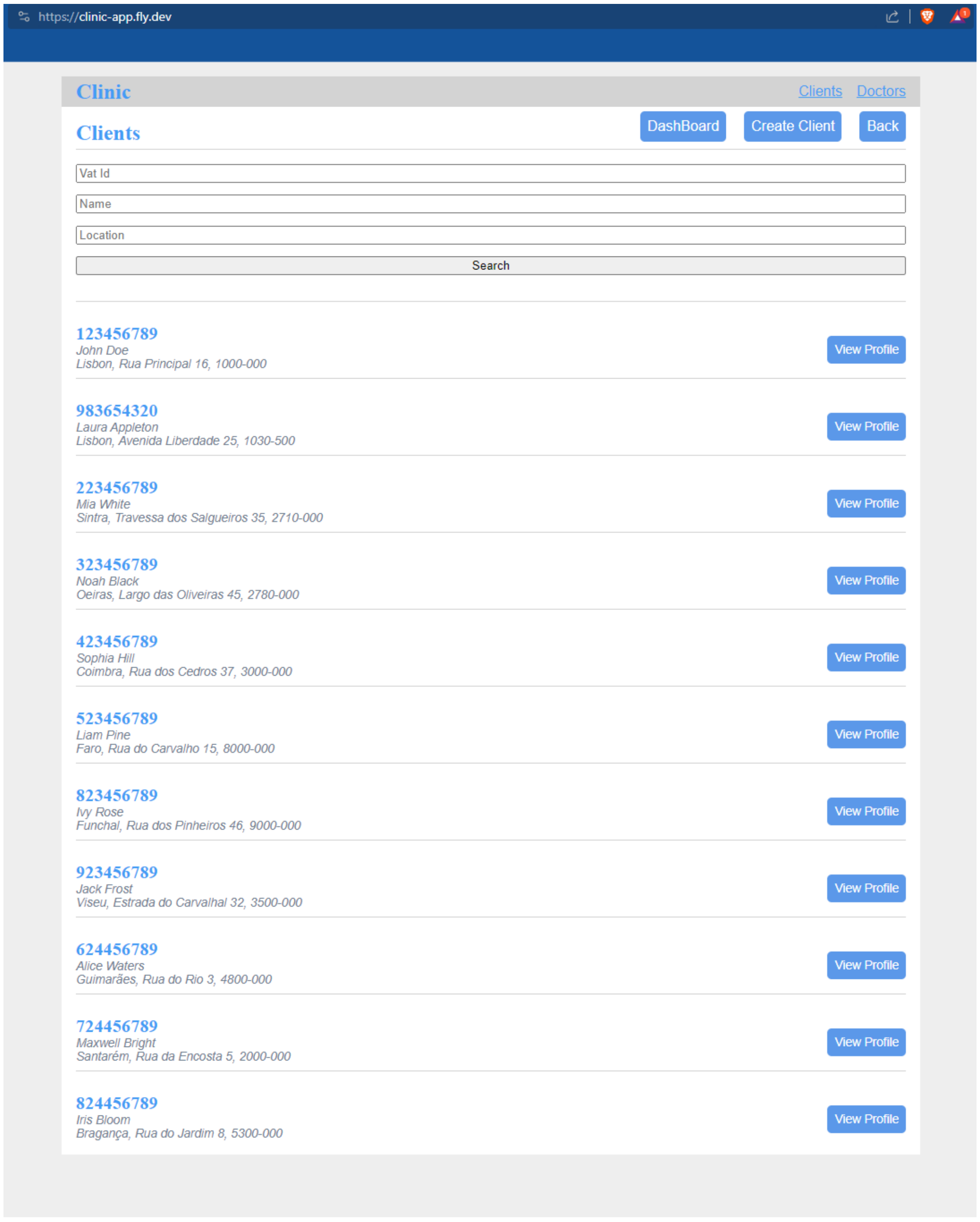
```sql
INSERT INTO procedure_imaging VALUES('Dental X-Ray', '987654320', '2023-01-10␣
 ↪10:00:00', '/path/to/dental_xray.png');

-- Views Creation

DROP VIEW IF EXISTS consultation_details;
DROP VIEW IF EXISTS facts_consultations;
DROP VIEW IF EXISTS dim_date;
DROP VIEW IF EXISTS dim_client;
DROP VIEW IF EXISTS dim_location;

CREATE VIEW dim_date(date, day_, month_, year_)
AS
SELECT date_timestamp, EXTRACT(day FROM date_timestamp), EXTRACT(month FROM␣
 ↪date_timestamp), EXTRACT(year FROM date_timestamp)
FROM consultation;

CREATE VIEW dim_client(VAT, gender, age)
AS
SELECT VAT_ID AS VAT, gender, EXTRACT(YEAR FROM AGE(CURRENT_DATE, birth_date))␣
 ↪AS age
FROM client;

CREATE VIEW dim_location(zip, city)
AS
SELECT zip_code AS zip, city
FROM client;

CREATE VIEW facts_consultations(VAT, date, zip, num_diagnostic_codes,␣
 ↪num_procedures)
AS
SELECT dc.VAT AS VAT, dd.date AS date, dl.zip AS zip, COUNT(DISTINCT cd.ID) AS␣
 ↪num_diagnostic_codes, COUNT(pc.name) AS num_procedures
FROM consultation c
JOIN appointment a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.
 ↪date_timestamp
JOIN dim_client dc ON a.VAT_client = dc.VAT
JOIN dim_date dd ON c.date_timestamp = dd.date
JOIN client cl ON cl.VAT_ID = dc.VAT
JOIN dim_location dl ON cl.zip_code = dl.zip
LEFT JOIN consultation_diagnostic cd ON c.VAT_doctor = cd.VAT_doctor AND c.
 ↪date_timestamp = cd.date_timestamp
LEFT JOIN Procedure_in_consultation pc ON c.VAT_doctor = pc.VAT_doctor AND c.
 ↪date_timestamp = pc.date_timestamp
GROUP BY dc.VAT, dd.date, dl.zip;
```

```
CREATE VIEW consultation_details(vat_client, vat_doctor, date_timestamp,␣
 ↪description, nurse_id, soap_s, soap_o, soap_a, soap_p, diag_desc, med_name,␣
 ↪lab, dosage, pr_description)
AS
SELECT ap.vat_client,
    ap.vat_doctor,
    ap.date_timestamp,
    ap.description,
    ca.vat_nurse AS nurse_id,
    co.soap_s,
    co.soap_o,
    co.soap_a,
    co.soap_p,
    d.description AS diag_desc,
    pr.name AS med_name,
    pr.lab,
    pr.dosage,
    pr.description AS pr_description
   FROM client c
     JOIN appointment ap ON c.vat_id = ap.vat_client
     LEFT JOIN consultation co ON ap.vat_doctor = co.vat_doctor AND ap.
 ↪date_timestamp = co.date_timestamp
     LEFT JOIN consultation_assistant ca ON ap.vat_doctor = ca.vat_doctor AND␣
 ↪ap.date_timestamp = ca.date_timestamp
     LEFT JOIN consultation_diagnostic cd ON ap.vat_doctor = cd.vat_doctor AND␣
 ↪ap.date_timestamp = cd.date_timestamp
     LEFT JOIN prescription pr ON pr.vat_doctor = cd.vat_doctor AND pr.
 ↪date_timestamp = cd.date_timestamp AND pr.id = cd.id
     LEFT JOIN diagnostic_code d ON pr.id = d.id;
```

