

|  |  |
| --- | --- |
| Student name: | Aiya Zeinulla |
| Group name: | Se-2008 |
| Teacher name: | Shynggys Alshynov |
| Project name: | Pet Store |
| Project type: | Group |
| Number of words | 4132 |

**INTRODUCTION**

What is a pet store? In our usual sense, this is a retail business that sells various types of animals. Pet stores also sell a variety of pet supplies and accessories for Pets. Products sold include food, treats, toys, collars, leashes, cat litter, cages, and aquariums.

Pet stores are extremely popular in modern society. In 2004, sales of live Pets in the pet industry reached approximately $ 1.6 billion, according to the American pet supplies Association.

IMPORTANCE OF THE PET STORE

In Kazakhstan, as in other countries, online stores are developing well. Pet stores are also gaining momentum. Owning a pet store can be a profitable business regardless of the performance of the larger economy. Pet stores have shown revenue growth even during recessions, since the owners have reduced their own costs before you can reduce the standard of living of their Pets. However, competition from large retailers means that a small business owner still needs to keep a close eye on their costs to join the ranks of profitable ones.

Advantages

Live Pets: pet stores, depending on whether they contain live animals, can offer a wide selection of small Pets from rodents to reptiles, as well as birds and fish. Many store chains have rules about quarantining new or sick animals to ensure that only healthy animals are offered for sale. Guarantees are usually offered so that buyers can return the animal if it becomes ill within a short time after purchase. Many stock up on informative brochures to advise owners on the proper care of a new pet, as well as stock up on all the necessary supplies.

Quality: pet stores usually offer a wide range of products to appeal to discerning customers. Some specialize in natural products, recognizing the market that has emerged because pet parents don't want to feed their animals artificial ingredients. Some keep lower-quality products on their shelves to attract customers who are just looking for lower prices.

WHY YOU NEED OUR PROJECT?

Many people want to buy a pet. They are excellent companions and those who can become kindred spirits with their owners and be their real support and support, are also able to relieve stress and brighten up loneliness. What's more, people living in the city are more lonely these days than they were a few decades ago. Most of them experience work stress, their problems in communicating with other people, and all this will make them feel lonely and can not get rid of their stress and loneliness. Having a pet, such as a dog or cat, can help them be more fun in their lives and easily connect with other people.

But sometimes people do not get Pets for simple reasons: they can not find a pedigree pet seller (for principled lovers), due to the busy schedule of work, study and daily routine, they can not find time to buy a pet, and much more.

For example, having a dog is a good choice for most people who want to have a pet. The dog is not only a very good companion, but also has deep emotional connections between people.

Having a pet helps the owner stay connected to social life. The owner must bring their pet for a walk, take it with them to visit the zoo, and pass a veterinary examination for their pet. All this is necessary for the owner to communicate with others and involve himself in social activities. In addition, people can make friends with those who have the same habit as their pet, go out for some exercise and activities.

We want to create a program that can develop into a full-fledged startup project in the future, in which we will create an application or website that can connect the buyer with sellers through our pet store. That is, let's say the buyer goes to the site or opens the interface of our program in which he leaves an ad that he wants to buy a certain pet, on our part we check the presence of a pet or find sellers who can sell it, arrange delivery if necessary, as well as provide various products for caring for the pet, food, toys and everything for the comfort of the pet. We can be called an ordinary pet store, but this is not the case. we will help customers buy the desired Pets without wasting precious time for them and will provide everything necessary for the pet and all this will happen in electronic format, so to speak, without leaving home.

Now we are only first-year students and we can't fully implement this idea, but now we are able to write a very simple program that will only be a prototype and template for a future startup.

**MAIN PART**

ANALYSIS OF MARKET, ANALYSIS OF TECHNOLOGIES, TECHNOLOGIES WE HAVE USED TO FULFILL THE TASK.

Since this is a startup project, the exact same technology is not available now.

Now online stores are very popular, millions of people make purchases every day, especially during the quarantine period. I think in this situation there are many people who want to buy Pets.

