Summary of the code in app.py:

Import necessary modules:

Flask from flask for creating the web application

SQLAlchemy from flask_sqlalchemy for working with the database

render_template, request, redirect from flask for rendering templates, handling HTTP requests, and redirecting

Migrate from flask_migrate for database migrations

psycopg2 for connecting to the PostgreSQL database

Create the Flask application:

Initialize the Flask app using Flask(__name__)

Set the SQLALCHEMY_DATABASE_URI configuration for connecting to the PostgreSQL database

Define the database models:

Customer model with columns: customer_id, name, contact_details, account_balance

RealEstateAgent model with columns: agent_id, name, contact_details

RealEstate model with columns: property_id, title, location, price, listing_date, sell_date, customer_id, agent_id

Initialize Flask-Migrate:

Create a Migrate object with the Flask app and the SQLAlchemy database instance

Define routes and view functions:

Define the home route (/) to render the index.html template

Implement CRUD routes for customers (/customers) with view functions for displaying the list of customers and creating new customers

Implement CRUD routes for real estate agents (/agents) with view functions for displaying the list of agents and creating new agents

Implement CRUD routes for real estates (/realestates) with view functions for displaying the list of real estates and creating/updating/deleting real estates

Implement routes and view functions for updating and deleting customers, agents, and real estates

Implement report routes and view functions:

Define routes for generating reports on customer property detail and listing details

Implement the view functions for generating and displaying the reports by connecting to the PostgreSQL database using psycopg2

Run the application:

Start the Flask development server with app.run() if the script is executed directly

This summarizes the main functionality of the app.py file, including the database models, CRUD operations for customers, real estate agents, and real estates, and report generation.

summary of the HTML templates used:

==========

index.html: It is the landing page of the application and contains links to the lists of real estates, customers, and agents.

customers.html: It displays a list of customers and their details.

create_customer.html: It provides a form to create a new customer.

update_customer.html: It provides a form to update the details of an existing customer.

agents.html: It displays a list of real estate agents and their details.

create_agent.html: It provides a form to create a new real estate agent.

update_agent.html: It provides a form to update the details of an existing real estate agent.

realestates.html: It displays a list of real estates and their details.

create_real_estate.html: It provides a form to create a new real estate property.

update_real_estate.html: It provides a form to update the details of an existing real estate property.

delete_real_estate.html: It displays a confirmation message for deleting a real estate property.

delete_customer.html: It displays a confirmation message for deleting a customer.

delete_agent.html: It displays a confirmation message for deleting a real estate agent.

report/customer_property_detail_report.html: It provides a form to generate a report for customer property details.

report/listing_details_report.html: It provides a form to generate a report for listing details.

These templates are rendered by the Flask application based on the corresponding routes and view functions defined in app.py.

Project app.py codes

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask import Flask, render_template, request, redirect
from flask_migrate import Migrate
```

```
from flask import Flask, render_template, request
import psycopg2

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql://postgres:Supergmat123#@localhost/estate'
db = SQLAlchemy(app)

# Initialize Flask-Migrate
migrate = Migrate(app, db)
# Customer model
class Customer(db.Model):
    __tablename__ = 'customer'
    customer_id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    contact_details = db.Column(db.String(100))
    account_balance = db.Column(db.Numeric(10, 2))

# RealEstateAgent model
class RealEstateAgent(db.Model):
    __tablename__ = 'real_estate_agent'
    agent_id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    contact_details = db.Column(db.String(100))

# RealEstate model
class RealEstate(db.Model):
    property_id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100))
    location = db.Column(db.String(100))
```

```
    price = db.Column(db.Numeric(10, 2))
    listing_date = db.Column(db.Date)
    sell_date = db.Column(db.Date)
    customer_id = db.Column(db.Integer, db.ForeignKey('customer.customer_id'))
    agent_id = db.Column(db.Integer, db.ForeignKey('real_estate_agent.agent_id'))

    customer = db.relationship('Customer', backref='real_estate', lazy=True)
    agent = db.relationship('RealEstateAgent', backref='real_estate', lazy=True)

# Home route
@app.route('/')
def index():
    # return 'Welcome to the Real Estate Application'
    return render_template('index.html')

# Customer CRUD routes
@app.route('/customers', methods=['GET'])
def customers():
    all_customers = Customer.query.all()
    return render_template('customers.html', customers=all_customers)