1.3.1 Url of the WebApp: https://clinic-app.fly.dev/

- The index page displays all the clients in the database with a filter feature
- On the upper right of the page there's a 'doctors' redirect button that displays the doctors available on a certain date
- You get redirected to a client creation page when you touch the 'Create client' button
- When you touch the 'View profile' button it redirects to a page presenting all appointments and consultations of that client
- Inside the appointments and consultations page, you can create a new appointment, where you first insert the date and then the doctors displayed are the ones available
- Inside the appointments and consultations page, you can touch the 'View details' button to get redirected to a page presenting information about the consultation(if most fields are not filled, there should appear a button on the screen to create a consultation)
- In this menu, you can fill SOAP notes, allocate a nurse assistant, choose the diagnostic code, and fill prescriptions
- Finally the dashboard menu presents information on all the clients of the database, where it is recorded the number of diagnostic codes and procedures that were assigned to each client

per year



https://clinic-app.fly.dev

Clinic — Clients Doctors

Clients — DashBoard | Create Client | Back

Vat Id

Name

Location

Search

123456789
John Doe
Lisbon, Rua Principal 16, 1000-000
View Profile

983654320
Laura Appleton
Lisbon, Avenida Liberdade 25, 1030-500
View Profile

223456789
Mia White
Sintra, Travessa dos Salgueiros 35, 2710-000
View Profile

323456789
Noah Black
Oeiras, Largo das Oliveiras 45, 2780-000
View Profile

423456789
Sophia Hill
Coimbra, Rua dos Cedros 37, 3000-000
View Profile

523456789
Liam Pine
Faro, Rua do Carvalho 15, 8000-000
View Profile

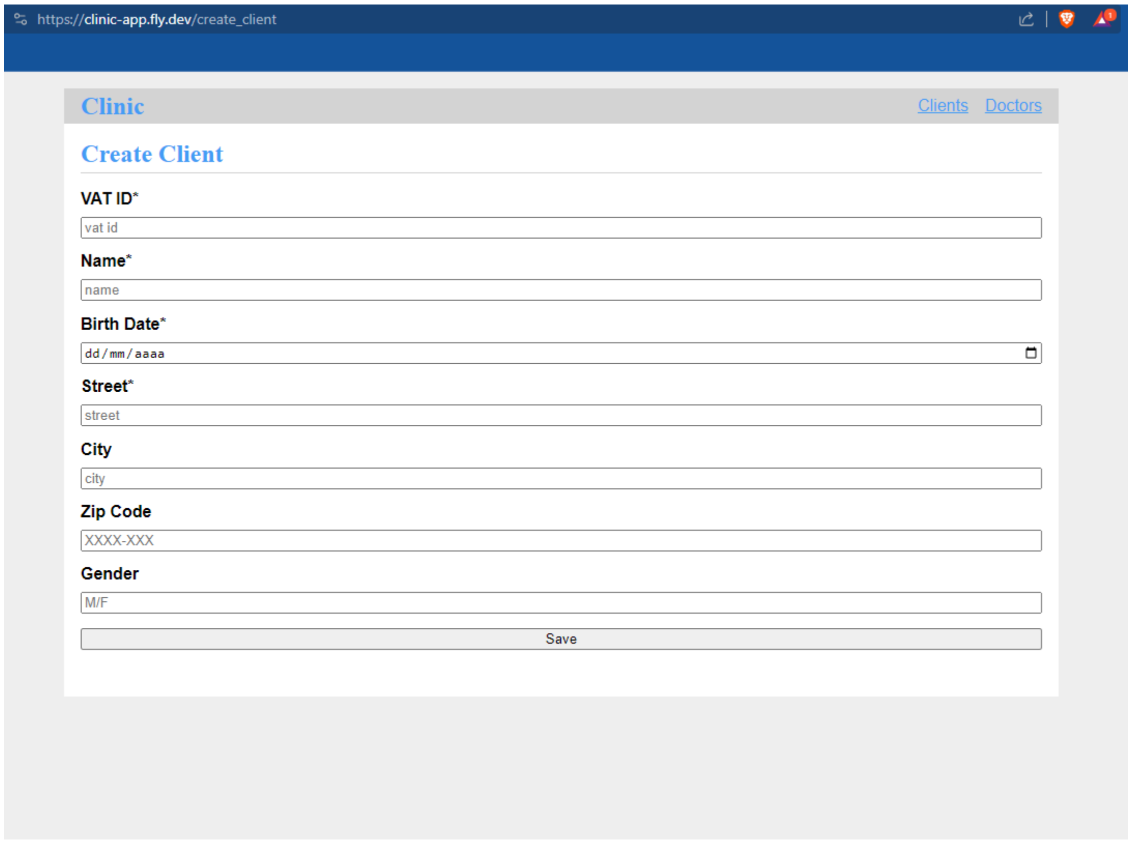
823456789
Ivy Rose
Funchal, Rua dos Pinheiros 46, 9000-000
View Profile