Modern online stores have mostly colorful, multi-functional mobile applications (for quick access and ease of use) and websites.

Almost any popular online store can be a good example for us and show the proven process of work.

We can't write an app or a website for our store yet, but we can write a simple console application using C++ and SQL programming languages. We integrate two programming languages into one another, SQL into C++, and into a regular programming language a database.

Since we learned the C++ and SQL programming languages in our first trimester at the University, we decided to create a project using these programming languages in our practical training project with the head of the training practice. but later, unfortunately, we had some problems integrating these languages. The fact is that it is quite difficult to connect a special library for working with databases in C++. it was extreme. it is difficult to download and run a special file for this, many who downloaded it could not get this library to work and run it in the compiler, many did not manage to download the library at all. For Mac users, it wasn't even possible to download the file at all. therefore, it was decided that we will switch from the C++ programming language to the Python programming language. in terms of working with databases, Python is much more flexible. since C++ is a more complex language than Python, we managed to switch from one programming language to another. In Python, we downloaded special idle to work in this programming language, where we wrote the main code and compiled it. During the practical classes, our practice coordinator explained to us how to write the beginning of our future console application and explained the basic syntax, after which we wrote the main part of the program ourselves.

**Description of the program workflow**

Now we want to briefly describe how our program works.

First of all the user opening the console in our program must register if it is not registered; the second step after registration, the user logs into your new account; then, before it displays the main menu of the shop where he chooses that he wants to buy; after selecting a specific option, opens a list of currently existing products and the buyer again chooses what wishes to buy; after you select all desired, the buyer can return to the main menu and see a list of his shopping or leave the store.

Now let's look at everything in more detail.

First, we need a database from where we will get our data in Python. For this, we drew a diagram on the Lucid website with all the entities and keys. It did not take us much time for this, although it was sometimes difficult to catch the relationship between the tables. For the first time, our diagram consisted of 9 tables, but over time we decided to reduce it to 4. Since I wrote a well-thought-out one for 9 tables, and for 4 tables we got a small but practical program. Below you can see our final diagram:

Diagram

Description automatically generated

Next, we had to fill in the database using this diagram. We have filled it in on the PostgreSQL website. It was not that difficult since we have a diagram as a guide. Then this is how we wrote our code in PosgreSQL:

CREATE TABLE users

(

user\_id SERIAL PRIMARY KEY,

user\_login varchar(255) unique,

user\_name varchar(255),

user\_surname varchar(255),

user\_password varchar(255),

user\_address varchar(255),

user\_phone varchar(255)

);

CREATE TABLE pets(

pet\_id decimal(9),

pet\_category varchar(15),

breed varchar(30),

cost decimal(11),

quantity decimal(11),

primary key(pet\_id)

);

CREATE TABLE pet\_products(

pp\_id decimal(9),

pp\_name varchar(30),

type varchar(20),

cost decimal(11),

belongs\_to varchar(20),

quantity decimal(11),

primary key (pp\_id)

);

CREATE TABLE sold\_products(

user\_id SERIAL,

pp\_name varchar(30),

pp\_quantity serial,

pet\_name varchar(30),

pet\_quantity serial,

total serial,

foreign key(user\_id) references users(user\_id)

);

And filling our tables:

insert into pets (pet\_id, pet\_category, breed, cost, quantity) VALUES

(1, 'dog', 'labrador', 80000, 2),

(2, 'cat', 'parsian', 30000, 3),

(3, 'dog', 'golden retriever', 65000, 4),

(4, 'dog', 'boxer', 15000, 1),

(5, 'cat', 'rag doll', 35000, 5),

(6, 'dog', 'st bernard', 10500, 2),

(7, 'dog', 'bulldog', 12000, 3),

(8, 'parrot','grey parrot', 20000, 2),

(9, 'lovebirds', 'black cheeked', 18000, 1),

(10, 'lovebirds', 'grey headed', 20000, 3),

(11, 'lovebirds', 'lilian', 25000, 1),

(12, 'cockatoo', 'white cockatoo', 50000, 2);

select \* from pets;

insert into pet\_products (pp\_id, pp\_name, type, cost, belongs\_to, quantity) VALUES