@app.route('/customers/create', methods=['GET', 'POST'])
def create_customer():
    if request.method == 'POST':
        name = request.form['name']
        contact_details = request.form['contact_details']
        account_balance = request.form['account_balance']
        new_customer = Customer(name=name, contact_details=contact_details,
account_balance=account_balance)
```

```
        db.session.add(new_customer)
        db.session.commit()
        return redirect('/customers')
    else:
        return render_template('create_customer.html')

# Add routes and view functions for update and delete operations for customers


# new entry
@app.route('/customers/<action>', methods=['GET', 'POST'])
@app.route('/customers/<action>/<int:customer_id>', methods=['GET', 'POST'])
def manage_customer(action, customer_id=None):
    if action == 'create':
        if request.method == 'POST':
            # Retrieve the data from the form
            name = request.form['name']
            contact_details = request.form['contact_details']
            account_balance = request.form['account_balance']

            # Create a new customer object
            new_customer = Customer(name=name, contact_details=contact_details,
account_balance=account_balance)

            # Add the customer to the database
            db.session.add(new_customer)
            db.session.commit()

            # Redirect to the customer list page
            return redirect('/customers')
```

```
    # Render the create customer form
    return render_template('create_customer.html')

elif action == 'update':
    # Retrieve the customer with the given customer_id from the database
    customer = Customer.query.get_or_404(customer_id)

    if request.method == 'POST':
        # Retrieve the updated data from the form
        name = request.form['name']
        contact_details = request.form['contact_details']
        account_balance = request.form['account_balance']

        # Update the customer data
        customer.name = name
        customer.contact_details = contact_details
        customer.account_balance = account_balance

        # Commit the changes to the database
        db.session.commit()

        # Redirect to the customer list page
        return redirect('/customers')

    # Render the customer update form
    return render_template('update_customer.html', customer=customer)

elif action == 'delete':
```

```
        # Retrieve the customer with the given customer_id from the database
        customer = Customer.query.get_or_404(customer_id)

        # Delete the customer from the database
        db.session.delete(customer)
        db.session.commit()

        # Redirect to the customer list page
        return redirect('/customers')

    else:
        # Invalid action, redirect to the customer list page
        return redirect('/customers')
#####

# Real Estate Agent CRUD routes
@app.route('/agents', methods=['GET'])
def agents():
    all_agents = RealEstateAgent.query.all()
    return render_template('agents.html', agents=all_agents)

@app.route('/agents/create', methods=['GET', 'POST'])
def create_agent():
    if request.method == 'POST':
        name = request.form['name']
        contact_details = request.form['contact_details']
        new_agent = RealEstateAgent(name=name, contact_details=contact_details)
        db.session.add(new_agent)
        db.session.commit()
```

```
        return redirect('/agents')
    else:
        return render_template('create_agent.html')

# Add routes and view functions for update and delete operations for agents

# Real Estate CRUD routes
@app.route('/realestates', methods=['GET'])
def real_estates():
    all_real_estates = RealEstate.query.all()
    return render_template('realestates.html', real_estates=all_real_estates)

@app.route('/realestates/create', methods=['GET', 'POST'])
def create_real_estate():
    if request.method == 'POST':
        title = request.form['title']
        location = request.form['location']
        price = request.form['price']
        listing_date = request.form['listing_date']
        sell_date = request.form['sell_date']
        agent_id = request.form['agent_id']
        customer_id = request.form['customer_id']
        new_real_estate = RealEstate(title=title, location=location, price=price,
                                     listing_date=listing_date, sell_date=sell_date,
                                     agent_id=agent_id, customer_id=customer_id)
        db.session.add(new_real_estate)
        db.session.commit()
        return redirect('/realestates')
    else:
```

```
    agents = RealEstateAgent.query.all()
    customers = Customer.query.all()
    return render_template('create_real_estate.html', agents=agents, customers=customers)

# Add routes and view functions for update and delete operations for real estates

@app.route('/realestates/update/<int:property_id>', methods=['GET', 'POST'])
def update_real_estate(property_id):
    real_estate = RealEstate.query.get(property_id)
    if not real_estate:
        return "Real Estate not found."

    if request.method == 'POST':
        # Retrieve the form data
        title = request.form['title']
        location = request.form['location']
        price = request.form['price']
        listing_date = request.form['listing_date']
        sell_date = request.form['sell_date']