923456789
Jack Frost
Viseu, Estrada do Carvalhal 32, 3500-000
View Profile

624456789
Alice Waters
Guimarães, Rua do Rio 3, 4800-000
View Profile

724456789
Maxwell Bright
Santarém, Rua da Encosta 5, 2000-000
View Profile

824456789
Iris Bloom
Bragança, Rua do Jardim 8, 5300-000
View Profile


https://clinic-app.fly.dev/create_client

Clinic — Clients Doctors

Create Client

VAT ID* [vat id]

Name* [name]

Birth Date* [dd/mm/aaaa]

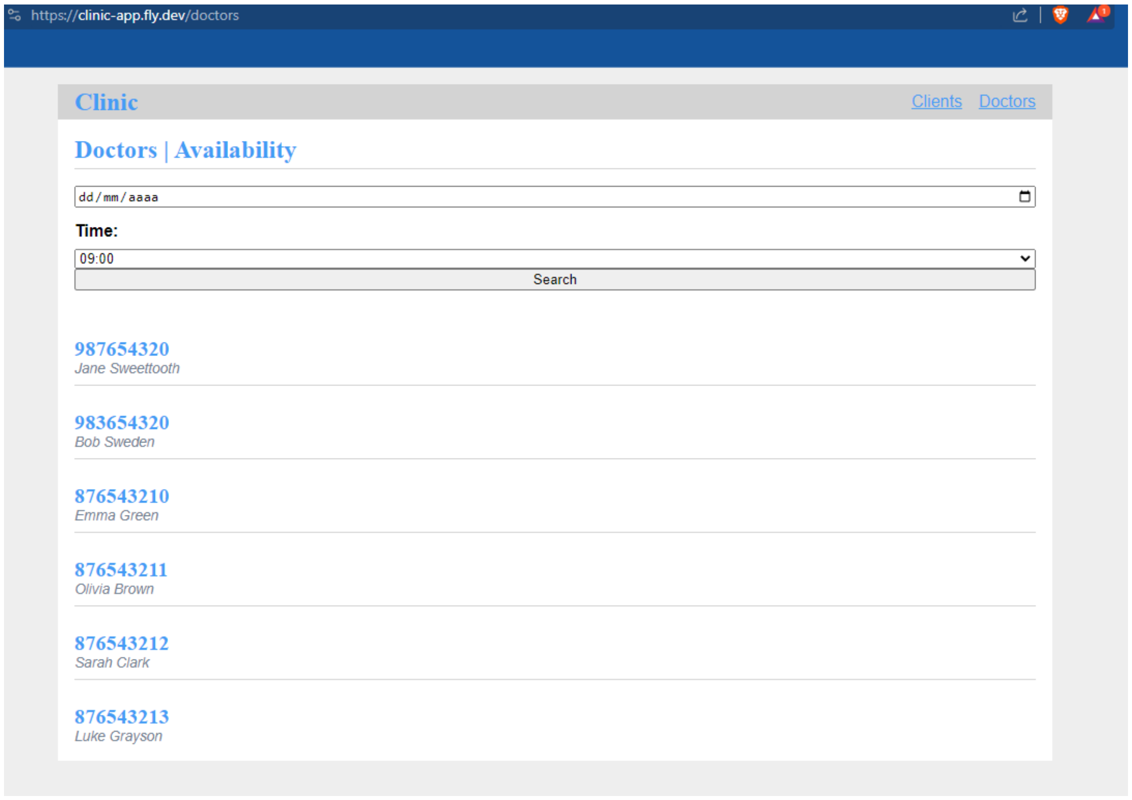
Street* [street]

City [city]

Zip Code [XXXX-XXX]

Gender [M/F]

Save


https://clinic-app.fly.dev/doctors

Clinic — Clients Doctors

Doctors | Availability

[dd/mm/aaaa]