(1, 'dog collar', 'accesories', 5000, 'dog', 10),

(2, 'chain', 'accesories', 1000, 'cat', 12),

(3, 'pedigree', 'food', 1500, 'dog', 30),

(4, 'mouth mask', 'accesories', 2500, 'dog', 4),

(5, 'food bowl', 'accesories', 2500, 'dog ', 5),

(6, 'bird feeds', 'food', 3000, 'birds', 20),

(7, 'bird`s cage', 'accesories', 8000, 'birds', 6);

select \* from pet\_products;

**First step: Register or log in to your account**

1)The user opens the program and a menu for registering or logging in to their account is displayed in front of them on the console.

# display menu options for users

def menu\_options(is\_authenticated, current\_user, cursor):

if not, is\_authenticated:

print (f " Welcome, {current\_user function}")

print ("1. Register", "2. Login", "3. Exit")= '\n')

2) if the user is not registered, they are asked to register by entering their username, first name, last name, asking to repeat the password twice, and then their residential address and phone number (for product delivery and communication with the buyer). After entering the required information, all user data is automatically saved in our database.

# make user registration

Def user\_registration( cursor, connect to):

form = ['username', 'first Name', 'last Name', 'password', 're-password', 'address', 'phone number'] # list of forms

user\_form = dict. fromkeys (form) # creating a dictionary from form elements

unique\_login = False # checking the user name for uniqueness

for i in the form: # run through each option in the form and fill it in

if I = = 'Login': # login verification

while not unique\_login: # loop until the user enters a unique username

user\_form[i] = input ("enter your" + i + ": \n")

for j in make\_select\_query (cursor, "select user\_name from users").fetchall ():

if user\_form. you ("login") in J:

print("this username already exists, Please choose another one!").

else: # if everything is in order, then we go to the following points of the form

unique\_login = true

still:

user\_form[i] = input ("enter your" + i + ": \n")

# taking the total number of users to create a user ID for a new user

function user\_id = int (make\_select\_query (cursor "select functions count (\*) from users").fetchall()[0][0]) + 1

# creating the request itself

query = f " insert in user values(\

{user ID},\

'{user\_form ['username']}',\

'{user\_form name ["]}',\

'{user\_form ['last name']}', \

'{user\_form['password']}',\

'{user\_form['address']}', \

'{user\_form ['phone']}' \

)"

# calling function to insert

make\_insert\_query (cursor, query, connection)

3) then, after the user has registered, they are asked to log in by entering their username and password. They are automatically checked and in case of incorrect input, the screen will display a message that "your username or password is not correct!", if this happens, the user will have to close the console and open it again, running the code again and select the log in option.

# login function

def login (cursor, connection):

user\_login = input ("enter your username:")

password = input ("Your password:")

global current\_user function

global is\_authenticated

if password = = make\_select\_query (cursor, f " select user password\_ from users, where user\_login='{user\_login}'").fetchone()[0]:

is\_authenticated = true

current\_user = user\_login function

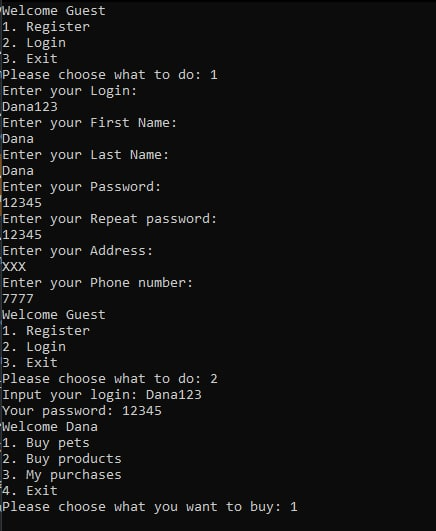
user\_id = make\_select\_query (cursor, f " select user\_id from users, where user\_login = '{current\_user}'").fetchone()[0]

make\_insert\_query (cursor, f " insert in sold\_products values ({user\_id function}, 0, 0, 0, 0, 0)", connection)

still:

print("your username or password is incorrect!")

login (cursor)



**Second step: Main menu of the store**

After logging in to your account, the user will see the main menu of the store, where they can choose which category of products they want to purchase. They can buy a pet or accessories, food and everything necessary for pet care. Also in the same menu, the buyer can see their shopping cart, that is, a list of products that they have purchased or left the store.