        # Update the database
        real_estate.title = title
        real_estate.location = location
        real_estate.price = price
        real_estate.listing_date = listing_date
        real_estate.sell_date = sell_date

        db.session.commit()
```

```
        # Redirect to the real estates page or display a success message
        # flash('Real estate entry updated successfully', 'success')
        return redirect('/realestates')

    return render_template('update_real_estate.html', real_estate=real_estate)


@app.route('/realestates/delete/<int:property_id>', methods=['GET', 'POST'])
def delete_real_estate(property_id):
    real_estate = RealEstate.query.get(property_id)
    if not real_estate:
        return "Real Estate not found."

    if request.method == 'POST':
        # Delete the entry from the database
        db.session.delete(real_estate)
        db.session.commit()

        # Redirect to the real estates page or display a success message
        # flash('Real estate entry deleted successfully', 'success')
        return redirect('/realestates')

    return render_template('delete_real_estate.html', real_estate=real_estate)


# all other CRUD RULE
# Customer CRUD routes
```

```
@app.route('/customers/update/<int:customer_id>', methods=['GET', 'POST'])
def update_customer(customer_id):
    customer = Customer.query.get(customer_id)
    if not customer:
        return "Customer not found."

    if request.method == 'POST':
        # Retrieve the form data
        name = request.form['name']
        contact_details = request.form['contact_details']
        account_balance = request.form['account_balance']

        # Update the database
        customer.name = name
        customer.contact_details = contact_details
        customer.account_balance = account_balance

        db.session.commit()

        return redirect('/customers')

    return render_template('update_customer.html', customer=customer)

@app.route('/customers/delete/<int:customer_id>', methods=['GET', 'POST'])
def delete_customer(customer_id):
    customer = Customer.query.get(customer_id)
    if not customer:
        return "Customer not found."
```

```
    if request.method == 'POST':
        # Delete the customer from the database
        db.session.delete(customer)
        db.session.commit()

        return redirect('/customers')

    return render_template('delete_customer.html', customer=customer)


# Real Estate Agent CRUD routes

@app.route('/agents/update/<int:agent_id>', methods=['GET', 'POST'])
def update_agent(agent_id):
    agent = RealEstateAgent.query.get(agent_id)
    if not agent:
        return "Agent not found."

    if request.method == 'POST':
        # Retrieve the form data
        name = request.form['name']
        contact_details = request.form['contact_details']

        # Update the database
        agent.name = name
        agent.contact_details = contact_details

        db.session.commit()
```

```
        return redirect('/agents')

    return render_template('update_agent.html', agent=agent)

@app.route('/agents/delete/<int:agent_id>', methods=['GET', 'POST'])
def delete_agent(agent_id):
    agent = RealEstateAgent.query.get(agent_id)
    if not agent:
        return "Agent not found."

    if request.method == 'POST':
        # Delete the agent from the database
        db.session.delete(agent)
        db.session.commit()

        return redirect('/agents')

    return render_template('delete_agent.html', agent=agent)

# End of other CRUD

# Report section

# Route for the customer property detail report
# ...

# Route for the customer property detail report
@app.route('/report/customer_property_detail', methods=['GET', 'POST'])
def customer_property_detail_report():
```

```
    if request.method == 'POST':
        customer_name = request.form['customer_name']
        connection = psycopg2.connect(
            user='postgres',
            password='Supergmat123#',
            host='localhost',
            port='5432',
            database='estate'
        )
        cursor = connection.cursor()
        query = "SELECT a.name, b.title FROM customer a, real_estate b WHERE a.customer_id = b.customer_id AND name LIKE '%{}%'".format(customer_name)
        cursor.execute(query)
        results = cursor.fetchall()
        cursor.close()
        connection.close()
        return render_template('report/customer_property_detail_report.html', results=results)
    return render_template('report/customer_property_detail_report.html')


# Route for the listing details report
@app.route('/report/listing_details', methods=['GET', 'POST'])
def listing_details_report():
    if request.method == 'POST':
        start_date = request.form['start_date']
        end_date = request.form['end_date']
        connection = psycopg2.connect(
            user='postgres',
            password='Supergmat123#',
            host='localhost',
```

```
        port='5432',
        database='estate'
    )
    cursor = connection.cursor()
    query = "SELECT * FROM real_estate WHERE listing_date BETWEEN '{}' AND '{}'".format(start_date, end_date)
    cursor.execute(query)
    results = cursor.fetchall()
    cursor.close()
    connection.close()
    return render_template('report/listing_details_report.html', results=results)
  return render_template('report/listing_details_report.html')


#


if __name__ == '__main__':
  app.run()
```