Time: 09:00

Search

987654320
Jane Sweettooth

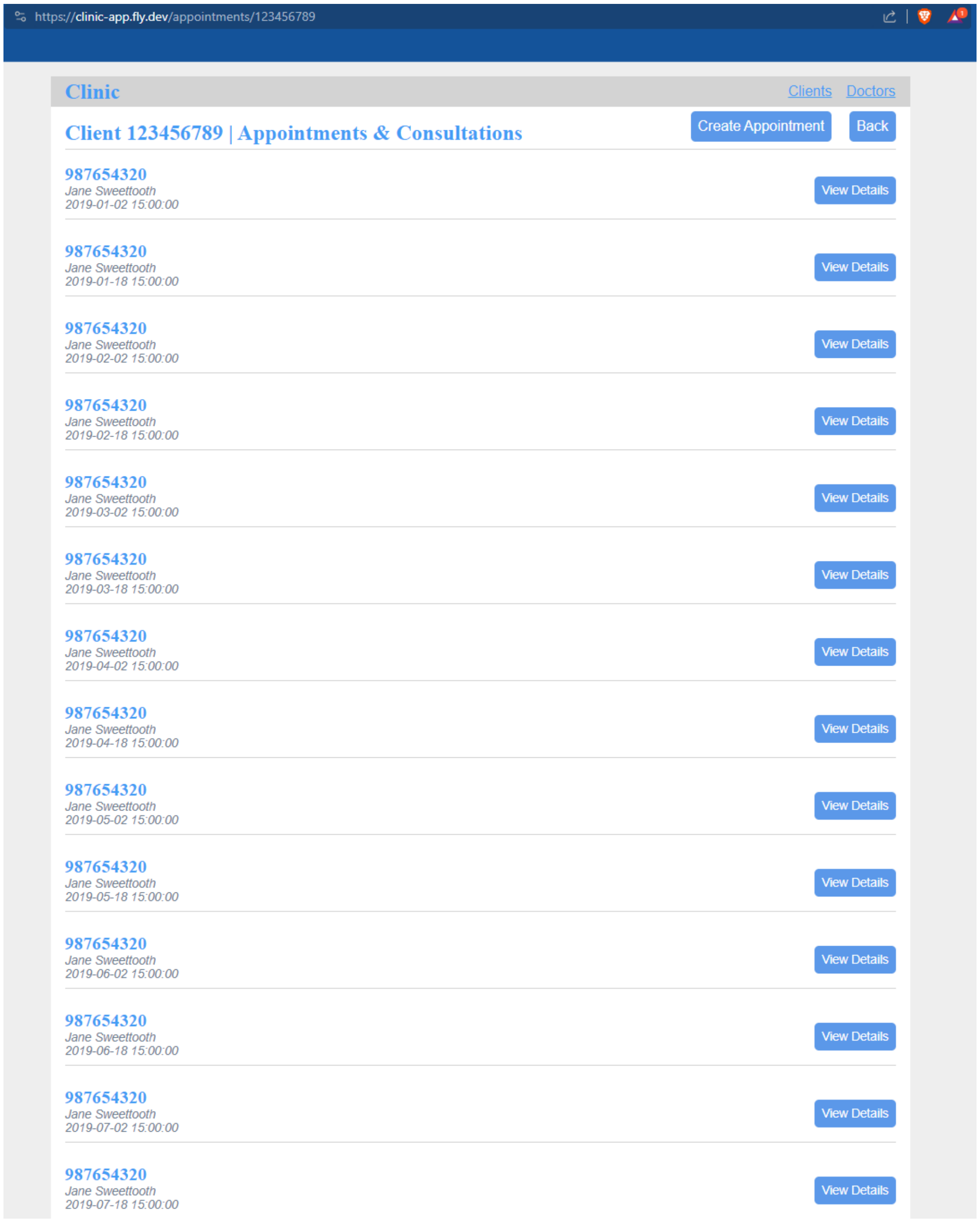
983654320
Bob Sweden

876543210
Emma Green

876543211
Olivia Brown

876543212
Sarah Clark

876543213
Luke Grayson


https://clinic-app.fly.dev/appointments/123456789

Clinic

Client 123456789 | Appointments & Consultations

987654320
Jane Sweettooth
2019-01-02 15:00:00

987654320
Jane Sweettooth
2019-01-18 15:00:00

987654320
Jane Sweettooth
2019-02-02 15:00:00

987654320
Jane Sweettooth
2019-02-18 15:00:00

987654320
Jane Sweettooth
2019-03-02 15:00:00

987654320
Jane Sweettooth
2019-03-18 15:00:00

987654320
Jane Sweettooth
2019-04-02 15:00:00

987654320
Jane Sweettooth
2019-04-18 15:00:00

987654320
Jane Sweettooth
2019-05-02 15:00:00

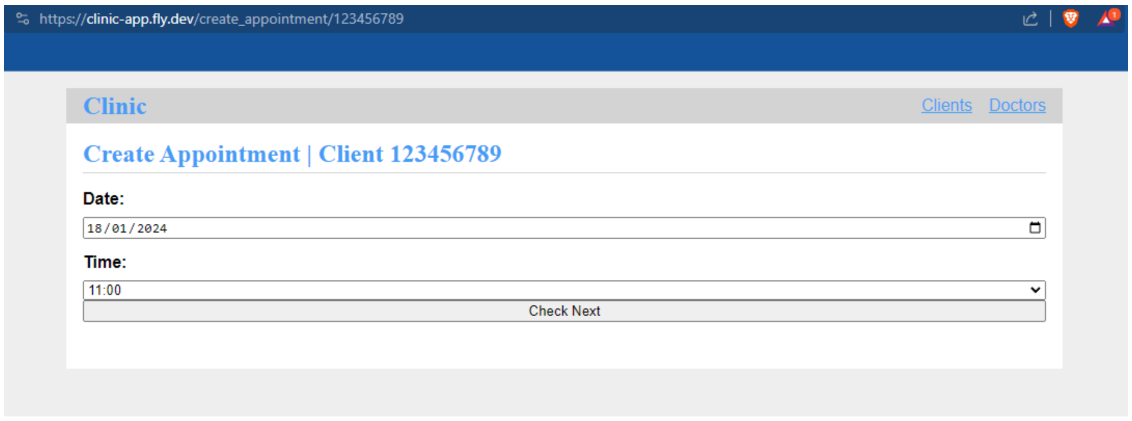
987654320
Jane Sweettooth
2019-05-18 15:00:00