Comment on this section of code: in the code of our program, we used and created custom functions, which we then called in the main function.

The section of code shown below is a user-defined function that displays a menu on the console. if the user is logged in, the store menu with a list of product categories appears in front of him, otherwise the registration menu appears in the system.

# display menu options for users

def menu\_options(is\_authenticated, current\_user, cursor):

if not, is\_authenticated:

print(f " Welcome, {current\_user function}")

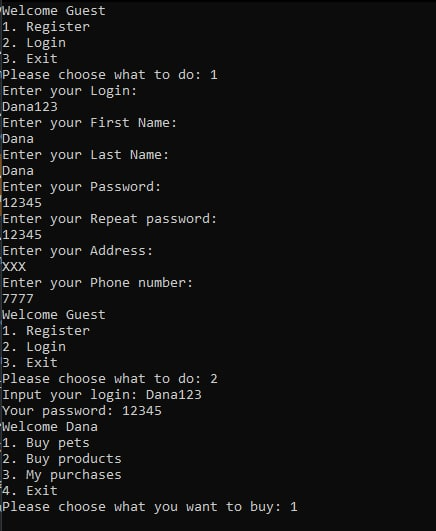
print ("1. Register", "2. Login"," 3. Logout", = '\n')

more:

user\_name = make\_select\_query (cursor, f " select user\_name from users, where user\_login = '{current\_user}'").fetchone()[0]

print (f " Welcome, {user\_name}")

print ("1. Buy Pets", "2. Buy products", "3. My purchases", "4. Exit", sep= '\n')



**Third step: selecting a specific product**

if the user selects buy a pet or buy pet products in the main menu, a list of currently existing products is displayed in front of them. If you want to buy a pet, a list of Pets appears, according to their ID number, type of pet (dog, cat, parrot...), breed, price and quantity (how many of them are currently in the store). When you select purchase pet products, a list of all products that are available in the store appears. They appear according to the ID number, product name, its type (food, collar, muzzle, cage....), for whom it is intended (for a dog, for a parrot....), price, quantity, and in the store.

After purchasing a particular product, whether it is a pet or a product for it, its quantity is automatically changed in our database according to the purchased quantity.

There are also two custom functions at the bottom for purchasing Pets or products for them.

#function for buying

Pets def buy\_pets (cursor, connection):

print ("select a pet:\N" )

cursor.run ("select \* from Pets")

because I'm in the cursor.fetchall():

print("Id = ", i[0], )

print("Pet category = ", i[1])

print("Breed = ", i[2])

print("Price = ", i[3])

print("Quantity = ", i[4], "\n")

chosen\_pet = input("select ID pet: ")

number = int(input("choose amount: "))

user\_id = make\_select\_query(cursor, f"select user\_id from users where user\_login = '{current\_user}'").fetchone()[0]

pet\_name = make\_select\_query (cursor, f " select pet\_category from animals, where pet\_id = {chosen\_pet}").fetchone()[0]

cost = make\_select\_query (cursor, f " select Cost from animals, where pet\_id = {chosen\_pet}").fetchone()[0]

if chosen\_pet = = street (make\_select\_query (cursor, f " select pet\_id from animals, where pet\_id={chosen\_pet}").fetchone()[0]):