987654320
Jane Sweettooth
2019-06-02 15:00:00

987654320
Jane Sweettooth
2019-06-18 15:00:00

987654320
Jane Sweettooth
2019-07-02 15:00:00

987654320
Jane Sweettooth
2019-07-18 15:00:00


https://clinic-app.fly.dev/create_appointment/123456789

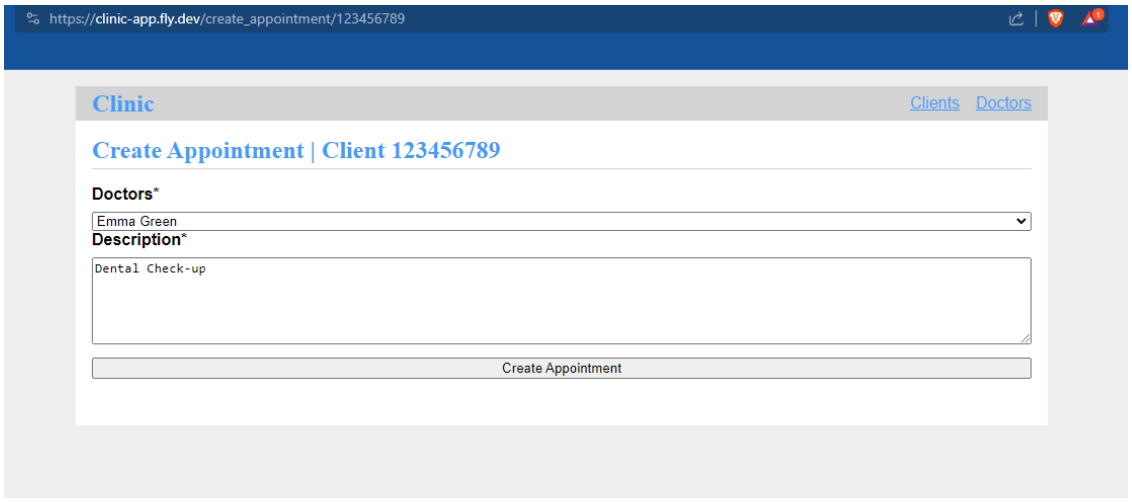
Clinic — Clients Doctors

Create Appointment | Client 123456789

Date:
18/01/2024

Time:
11:00

Check Next


https://clinic-app.fly.dev/create_appointment/123456789

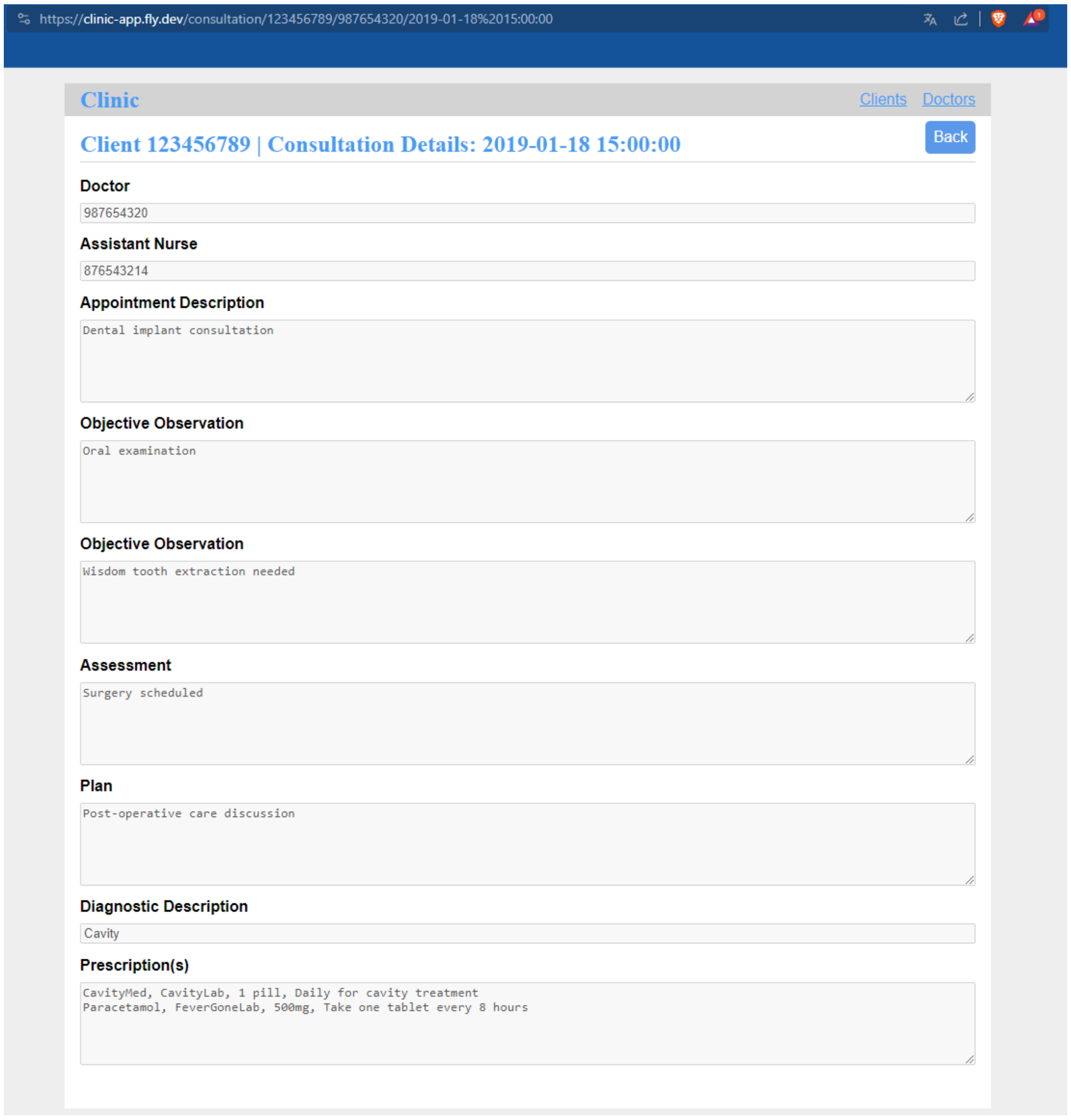
Clinic — Clients Doctors

Create Appointment | Client 123456789

Doctors*
Emma Green

Description*
Dental Check-up

Create Appointment


https://clinic-app.fly.dev/consultation/123456789/987654320/2019-01-18%2015:00:00

Clinic

Clients Doctors

Client 123456789 | Consultation Details: 2019-01-18 15:00:00

Back

Doctor

987654320

Assistant Nurse

876543214

Appointment Description

Dental implant consultation

Objective Observation

Oral examination

Objective Observation

Wisdom tooth extraction needed

Assessment

Surgery scheduled

Plan

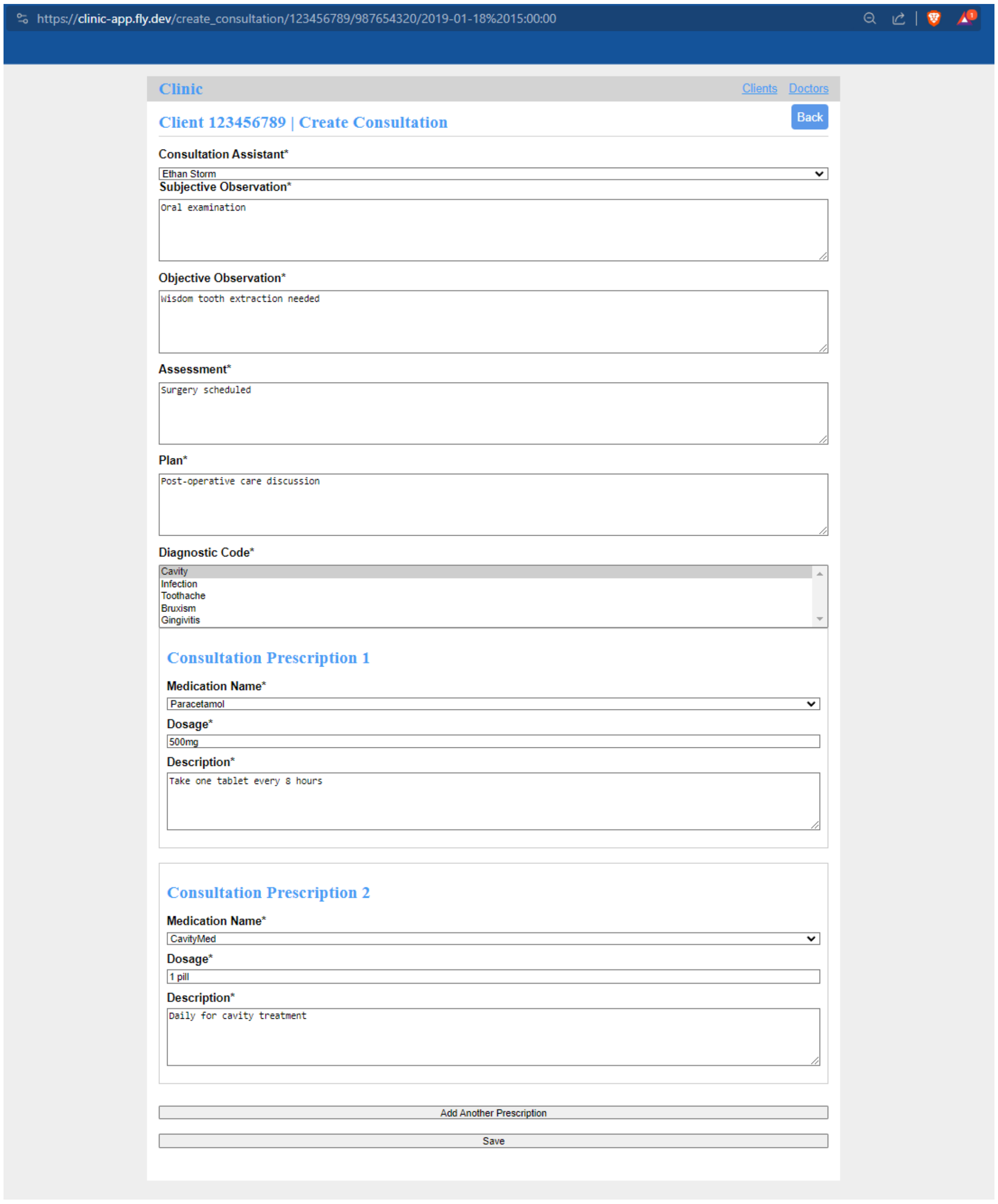
Post-operative care discussion

Diagnostic Description

Cavity

Prescription(s)

```
CavityMed, CavityLab, 1 pill, Daily for cavity treatment
Paracetamol, FeverGoneLab, 500mg, Take one tablet every 8 hours
```



https://clinic-app.fly.dev/create_consultation/123456789/987654320/2019-01-18%2015:00:00

Clinic

Clients Doctors

Client 123456789 | Create Consultation