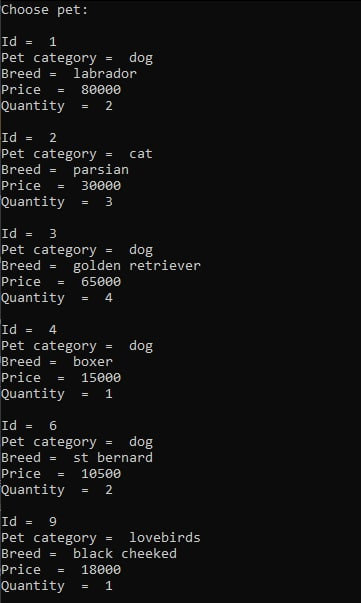
query\_for\_pets = f " update sold\_products set function user\_id = {function user\_id}, pet\_name = '{pet\_name} ' where user\_id = '{function user\_id}'"

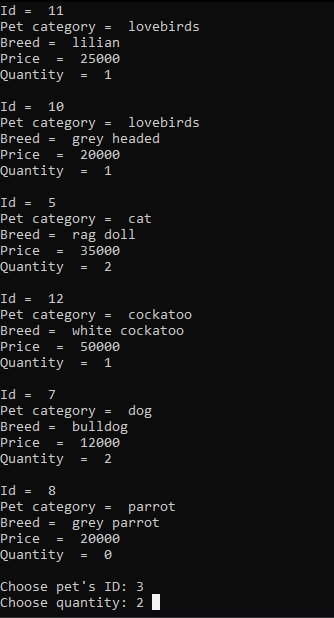
query\_for\_pets\_update = update f " home set quantity = quantity-{quantity}, where pet\_id = '{chosen\_pet}' "

make\_insert\_query(cursor, query\_for\_pets, connection)

make\_insert\_query (cursor, f " update sold\_products set pet\_quantity = pet\_quantity+{quantity}, amount = amount+({cost}\*{quantity}) Where user\_id = '{user\_id}'", connection)

make\_insert\_query(cursor, query\_for\_pets\_update, connection)





def buy\_products(cursor, connection):

print ("select product:\n")

cursor.run ("select \* from pet\_products")

because I'm in the cursor.fetchall():

print("Id = ", i[0], )

print("Product = ", i[1])

print("Type = ", i[2])

print("For = ", i[4])

print("Cost = ", i[3])

print("Quantity = ", i[5], "\n")

chosen\_product = input("choose the ID of the product: ")

quantity = int(input ("select quantity:"))

user\_id = make\_select\_query (cursor, f " select user\_id from users, where user\_login = '{current\_user}'").fetchone()[0]

pp\_name = make\_select\_query (cursor, f " select pp\_name from pet\_products where pp\_id = {chosen\_product}").fetchone()[0]

cost = make\_select\_query (cursor, f " select cost from pet\_products where pp\_id = {chosen\_product}").fetchone()[0]

if chosen\_product = = street (make\_select\_query (cursor, f " select pp\_id from pet\_products where pp\_id={chosen\_product}").fetchone()[0]):

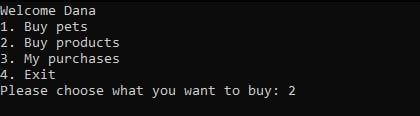
query\_for\_products = f " update sold\_products set function user\_id = {function user\_id}, pp\_name = '{pp\_name} ' where user\_id = '{function user\_id}'"

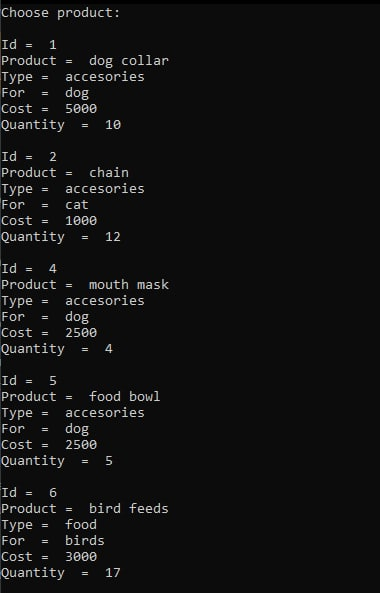
query\_for\_products\_update = F " pet\_products update set quantity = quantity-{quantity}, where pp\_id = '{chosen\_product}' "