Back

Consultation Assistant*

Ethan Storm

Subjective Observation*

```
Oral examination
```

Objective Observation*

```
Wisdom tooth extraction needed
```

Assessment*

```
Surgery scheduled
```

Plan*

```
Post-operative care discussion
```

Diagnostic Code*

Cavity
Infection
Toothache
Bruxism
Gingivitis

Consultation Prescription 1

Medication Name*

Paracetamol

Dosage*

500mg

Description*

```
Take one tablet every 8 hours
```

Consultation Prescription 2

Medication Name*

CavityMed

Dosage*

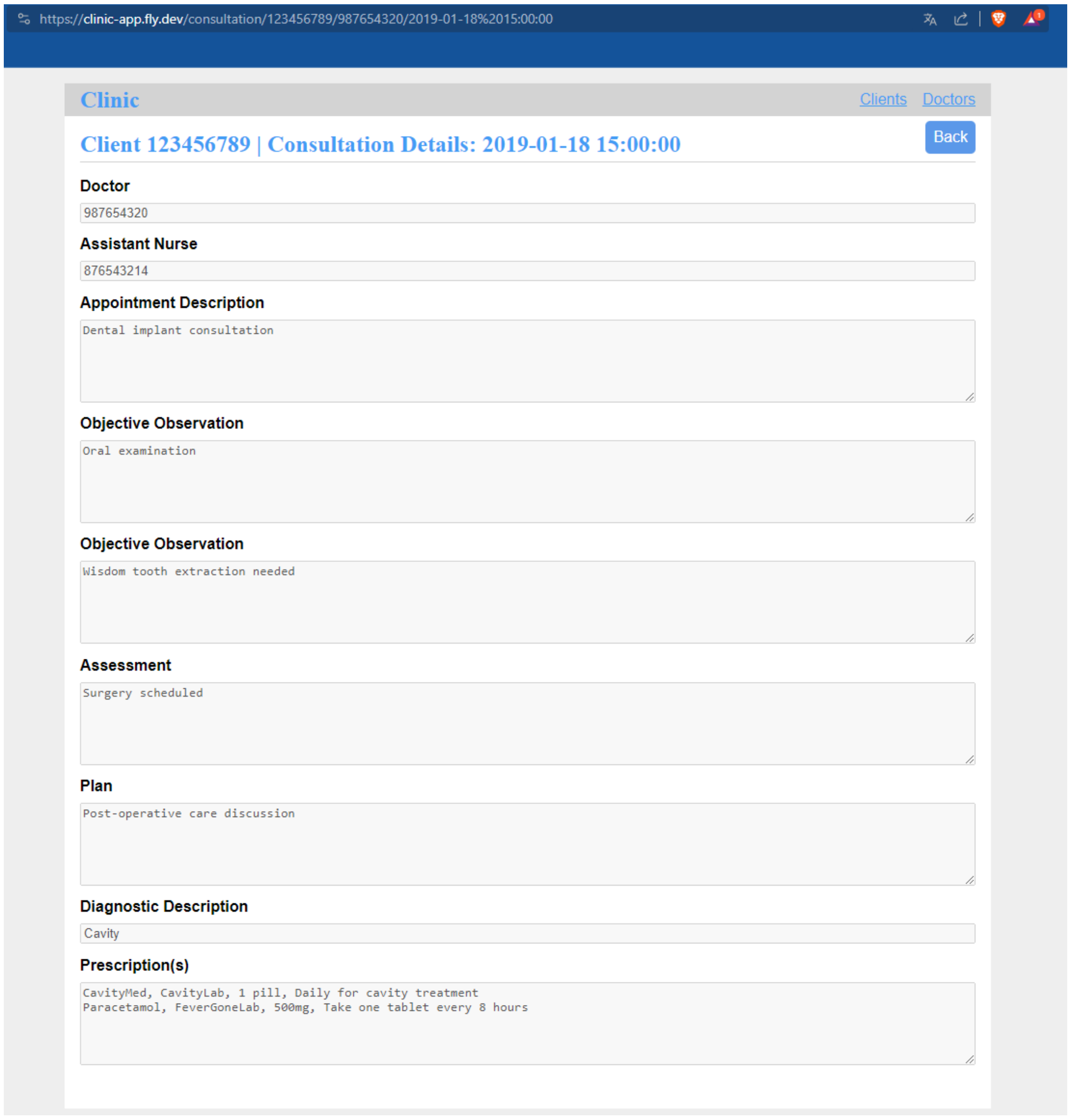
1 pill

Description*

```
Daily for cavity treatment
```

Add Another Prescription

Save


https://clinic-app.fly.dev/consultation/123456789/987654320/2019-01-18%2015:00:00