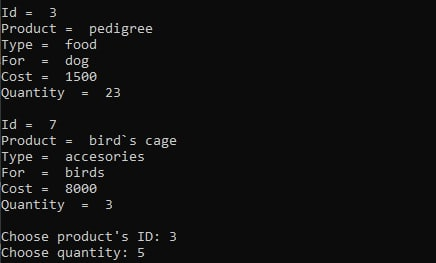
make\_insert\_query(cursor, query\_for\_products, connection)

make\_insert\_query(cursor, f" update sold\_products set pp\_quantity = pp\_quantity+{quantity}, amount = amount+({cost}\*{quantity}) Where user\_id = '{user\_id}'", connection)

make\_insert\_query(cursor, query\_for\_products\_update, connection)







**The fourth step: preview your purchased items**

If necessary or if desired, the buyer can view the list of their purchases (Pets and products together) and find out the total price. They need to go back to the main menu of the store and select the view my purchases option.to do this, they will need to log in to the program again. After selecting the desired option, the screen displays a list of all purchases, both Pets and products, their number and the total cost of all purchases.

def sold\_products (cursor):

user\_id = make\_select\_query (cursor, f " select user\_id from users, where user\_login = '{current\_user}'").fetchone()[0]

cursor#.run ("select (amount (pp\_quantity), amount (pet\_quantity), amount(amount)) from sold\_products where user\_id = ' {function user\_id}'")

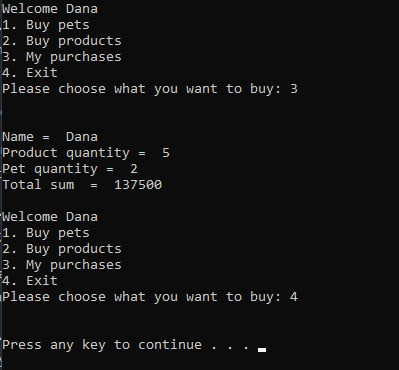
#for me in the cursor.fetchall ():

print ("name = ", make\_select\_query (cursor, f " select a user name from users, where user\_login = '{current\_user function}'").fetchone()[0])

Print ("product quantity =", make\_select\_query (cursor, f " select amount (pp\_quantity) from sold\_products where user\_id = '{function user\_id}'").fetchone()[0])

print ("pet quantity =", make\_select\_query (cursor, f " select amount (pet\_quantity) from sold\_products where user\_id = '{user\_id function}'").fetchone()[0])

print ("amount = ", make\_select\_query (cursor, f " select amount (amount) from sold\_products where user\_id = '{user\_id function}'").fetchone()[0], "\n")



**Fifth step: General program algorithm**

Before that, we described all the steps of the program, but using custom functions in the explanation, now we will look at how the entire program looks in the main function.

When the user authorization and registration menu is displayed, if the user selects the number 1, i.e. Registration, the function for registering the user user\_registration is launched

If you select the number 2-log in, the log in function starts running

If you select option 3-exit, the user leaves the program

If you enter something other than 1,2 and 3, the message "choose 1 or 2"

is displayed

# the main algorithm of the program

def main ():

connection = db\_connect ()

cursor = connection.cursor ()

and is\_authenticated = = false:

menu\_options(is\_authenticated, current\_user, cursor)

choose\_option = input("Please select what to do: ")

if choose\_option == '1':

user\_registration(cursor, connection)

elif choose\_option == '2':

logon(cursor, connection)

elif choose\_option == '3':

break

more:

print("Please select 1 or 2")

continue

Then if the user is already registered and he logged into your account by selecting the last menu option 2, appears before him the main menu of the shop where he is given the choice of categories of goods which are in stock.

When you select option 1-buy Pets, the program starts compiling the buy\_pets function

If the user selects 2-buy products, the buy\_products function is launched

Also, as we mentioned, the buyer can see a list of all the products they purchased. to do this, they need to select option 3-my purchases, and then the sold\_products function will be executed.

We want to explain that when buying certain products, their number changes in the database: it decreases in the store's availability lists and increases accordingly in the user's purchase lists. As soon as the user has purchased a certain product, its quantity is reduced in stock and added as a new item in purchases.

Most of the information in the database is filled in during the compilation of the program, and specifically all information about customers: their personal data and information about purchases, as well as data about purchased goods. From the already filled in database tables, this is all information about Pets and products that are available.

and is\_authenticated == true:

menu\_options(is\_authenticated, current\_user, cursor)

choose\_option = input("Please select what you want to buy: ")

print("\n")

if choose\_option == '1':

buy\_pets(cursor, connection)

elif choose\_option == '2':

buy\_products(cursor, connection)

elif choose\_option == '3':

sold\_products (cursor)

elif choose\_option == '4':

break

more:

print ("Please select 1 or 2")

continue

# calling the main function

if \_\_name\_ \_ = = '\_\_main\_\_':

main()

**CONCLUSION**

The world is on track to meet the goal of reducing the proportion of people who do not have the constant opportunity and time to buy at regular stores, although it is difficult to keep up with population growth and ever-accelerating urbanization. Obviously, the world will never be able and will not want to completely switch to the version of online stores with delivery, since it is possible that in some places this is not logical and will not take root economically in our realities.

However, it is clear that at least part of our world will nevertheless switch to the online version of stores, since not all people have time for the traditional method of buying something, and even if people have time for such things, not everyone wants to spend it, and here to help online stores come, such as we are planning. We can say that we have created a very convenient console application that can help people easily find the animal that a person is looking for. As we planned, we can provide a variety of pet care products, food, toys and supplies for the comfort of your pet, and we can arrange for the delivery of these products and pets as needed. However, this is still a template, but the beginning has already been laid.

We managed to achieve our main goal and created very user-friendly console application, which can help people to easily find animal which person is looking for. Our program can help in searching needed animal breed, fur, color, weight etc. As we planned we can provide various products for caring for the pet, food, toys and everything for the comfort of the pet and provide delivery of those products and pets if necessary.

Key lessons learned

A number of lessons have been learned within the sector. For example, that compiling a database and creating a working code for our online store, although this is just a template, we realized that it is not as easy as it seems at first glance, it takes a lot of time and by trial and error we still managed to get it the current result.

Possible goals achievement

After completing our template and working in general, our team had new ideas on how to further develop the facilitation of online shopping and we hope that we can implement them in the near future.

It was a wonderful learning experience for us while working on this project. This project took us through the various phases of project development and gave us real insight into the world of software engineering. The joy of working and the thrill involved while tackling the various problems and challenges gave us a feel of the developers’ industry.

**REFERENCES**

<https://lucid.app/lucidchart/7507182f-da1b-4b89-ba9e-a720693e525c/edit?shared=true&page=0_0#?folder_id=home&browser=icon>

<https://github.com/>

<https://www.postgresql.org/>

<https://python-scripts.com/database>

<https://pynative.com/python-postgresql-insert-update-delete-table-data-to-perform-crud-operations/>

<https://postgresqltutorial.com/postgresql-python/update/>

<https://stackoverflow.com/questions/7458749/psycopg2-insert-update-writing-problem>

**GLOSSARY:**

• ERD Diagram.

Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. An ERD contains different symbols and connectors that visualize two important information: The major entities within the system scope, and the inter-relationships among these entities.

• Function

In Python, a function is a group of related statements that performs a specific task.

Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable. Furthermore, it avoids repetition and makes the code reusable.

• PostgreSQL

PostgreSQL also known as Postgres, is a free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. It was originally named POSTGRES, referring to its origins as a successor to the Ingres database developed at the University of California, Berkeley. In 1996, the project was renamed to PostgreSQL to reflect its support for SQL.

• Psycopg2

Psycopg2 is the most popular PostgreSQL database adapter for the Python programming language.