Client 123456789 | Consultation Details: 2019-01-18 15:00:00

Doctor

987654320

Assistant Nurse

876543214

Appointment Description

Dental implant consultation

Objective Observation

Oral examination

Objective Observation

Wisdom tooth extraction needed

Assessment

Surgery scheduled

Plan

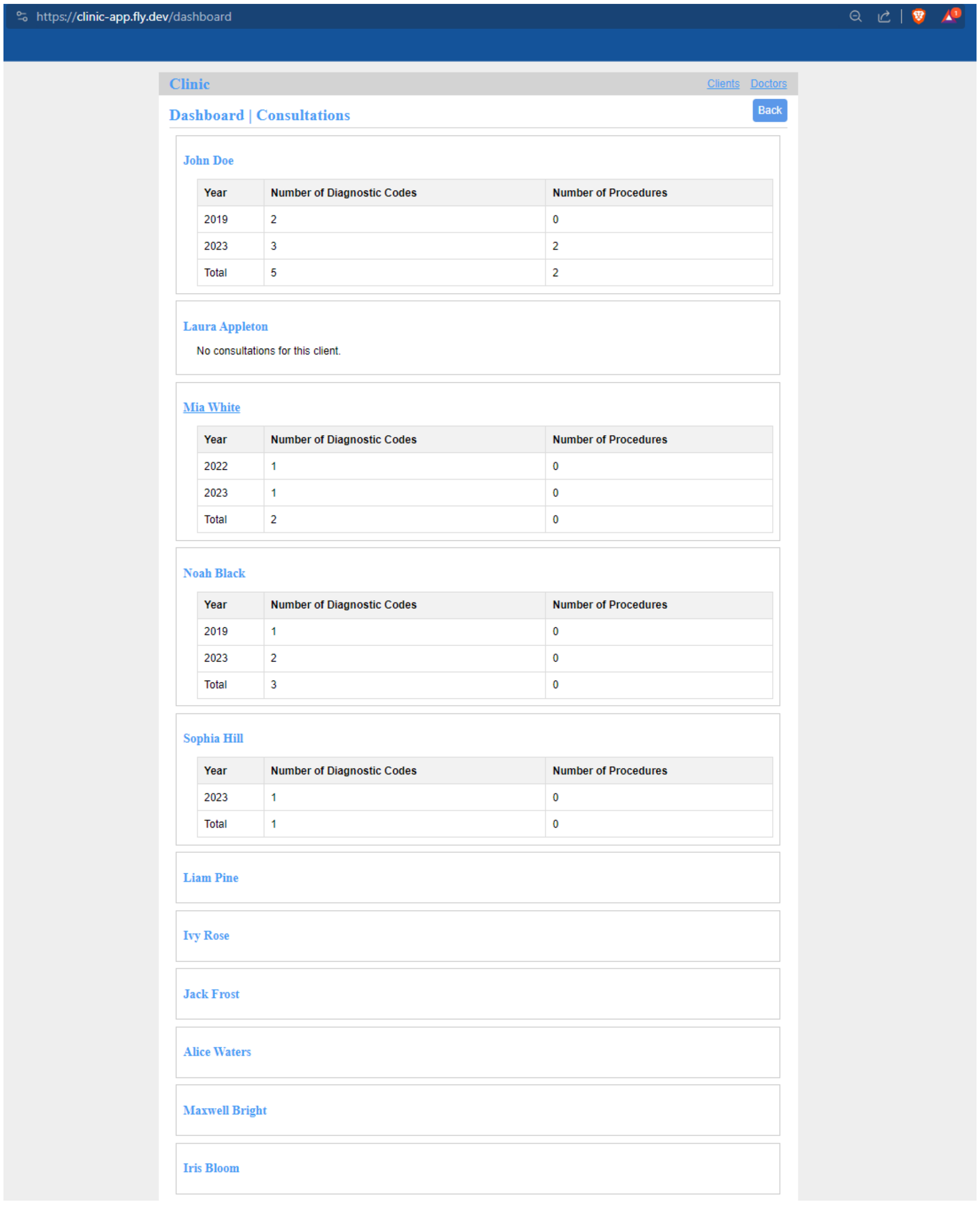
Post-operative care discussion

Diagnostic Description

Cavity

Prescription(s)

```
CavityMed, CavityLab, 1 pill, Daily for cavity treatment
Paracetamol, FeverGoneLab, 500mg, Take one tablet every 8 hours
```



https://clinic-app.fly.dev/dashboard

Dashboard | Consultations

John Doe

Year	Number of Diagnostic Codes	Number of Procedures
2019	2	0
2023	3	2
Total	5	2

Laura Appleton

No consultations for this client.

Mia White

Year	Number of Diagnostic Codes	Number of Procedures
2022	1	0
2023	1	0
Total	2	0

Noah Black

Year	Number of Diagnostic Codes	Number of Procedures
2019	1	0
2023	2	0
Total	3	0

Sophia Hill

Year	Number of Diagnostic Codes	Number of Procedures
2023	1	0
Total	1	0

Liam Pine

Ivy Rose

Jack Frost

Alice Waters

Maxwell Bright

Iris Bloom